Ondersteuning door informatiesystemen in het architectuurontwerp

Reconsidering Information System Support
for Architectural Design Thinking

Pieter Pauwels

Promotoren: prof. dr. R. De Meyer, prof. dr. ir. J. Van Campenhout
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Architectuur

Vakgroep Architectuur en Stedenbouw
Voorzitter: prof. dr. ir. -architect P. Uyttenhove
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2011 - 2012

UNIVERSITEIT
GENT

ISBN 978-90-8578-509-5
NUR 956, 965
Wettelijk depot: D/2012/10.500/35

Dankwoord

De dingen krijgen enkel en alleen betekenis door hun context. Dit heb ik gedurende mijn periode als doctoraatsstudent continu onthouden *en* ondervonden. De onderzoeksresultaten die ik in dit proefschrift voorleg, komen er door jullie. Ik ben bewust iedereen en alles in die context bijzonder dankbaar. Vier jaar aan 'alles en iedereen' opsommen gaat niet, maar ik wil in dit dankwoord wel even de schijnwerpers richten op de mensen die het meest bijgedragen hebben aan dit doctoraat.

Prof. Ronny De Meyer, jij was degene die me jaren geleden op het spoor van een doctoraatsonderzoek zette. Jouw ambitie om een digitale architecturale ontwerpomgeving te realiseren op maat van de architect was de initiële drijfveer van dit doctoraatsonderzoek. Je hebt heel wat getrotseerd om dit toen mogelijk te maken, waarvoor ik je uitdrukkelijk wil bedanken. Inhoudelijk heb je bijzonder veel bijdrage geleverd door je grote betrokkenheid in het onderwerp. Ondertussen, vier jaar later, zijn de oorspronkelijke ambities dan ook behoorlijk geëvolueerd, wat getuigt van een gezonde overleg- en onderzoeksmethodiek. Het continue motiveren om verder te zoeken naar mogelijkheden en het continue aangeven van onderzoekspistes heeft me heel wat bijgebracht, waarvoor ik heel dankbaar ben.

De rol van perfecte copromotor werd vier jaar lang opgenomen door Prof. Jan Van Campenhout. Jij kwam net genoeg aan mijn deur aankloppen om een belangrijke impact te hebben op mijn doctoraatsonderzoek. En, hoewel ik van jou onderdak kreeg in het Technicum, stond je gelukkig ook niet elke dag aan mijn deur. Hartelijk dank voor het belangrijke bijstuurwerk en voor de onmisbare tweede mening, niet enkel over het onderzoeksonderwerp, maar ook over de werkmethode als doctoraatsonderzoeker in het algemeen.

Verder bedank ik heel graag mijn naaste collega's Ruben, Tine en Tiemen voor de vruchtbare samenwerkingen in allerhande projecten. De broodjes van resto de Brug zijn voor mij ondertussen een referentie geworden. Ook de nodige pauzes in de Vooruit deden bijzonder deugd als onderbrekingsmomenten van het onderzoek. Daarnaast bedank ik graag de verschillende andere onderzoekspartners met wie ik in de loop van mijn

doctoraatsonderzoek overlegd of samengewerkt heb. Meer specifiek wil ik hier in de eerste plaats Davy Van Deursen en Jos De Roo bedanken voor de verschillende bijzonder interessante samenwerkingen en voor de nuttige vergaderingen in de Zuiderpoort. Zonder jullie bijdrage vanuit de vakgroep Elektronica en Informatiesystemen had dit proefschrift er helemaal anders uit gezien. Ook Kathryn Piquette bedank ik graag om me van bij het begin van mijn doctoraat vanop een afstand op te volgen. Deze overlegmomenten deden me bijzonder deugd. Vitor Santos Costa bedank ik graag voor de samenwerking in Porto. Atocha Aliseda bedank ik graag om me als bezoekend onderzoeker op te vangen in Mexico-stad en voor de verschillende overlegmomenten nadien. Ook Laura bedank ik graag expliciet om me te laten proeven van het Mexicaanse leven. Het was een hele eer om bij jullie in de leer te kunnen gaan. Tot slot was ook de nauwe samenwerking met Danilo Di Mascio in het Boekentorenproject een bijzonder fijne ervaring, waarvoor ik heel dankbaar ben.

Ik bedank ook graag al mijn vrienden, familie en kennissen die steeds begaan waren met het wat, hoe en waarom van mijn onderzoek, zonder hier al te ver in te willen gaan. Het was jullie misschien niet altijd even duidelijk, maar jullie interesse werd bijzonder gewaardeerd. Een bijzonder medaille gaat hierbij naar mijn ouders, die van in het begin op de eerste rij zaten om mijn onderzoek mee op te volgen. Jullie onvoorwaardelijke steun lijkt misschien vanzelfsprekend, maar dat is het niet. Bijzonder bedankt!

Tot slot bedank ik graag de twee vrouwen van mijn leven, Kristy en Marie! Marie, bedankt om me niet te veel korte nachten te bezorgen en om me regelmatig met een lach van achter de boeken te halen. Kristy, zonder jou zou ik hier zelfs niet aan begonnen zijn. Jouw alledaagse steun was absoluut noodzakelijk, van in de kleinste details tot in de belangrijkste keuzes van het leven. Je hebt deze taak met glans volbracht, zoals verwacht eigenlijk, en daar ben ik heel erg fier op. We hebben reeds een perfecte weg afgelegd, als één team, en ik hoop dit we dit nog lang mogen blijven doen! Dankjewel.

Gent, juni 2012
Pieter Pauwels

Table of Contents

List of Figures

List of Acronyms

A

AEC	Architecture, Engineering and Construction
AI	Artificial Intelligence
AIM	Architectural Information Modelling
AM	Architectural Memory
API	Application Programming Interface

B

BIM	Building Information Modelling

C

CA(A)D	Computer-Aided (Architectural) Design
CEN	European Committee for Standardization
CFD	Computational Fluid Dynamics
CWM	Closed World Machine

D

DCC	Dynamic Content Creation
DL	Description Logic

E

EYE	Euler Yap Engine

F

FBM	Feature-Based Modelling
FM	Facility Management

G

GUI	Graphical User Interface

H

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol

I

ICT	Information and Communication Technology
IFC	Industry Foundation Classes

L

LOD	Linked Open Data

M

MACE	Metadata for Architectural Contents in Europe
MR	Mixed Reality

N

NBN	National Bureau for Normalization

O

OLE	Object Linking and Embedding
OWL	Web Ontology Language

P

PLM	Product Lifecycle Management

R

RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIF	Rule Interchange Format

S

SDK	Software Development Kit
SPARQL	Simple Protocol and RDF Query Language
SQL	Standard Query Language
SWRL	Semantic Web Rule Language

U

UG-EPW	UGent - Energie Prestatie in Woningen (Energy Performance in Residential Buildings)
URI	Unique Resource Identifier

V

VR	Virtual Reality
VRVS	Virtual Reality - Visual Simulation
VSC	Virtual Simulation and Calculation

W

W3C	World Wide Web Consortium
WWW	World Wide Web

X

X3D	eXtensible 3D
XML	eXtensible Markup Language

Y

YAP	Yet Another Prolog

Nederlandse samenvatting

In de afgelopen jaren werden diverse informatiesystemen uitgewerkt ter ondersteuning van aspecten in het architectuurontwerp. Deze systemen trachten de ontwerper op verschillende manieren te helpen: ze bieden een digitale schetsomgeving aan, ze laten foto-realistische visualisaties toe, ze bieden ondersteuning bij het maken van berekeningen en simulaties voor structurele, thermische, akoestische en allerlei andere aspecten van het ontwerp, enzovoort. Ondanks deze diversiteit bieden informatiesystemen in het domein van de architectuur nog steeds onvoldoende ondersteuning aan ontwerpers bij hun creative denkproces. Verschillende aspecten van het ontwerpen, in het bijzonder die aspecten waarbij de kennis, expertise en opgedane ervaringen van de ontwerper een rol spelen, worden amper ondersteund door informatiesystemen. Dit proefschrift behandelt de vraag die in deze stelling vervat zit: waarom bieden informatiesystemen geen afdoende ondersteuning in het architectuurontwerp?

Deze vraag wordt initieel behandeld met een onderzoek van centrale problemen in het gebruik van bestaande informatiesystemen. Vier types informatiesystemen worden daarbij beschouwd: modelleertoepassingen, archieftoepassingen, rekentoepassingen en visualisatietoepassingen. Toepassingen in deze categorieën zijn zeer divers in structuur en uitwerking, maar ze tonen regelmatig zeer gelijkaardige tekortkomingen. De informatie die kan beschreven worden in modelleertoepassingen is 'onvoldoende en te eenvoudig' of 'te veel en te complex'; de functionaliteit waarin voorzien wordt door rekentoepassingen is 'niet juist' of 'niet relevant'; enzovoort. Bijkomend wordt informatie van toepassingen amper effectief hergebruikt door andere toepassingen.

Verschillende elementen blijken dus mee te spelen in de bovenstaande onderzoeksvraag. Elk element is daarbij op een bepaalde manier gerelateerd aan tekortkomingen in de uitwisseling van informatie (1) tussen informatiesystemen onderling en (2) tussen informatiesystemen en de ontwerper. De eerste tekortkoming kan geïnterpreteerd worden als een probleem van interoperabiliteit tussen informatiesystemen. Dit probleem wordt ook herkend in verschillende andere domeinen naast architectuurontwerp. De tweede tekortkoming heeft te maken met een functionaliteitsprobleem: de functionaliteit die aangeboden wordt door informatiesystemen beantwoordt niet aan de functionaliteit die verwacht wordt door

de eindgebruiker. Ook deze tekortkoming kan herkend worden in andere toepassingsdomeinen. Deze twee centrale problemen vormen de twee onderzoekslijnen van dit proefschrift. Door deze onderzoekslijnen te volgen, wordt onderzocht in hoeverre en op welke manieren het architectuurontwerp kan ondersteund worden door informatiesystemen.

De eerste onderzoekslijn behandelt het bovenstaande interoperabiliteitsprobleem. Verschillende aanpakken voor dit probleem worden bekeken en gedocumenteerd: ongecontroleerd uitwisselen van informatie, hermodelleren van informatie, interoperabiliteit op kernel-niveau, het gebruiken van een centrale informatiestructuur, de strategie met softwarepakketten, en de aanpak met technologieën uit het semantisch web. De laatste aanpak blijkt het meest veelbelovend te zijn, doordat informatie uit heel diverse toepassingsdomeinen met deze aanpak gemakkelijk expliciet en ondubbelzinnig aan elkaar gekoppeld kan worden. Door deze koppeling zijn informatiesystemen beter in staat om elkaars informatie te gebruiken. Technologieën uit het semantisch web maken bovendien gebruik van één generiek formalisme, namelijk het 'Resource Description Framework' (RDF), en ze blijken wereldwijd gebruikt te worden in heel uiteenlopende domeinen. Bijgevolg wordt de kans merkelijk groter dat informatie beschikbaar is en direct of indirect gekoppeld is aan informatie uit andere kennisdomeinen die mogelijk van nut is voor de ontwerper.

Uit het experimenteel onderzoek in deze thesis blijken twee gepaste methoden om deze koppelingen te realiseren met deze aanpak. Enerzijds kan informatie gekoppeld worden door expliciet de verbanden te beschrijven die tussen concepten in toepassingsdomeinen bestaan. Anderzijds kan een methode gebruikt worden die op regels is gebaseerd en bijgevolg een actieve conversie mogelijk maakt tussen informatiestructuren.

Naast het interoperabiliteitsprobleem zou de aanpak met technologieën uit het semantisch web ook kunnen gebruikt worden voor het bovenstaande functionaliteitsprobleem. Dit wordt getest voor de vier soorten informatiesystemen die eerder aangehaald werden. Uit deze testen wordt geconcludeerd dat verbeteringen mogelijk zijn in de zin dat het onderliggende web van gekoppelde informatie in RDF eveneens een eenduidig model kan bevatten dat tot op zekere hoogte aspecten van de ontwerpkennis van een ontwerper voorstelt. Door dit model te gebruiken zouden informatiesystemen beter kunnen aangepast worden aan de ontwerper en kan de functionaliteit aangeboden worden die gevraagd of benodigd is door de ontwerper. Dit leidt logischerwijs tot de volgende twee samenhangende onderzoeksvragen.

1. In hoeverre kan ontwerpkennis van een ontwerper in een eenduidig model of web van informatie voorgesteld worden?

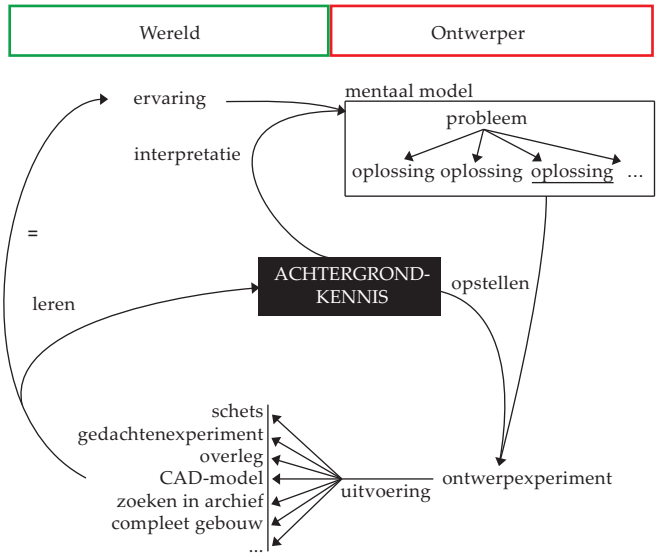
2. Welke rol speelt deze voorstelling in het ontwerpproces van een ontwerper?

De tweede onderzoekslijn van dit proefschrift, die zich richt op het functionaliteitsprobleem, behandelt beide onderzoeksvragen. Dit wordt gedaan aan de hand van bestaande theorieën over hoe ontwerpers denken. Een overzicht wordt gegeven van de voornaamste theorieën over hoe ontwerpers omgaan met ontwerpsituaties, al dan niet met behulp van informatiesystemen, en hoe ze ontwerpkennis opbouwen en gebruiken. Uit dit onderzoek kunnen vier elementen onderscheiden worden die van cruciaal belang zijn in het ontwerpproces:

- het belang van *analoog redeneren* voor het produceren van creative ideëen,
- de *co-evolutie* van (sub)problemen en (sub)oplossingen,
- de wereld waarmee de ontwerper interageert (*ervaringen*), en
- *achtergrondkennis* of vuistregels opgebouwd door ervaringen.

Achtergrondkennis of vuistregels staan centraal in de manier waarop ontwerpers denken. Deze ontwerpkennis blijkt te bestaan uit een soort ontwerppatronen die opgedaan worden door ervaring. Deze ontwerpkennis helpt ontwerpers nieuwe ervaringen te herkennen en te categoriseren door middel van analoge redeneerprocessen, ongeacht of het gaat om de 'interface' van een informatiesysteem, de respons of reactie gegeven door een andere ontwerper, een schets, een gedachtenexperiment, of een CAD-model. Daarnaast gebruiken ontwerpers hun ontwerpkennis om concrete ontwerpexperimenten te bedenken en uit te voeren. De resultaten van deze ontwerpexperimenten kunnen dan gebruikt worden om het oorspronkelijk mentaal model van een ontwerpsituatie te bevestigen of te weerleggen. Tot slot maakt deze ontwerpkennis het ook mogelijk om te leren uit ervaringen die voortkomen uit de gemaakte ontwerpexperimenten. Ontwerpsituaties worden dus voortdurend gekaderd en herkaderd (co-evolutie) door eindeloos te itereren door een proces van (1) analoog redeneren, (2) opstellen van ontwerpexperimenten, en (3) uitvoeren van ontwerpexperimenten en leren uit ervaringen (Fig. 1).

Uit dit overzicht van theorieën over hoe ontwerpers denken wordt geconcludeerd dat informatiesystemen kunnen beschouwd worden als extra omgevingen waarin ontwerpers ontwerpexperimenten kunnen uitvoeren (linksbeneden in Fig. 1). Het eerder voorgestelde gebruik van technologieën uit het semantisch web verandert dit niet. De representaties die in dergelijke omgevingen gemaakt worden, wijken echter wezenlijk af van de ontwerpkennis van een ontwerper. Dit geldt net zo goed voor andere externe representaties, zoals schetsen, fysieke modellen, conversaties, enzovoort. Een externe representatie is een statische momentopname van aspecten van een bepaald mentaal model, terwijl de ontwerpkennis waarmee geredeneerd wordt door een ontwerper, bestaat uit een dynamische



Figuur 1: Ontwerpsituaties worden voortdurend gekaderd en herkaderd (co-evolutie) door eindeloos te itereren door een proces van (1) analoog redeneren, (2) opstellen van ontwerpexperimenten, en (3) uitvoeren van ontwerpexperimenten en leren uit ervaringen.

verzameling van ervaringen en de bijbehorende mentale modellen die veel rijker is. Deze statische momentopname kan dus, zoals geanticipeerd, niet de rijkdom aan concepten bevatten die inherent aanwezig is in menselijke (ontwerp)kennis, en dient eerder beschouwd te worden als een *resultaat* van het denkproces van een ontwerper dat tot op bepaalde hoogte enkele van deze concepten expliciet voorstelt. Met andere woorden, ontwerpers steunen in de eerste plaats op hun eigen ontwerpkennis en maken gebruik van externe representaties als externe hulpmiddelen in ontwerpexperimenten. Dit geeft de begrensde invloed weer die huidige informatiesystemen hebben over het totale ontwerpproces. Ook de voorgestelde aanpak aan de hand van technologieën uit het semantisch web resulteert in een dergelijk ondersteuning.

Niettemin laten technologieën uit het semantisch web merkelijke verbeteringen toe in het informatiebeheer en in de uitwisseling van informatie tussen informatiesystemen die gebruikt worden als omgevingen voor ontwerpexperimenten. Koppelingen kunnen gemaakt worden tussen verschillende voorstellingen van informatie uit verschillende informatiesystemen. Dit kan statisch en direct gebeuren door het gebruiken van expliciete koppelingen tussen concepten. Anderzijds kan dit ook zeer dynamisch en indirect gebeuren, door gebruik te maken van complexe verzamelin-

gen regels. Naast deze verbeterde informatie-uitwisseling krijgen ontwerpers ook verbeterde faciliteiten om aspecten van een ontwerpsituatie uit te drukken volgens hun eigen termen en concepten. Dit kan verbeteringen brengen aan standaardtoepassingen voor de ontwerper (modelleertoepassingen, archieftoepassingen, rekentoepassingen en visualisatietoepassingen), in de zin dat deze toepassingen hun functionaliteit kunnen aanpassen aan specifieke situaties op basis van informatie van de ontwerper.

Ook veel complexere functionaliteit kan aangeboden worden vanuit een omgeving voor ontwerpexperimenten. Complexe heuristieken kunnen gebruikt worden op basis van de informatie die aangeboden wordt door de ontwerper. Deze heuristieken nemen typisch meer parameters in rekening dan menselijke ontwerpers in een korter tijdsbestek. Op dergelijke manier kunnen extra omgevingen aangeboden worden aan de ontwerper waarin op korte tijd uitgebreide verkenningen van de ontwerpruimte gemaakt kunnen worden. Wanneer dit gekoppeld wordt aan generatieve technieken zouden dergelijke systemen specifieke suggesties kunnen aanbieden op basis van de resultaten van de verkenningen van de ontwerpruimte. Binnen deze context bieden technologieën uit het semantisch web in de eerste plaats de mogelijkheid aan ontwerpers om gemakkelijker die aspecten van hun informatie te beschrijven die ze in rekening willen brengen. Bijgevolg kunnen de verkenningen van de ontwerpruimte en de generatieve ontwerpsystemen gemakkelijker aangepast worden aan de noden van de ontwerper. Op die manier kunnen deze technologieën de ontwerpers een stap dichter brengen bij gespecialiseerde agenten die slechts een deel van de ontwerpsituatie in rekening brengen, maar die wel gemakkelijker aan de noden van de ontwerper in een specifieke ontwerpsituatie kunnen voldoen.

Dit leidt tot de volgende conclusie van dit proefschrift. De voorgestelde aanpak met technologieën uit het semantisch web kan tot op zekere hoogte verbeteringen bieden voor de twee centrale problemen die zich momenteel voordoen in het gebruik van informatiesystemen in het architectuurontwerp. Informatiesystemen zijn voornamelijk te beschouwen als instrumenten voor het maken van ontwerpexperimenten. Hierbij wordt informatie ter interpretatie gepresenteerd aan de ontwerper, en de manier waarop de ontwerper dit gebruikt, kan op geen manier beheerd worden. Deze aanpak zal niet in staat zijn om het gegeven interoperabiliteitsprobleem en het functionaliteitsprobleem volledig 'op te lossen'. Desondanks kan de voorgestelde aanpak belangrijke verbeteringen brengen in de ondersteuning van ontwerpers door informatiesystemen, onder de vorm van verbeterde mogelijkheden tot informatiebeheer.

English summary

Over the past years, numerous information systems have been realized which support various aspects in architectural design. These systems help the designer in various ways: they provide drafting support, they allow a variety of photorealistic visualizations, they provide very diverse calculation and simulation aids for structural, thermal, acoustic and other aspects of the design. Yet, despite the richness of the design tools currently available, various deficiencies remain in the way in which they can be used to effectively support design activities. Several aspects of design, in particular there where the designer's knowledge, expertise and past experiences are involved, remain essentially without proper information system support. This thesis addresses the question embodied in this statement: why is it that computer-based design aids still lack in providing full design support?

In addressing this question, this thesis starts by looking into some of the central issues in existing information system support for architectural design thinking. Four types of information system support are outlined: modeling applications, archive applications, calculation applications and visualization applications. Applications in these categories are very diverse in their design and implementation, yet they often show very similar shortcomings when they are evaluated in a real-world context. The information that can be described within modeling applications is either 'not enough and too simple', or 'too much and too complex'; the functionality provided by simulation applications is 'not correct' or 'irrelevant'; and so forth. Additionally, none of the applications effectively reuses information from any of the other applications.

So it turns out that various effects are in play, all in some way related to deficiencies in the information exchange (1) among information systems, as well as (2) between information systems and the designer. The first deficiency relates to the issue of interoperability between information systems that has been distinguished in various domains, not only architectural design. The second deficiency relates to the mismatch that is typically encountered between the functionality provided by information systems and the functionality supposedly desired by the end user. Also this deficiency is not only encountered in architectural design, but can be distinguished in other domains. These two main lines of investigation are fol-

lowed throughout this thesis. As such, it is investigated in what ways and to what extent design activity –commonly called design thinking– could be supported by information systems.

The first research line in this thesis focuses on the outlined interoperability issue in information system support. Diverse approaches are documented to address this issue: sharing information in the wild, the remodeling effort, kernel-level interoperability, the centralized information structure, the software suite strategy, and the linked data approach. When relying on semantic web technologies, the last approach appears to be the most promising, because these technologies allow to explicitly and unambiguously connect information deployed in diverse application domains and applications. Additionally, semantic web technologies use a generic description language, namely the Resource Description Framework (RDF), and they appear to be deployed on a global scale. Information thus has a notably higher chance of being available and related to information in other knowledge domains of possible use to the architectural designer.

The experimental investigation of a linked data approach with semantic web technologies suggests two methods for combining information in different application domains, namely a method relying on explicit links between concepts residing in different application domains, and a rule-based method which enables an active conversion between diverse information structures that represent nearly the same information.

Apart from the interoperability issue, the linked data approach might also address the functionality mismatch issue. This is tested for the four application types outlined earlier. From this tests, it is concluded that improvements are possible in the sense that the web of information used by applications can include an unambiguous RDF graph that to some extent represents aspects of design information of a designer. By relying on this information, information systems might thus be customized so that they provide the functionality requested and needed by the designer. This leads naturally to the two following connected research questions.

1. To what extent can aspects of design information be represented in an unambiguous model or web of information?

2. What is the role and effect of this representation in the design process of a designer?

The second line of research, which focuses on the functionality mismatch issue, addresses these two research questions. It does so by going through existing theories of design thinking. An outline is given of the most significant theories of how designers deal with the design situations they encounter, with or without information system support, and how they construct and use information. This research points towards four key elements in the design thinking process:

- the importance of *analogical reasoning* in producing creative ideas,
- the concept of *co-evolution* of (sub)problems and (sub)solutions,
- the world with which the designer interacts (*experiences*), and
- *guiding principles* or background knowledge built up by experiences.

Central in the way in which designers think is the 'guiding principles' element, which is the knowledge by experience of the designer. This design knowledge is formed by a set of personally collected and thus familiar design patterns. These guiding principles help architectural designers to recognize and categorize a new experience through an analogical reasoning process, whether this be the interface of an information system, the feedback given by a colleague designer, a sketch, a thought experiment, or a CAD model. Furthermore, guiding principles help designers to devise design tryouts that can be used to confirm or refute their mental model of a design situation. And finally, these guiding principles also allow learning from experiences resulting from design tryouts. Design situations are continuously framed and reframed (co-evolution) by endlessly iterating through a process of (1) analogical reasoning, (2) devising design tryouts, and (3) performing design tryouts and learning from experiences (Fig. 2).

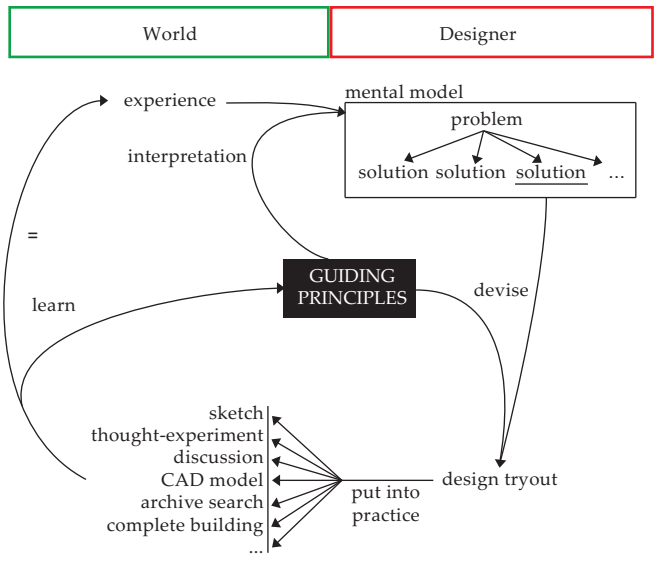


Figure 2: Design situations are continuously framed or reframed (co-evolution) by endlessly iterating through a process of (1) analogical reasoning, (2) devising design tryouts, and (3) performing design tryouts and learning from experiences.

From this overview of theories in design thinking, it is finally concluded that information systems essentially provide additional environments to architectural designers for externalizing their design knowledge and, as such, make design tryouts. However, external representations of design knowledge, such as the kind of representations that would result from using semantic web technologies, differ substantially from the design knowledge in the human mind. The former typically represents a static snap-shot of a particular mental model, whereas the latter consists of a dynamic set of experiences and corresponding mental models. In other words, architectural designers primarily rely on their own design knowledge and primarily use external representations, such as a graph in the semantic web, a sketch or a physical model, as external mediators for their design tryouts. This indicates the bounded impact that current information systems have on the complete design process: information systems that provide support to designers should primarily be considered as additional parts of the world, with which architectural designers can engage for making design tryouts. Similar to a paper and pencil environment, a CAD system or a simulation environment allows a designer to make design tryouts. This is how information systems are generally used nowadays, knowingly or not. Also the suggested linked data approach results in such an information system support.

Nevertheless, information management and exchange among information systems for design tryouts can be notably improved with semantic web technologies. Links can be made between different representations of information residing in different information systems. This can be done both in a static and direct way, using explicit links, or in a dynamic and indirect way, using complex rule sets. Additionally, designers have improved facilities to model aspects of a design situation in their terms and concepts, and link these to concepts residing in information systems. This can bring improvements to standard applications (modeling, archive, calculation, and visualization applications) which are then able to customize their functionality based on the designer's information.

Also far more complex functionality can be provided from within a design tryout environment. Based on the information provided by the designer, complex heuristic methods can be used, taking into account more parameters than human designers can in a time period that is shorter. As such, they provide extra environments for making elaborate design space explorations in a very short time. When coupled to generative techniques, such systems might provide specific suggestions based on findings that result from the considered design space exploration. In this context, a linked data approach with semantic web technologies enables designers to construct their proper representations of the design situation, so that far more customized design space explorations or generative designs are feasible. As such, they could bring designers another step closer to expert agents that take only a part of the design situation into account, but that are at

least more easily tailored to the designer's needs in this specific design situation.

This leads to the final conclusion of this thesis. The suggested linked data approach based on semantic web technologies can to some extent improve the two main issues currently encountered in information system support for architectural design. In this case, information systems are to be considered as tools for design tryouts, in which information is presented for a designer to interpret, and the way in which the designer makes this interpretation cannot be 'managed' or 'determined' in any way. This approach will most probably not be able to completely address the interoperability issue and the functionality issue. Nevertheless, the linked data approach can bring significant improvements to both issues, because of the improved possibilities for information management.

1 Introduction

1.1 Information system support in architectural design

Over the past years, numerous software applications have been realized which support various aspects in architectural design. These systems help the designer in various ways: they provide drafting support, they allow a variety of photorealistic visualizations, they provide very diverse calculation and simulation aids for structural, thermal, acoustic and other aspects of the design. Yet, despite the richness of the design tools currently available, various deficiencies remain in the way in which they can be used to effectively support design activities. Several aspects of design, in particular there where the designer's knowledge, expertise and past experiences are involved, remain essentially without proper information system support.

This thesis addresses the question embodied in the above statement: why is it that computer-based design aids still lack in providing full design support? It turns out that various effects are in play, all in some way related to deficiencies in the information exchange (1) among software components, as well as (2) between software components and the designer. The first deficiency relates to the issue of *interoperability* between information systems that has been distinguished in various domains, not only architectural design. The second deficiency relates to the mismatch that is

typically encountered between *functionality* provided by information systems and the functionality supposedly wanted by the end user. Also this deficiency is not only encountered in architectural design. These two main lines of investigation are followed throughout this thesis. As such, it is investigated in what ways and to what extent design activity –commonly called design thinking– could be supported by information systems.

1.2 Outline of the thesis

We will start in **Chapter 2** with a brief overview of the most central issues in existing information system support for architectural design thinking. In this chapter, a brief and general overview is given of existing information systems and their applications in the domain of architectural design thinking. This context serves to situate and document (1) the lack of interoperability among existing information systems, and (2) the mismatch that is typically encountered between functionality provided by information systems and functionality supposedly wanted by architectural designers, which are the two lines of investigations outlined earlier.

Chapter 3 proceeds with an approach suggested in this thesis to tackle the issues outlined in chapter 2. This approach relies on a linked data approach with the newly emerging semantic web technologies. Because semantic web technologies allow to explicitly and unambiguously connect information deployed in diverse information systems, they might bring about an improved level of interoperability among these information systems. The resulting approach might also bring about improvements to the functionality mismatch issue between architectural designers and information systems. By relying on semantic web technologies, namely, information systems might be customized so that they provide the functionality requested and needed by the designer.

This second line of research is documented in **Chapter 4**, which gives an overview of existing theories of design thinking. This chapter gives an outline of the most significant theories of how designers deal with the design situations they encounter, with or without information system support. This research points towards four key elements in the design thinking process, namely, (1) analogical reasoning, (2) co-evolution, (3) experiences in the design context, and (4) guiding principles. The final summarizing overview indicates what is causing the outlined functionality mismatch of current information systems and architectural designers. The information structures that lie at the foundations of the diverse information systems used in the design process, namely, are only to a limited extent able to capture design information. These representations of design information

should rather be considered as parts of bounded design tryouts that represent only very specific parts of the whole design situation.

This leads to the conclusion in **Chapter 4** and **Chapter 5** that a linked data approach with semantic web technologies will only to a limited extent be able to address the interoperability issue and the functionality mismatch issue. However, considering the significant possibilities for information management that are provided by semantic web technologies, this approach is one of the better options to consider. With their ability to explicitly and unambiguously connect diverse information structures, they provide not only an improved answer to the interoperability issue, but also to the functionality mismatch issue. As such, these technologies bring powerful facilities for customization to standard information systems (modeling, archive, calculation, and simulation applications). Also more complex and specialized information systems, for instance for design space exploration, generative design, and agent-based advise, can more easily be customized.

1.3 Publications

This thesis presents in a coordinated text the research that was previously published piecemeal in several separate publications, to which the interested reader is referred for more detail. The following publications are recommended in particular, because of their close relation with specific chapters in this thesis.

- chapter 2
  - P. Pauwels, P. Present, and T. Strobbe. *A pragmatic approach towards software usage in construction projects : the Harbour House in Antwerp, Belgium.* In Proceedings of the 9th European Conference on Product and Process Modelling (in press), 2012.
  - P. Pauwels, R. De Meyer, and J. Van Campenhout. *Interoperability for the design and construction industry through semantic web technology.* In Proceedings of the 5th International Conference on Semantic and Digital Media Technologies. In Lecture Notes in Computer Science 6725, pages 143-158, 2010.
- chapter 3
  - P. Pauwels, R. De Meyer, and J. Van Campenhout. *Linking a game engine environment to architectural information on the semantic web.* Journal of Civil Engineering and Architecture 5(9):787-799, 2011.

  - P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout. *A semantic rule checking environment for building performance checking.* Automation in construction 20(5):506-518, 2011.
  - P. Pauwels, D. Van Deursen, J. De Roo, T. Van Ackere, R. De Meyer, R. Van de Walle, and J. Van Campenhout. *Threedimensional information exchange over the semantic web for the domain of architecture, engineering and construction.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing 25(4):317-332, 2011.
  - P. Pauwels, R. De Meyer, and J. Van Campenhout. *Extending the design process into the knowledge of the world.* In Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAADFutures), pages 203-216, 2011.

- chapter 4

  - P. Pauwels, R. De Meyer, and J. Van Campenhout. *A critical evaluation of information system support for design thinking.* Design Issues (in press), 2012.

1.3.1 Publications in international journals

P. Pauwels, R. De Meyer, and J. Van Campenhout. *A critical evaluation of information system support for design thinking.* Design Issues (in press), 2012.

P. Pauwels, R. De Meyer, and J. Van Campenhout. *Linking a game engine environment to architectural information on the semantic web.* Journal of Civil Engineering and Architecture 5(9):787-799, 2011.

P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout. *A semantic rule checking environment for building performance checking.* Automation in construction 20(5):506-518, 2011.

P. Pauwels, D. Van Deursen, J. De Roo, T. Van Ackere, R. De Meyer, R. Van de Walle, and J. Van Campenhout. *Threedimensional information exchange over the semantic web for the domain of architecture, engineering and construction.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing 25(4):317-332, 2011.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. *Semantics-based design: can ontologies help in a preliminary design phase?* Design Principles and Practices: An International Journal 3(5):263-276, 2009.

1.3.2 Publications in international conferences

P. Pauwels, R. De Meyer, and J. Van Campenhout. *Information system support in construction industry with semantic web technologies and/or autonomous reasoning agents.* In Proceedings of the 9th European Conference on Product and Process Modelling (in press), 2012.

P. Pauwels, P. Present, and T. Strobbe. *A pragmatic approach towards software usage in construction projects : the Harbour House in Antwerp, Belgium.* In Proceedings of the 9th European Conference on Product and Process Modelling (in press), 2012.

T. Strobbe, P. Pauwels, R. De Meyer, R. Verstraeten, and J. Van Campenhout. *Optimization in compliance checking using heuristics: Flemish Energy Performance Regulations (EPR).* In Proceedings of the 9th European Conference on Product and Process Modelling (in press), 2012.

P. Pauwels and R. Bod. *'Applications for experimenting' or 'reasoning agents' as design decision support tools?* In Proceedings of the 5th International Conference on Design Computing and Cognition (in press), 2012.

P. Pauwels. *A brief look into the head of an architectural designer.* In Proceedings of the 3rd International Symposium for Humanities and Technology (Interface), pages 57-58, 2011.

P. Pauwels, R. De Meyer, and J. Van Campenhout. *Extending the design process into the knowledge of the world.* In Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAADFutures), pages 203-216, 2011.

T. Strobbe, P. Pauwels, R. Verstraeten, and R. De Meyer. *Metaheuristics in architecture: using genetic algorithms for constraint solving and evaluation.* In Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAADFutures), pages 866-867, 2011.

P. Pauwels, R. De Meyer, K. Samyn, and M. Audenaert. *The role of game rules in architectural design environments.* In Proceedings of the 3rd International Conference on Games and Virtual Worlds for Serious Applications, pages 184-185, 2011.

P. Pauwels, T. Jonckheere, R. De Meyer, and J. Van Campenhout. *Increasing information feed in the process of structural steel design.* In Sustainable Construction And Design, pages 180-189, 2011.

T. Strobbe, P. Pauwels, R. Verstraeten, and R. De Meyer. *Metaheuristics in architecture.* In Sustainable Construction And Design, pages 190-196, 2011.

T. Jonckheere, R. Verstraeten, P. Pauwels, and R. De Meyer. *Ontwikkeling van een Google SketchUp-plugin als ontwerpinstrument voor een energiezuinige architectuur.* In International Building Performance Simulation Association Nederland/Vlaanderen 2010 Event, 2010.

P. Pauwels, R. Verstraeten, T. Jonckheere, R. De Meyer, and J. Van Campenhout. *3D architectural design in the semantic web.* In Proceedings of the 7th Extended Semantic Web Conference, 2010.

P. Pauwels, R. De Meyer, and J. Van Campenhout. *Interoperability for the design and construction industry through semantic web technology.* In Proceedings of the 5th International Conference on Semantic and Digital Media Technologies. In Lecture Notes in Computer Science 6725, pages 143-158, 2010.

P. Pauwels, R. De Meyer, and J. Van Campenhout. *Visualisation of semantic architectural information within a game engine environment.* In Proceedings of the 10th International Conference on Construction Applications of Virtual Reality, pages 219-228, 2010.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. *Architectural information modelling to address limitations of BIM in the design practice.* In Proceedings of the 5th Conference on Information and Knowledge Management in Building, pages 15-17, 2009.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. *Architectural information modelling in construction history.* In Proceedings of the Third International Congress on Construction History, pages 1139-1146, 2009.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. *Architectural information modelling for virtual heritage application.* In Proceedings of the 14th International Conference on Virtual Systems and Multimedia - Digital Heritage, pages 18-23, 2008.

R. Verstraeten, P. Pauwels, W. Meeus, R. De Meyer, and J. Van Campenhout. *Industry Foundation Classes: a space-based model scheme?* In Proceedings of the 26th eCAADe Conference on Education and Research in Computer Aided Architectural Design in Europe, pages 117-124, 2008.

R. Verstraeten, P. Pauwels, R. De Meyer, W. Meeus, J. Van Campenhout, G. Lateur. *IFC-based calculation of the Flemish energy performance standard.* In Proceedings of the 7th European Conference on Product and Process Modelling, pages 437-443, 2008.

1.3.3 Publications in national conferences

P. Pauwels. *Architectural information modelling.* In 9e UGent-FirW doctoraatssymposium, 2008.

P. Pauwels. *Architectural information modelling: a semantic description framework for historical and theoretical knowledge in architecture.* In Proceedings of the 15th Joint Doctoral Seminar on Theory and History of Architecture, 2008.

2

Central issues in information system support for architectural design thinking

THIS thesis starts with an overview and discussion of central issues encountered in existing information system support for architectural design thinking. In this chapter, a brief and general overview is given of existing information systems and their applications in the domain of architectural design thinking. This context serves to situate and document two main issues, namely (1) the lack of *interoperability* among existing information systems, and (2) the mismatch between *functionality* provided by information systems and functionality supposedly required by architectural designers. Through the overview and discussion in this chapter, we are able to name the main reasons why information systems are apparently unable to provide effective support to the architectural designer.

2.1 Computer-aided design or computer-aided drafting?

Many information systems have been designed and implemented for the domain of architecture, engineering and construction (AEC). These systems can be categorized in modeling applications, archive applications,

calculation applications and visualization applications. Applications in these categories help the designer in various ways: they provide drafting support, they allow a variety of photorealistic visualizations, they provide very diverse calculation and simulation aids, and so forth. Nevertheless, designers and construction specialists often complain about the provided functionalities in real AEC projects: simulations are not correct or irrelevant, visualizations do not communicate the required information, archive applications contain the wrong or irrelevant information, and finally, none of the applications effectively reuses information from any of the other applications.

In this section, we will briefly consider applications of the given application categories, and point out some of the complaints that are typically made in AEC projects. Although implicit indications are often given to such complaints and shortcomings, few studies are available that only focus on outlining these shortcomings. We will therefore rely on conclusions that were made in Pauwels et al. [2012b] regarding the usage of information systems in a real project, namely, the construction project of the Port House in Antwerp, Belgium (Fig. 2.1). The executive architect in this project, Bureau Bouwtechniek, used diverse information systems for various purposes, among which modeling applications, calculation applications and visualization applications, thereby aiming at an integrated approach. Complaints, such as the ones given earlier, arose in the attempt of using particular information systems. This eventually resulted in a more pragmatic usage of these information systems, in which information systems have a more confined and specific purpose in the design process.

2.1.1 Modeling applications

The best known information systems for the design and construction industry are modeling applications: CAD applications, building information modeling (BIM) applications [Eastman et al., 2008], parametric design applications, and basic 2D and 3D modeling applications (Fig. 2.2). These applications allow designers to model a design into a certain information structure. This information structure is defined by the program code behind the modeling application in question. Basic 2D CAD applications enable the user to model a design in 2D geometric object models, using lines, points, surfaces, and so forth. Basic 3D applications allow this in 3D, using boxes, spheres, voids, and so forth. More advanced CAD systems typically focus on information management, and thus enable the user to model a design in more 'informative' elements, such as walls, windows, columns, beams, and so forth. These are typically called BIM applications

Figure 2.1: Impression of the future Port House (courtesy of Zaha Hadid Architects).

[Eastman et al., 2008], although references can also be made to feature-based modeling (FBM) applications [Shah and Rogers, 1988].

In all these cases, the way in which the information is stored, whether this be points, cubes, or walls, follows a quite rigid structure. The application, namely, provides the user with a specific object library and only enables the user to create instances of the objects in this library and alter their predefined parameters [Booch et al., 2007, Rumbaugh et al., 1991]. Parametric design applications follow a somewhat different approach in that they let designers define and modify parameters by themselves (see parametric tables [Eastman et al., 2009]). When considering such modeling applications in more detail, however, they also rely mainly on basic objects (boxes, spheres, and so forth) and operations (stretch, combine, numeric transformations, and so forth). They are thus essentially not that different from the other modeling applications: the design model in the designer's mind needs to be restructured so that it fits into the information structure provided by the modeling application.

In the case of the Port House project [Pauwels et al., 2012b], the design model was to be modeled in the BIM application Revit Architecture 2011. This application provides a complete library of parametric objects for modeling a building or design. This library includes doors, windows, floors, walls, and so forth. The engineering team in charge of modeling the Port House felt that many of the building elements of the Port House did not fit into this provided information structure: they can hardly be mod-

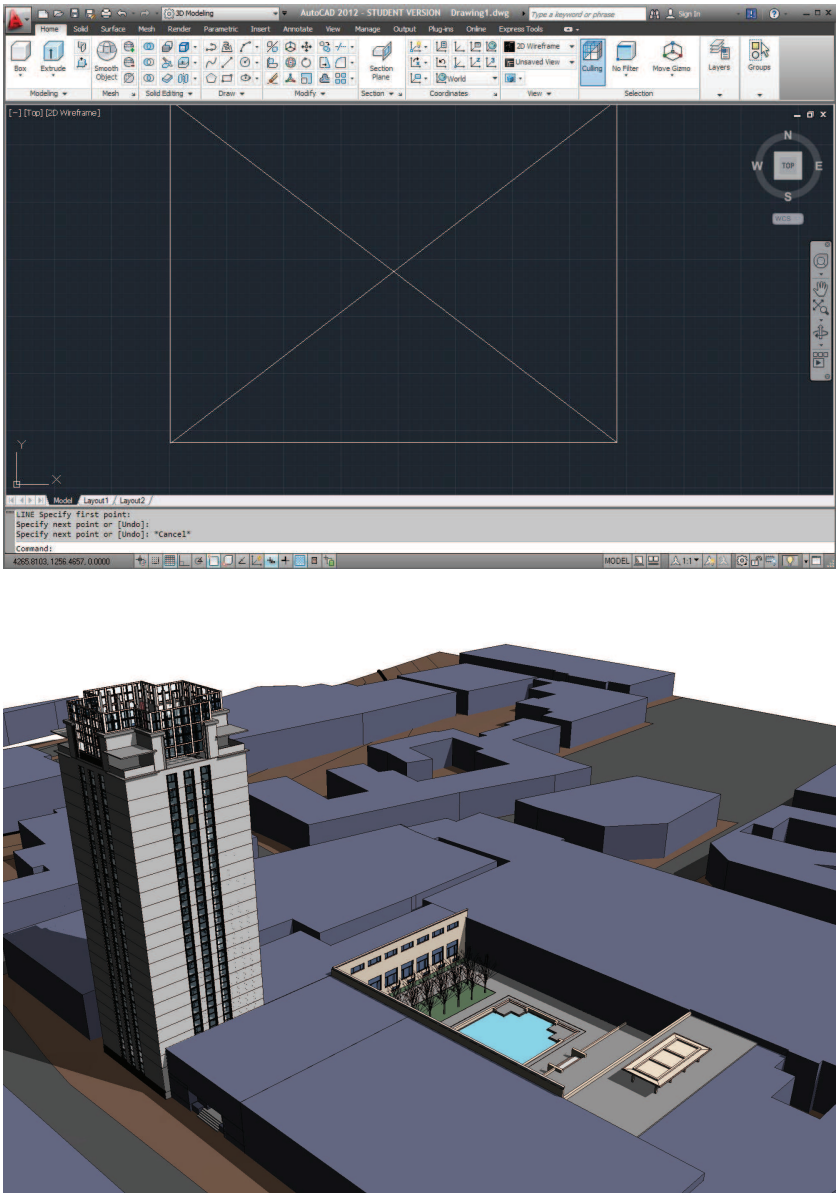


Figure 2.2: Modeling applications AutoCAD 2012 [Autodesk, 2012a] and Revit Architecture 2011 [Autodesk, 2012c] allow architectural designers to model a representation of their design. This can result, for instance, in simple line drawings (bottom) or in complex building information models (top).

eled with the standard door, window and wall elements. As a result, the glass façade, for instance, was modeled using 'generic models', which are simple geometric shapes that can be labeled with the appropriate terms ('window', for instance). The result is shown for a window module in Fig. 2.3. Note that also these geometric models do not precisely reflect the geometric shape of the actual window. Also at this level, the geometric shape needs to be restructured to fit into the geometric library of the application.

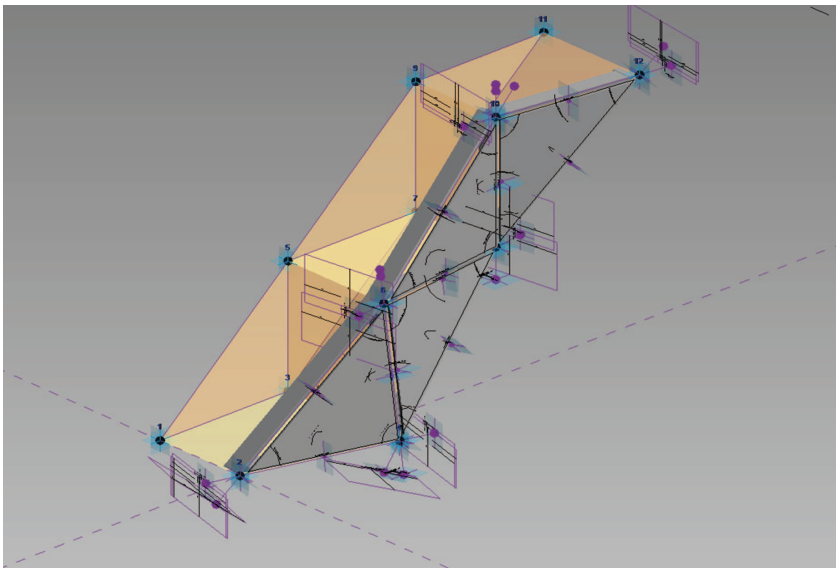


Figure 2.3: The parametric model of the glass module as it was modeled in Revit for the Port House in Antwerp (original image from Paulus Present - Bureau Bouwtechniek).

2.1.2 Archive applications

The main purpose of archive applications is to enable designers to more easily find information that could in some way be useful for their design situation. Information that is typically stored in books, in the minds of people, or in pictures and sketches, is brought together, digitized and made accessible via digital archive applications (Fig. 2.4). In almost all cases, the Internet serves as a vital gateway towards these archives. The archives themselves usually consist of a complex network of web pages, linked to databases of images, 3D models, texts, audio records, movies and/or other multimedia. To facilitate search in these databases, an additional metadata archive typically is set up, which stores the most important keywords or

'tags' linked to the content stored in these databases. By specifying certain keywords, related content can be retrieved and used by designers in their design process.

Crucial in such archive applications is the metadata structure that comes with the archive, since this is the gateway through which the user is able to find certain content. The tags in this metadata structure are either unorganized single keywords or keywords organized in specific structures (hierarchical, networked, and so forth). In the AEC domain, the latter organized structure is often preferred over the former unorganized structure. This implies that, similar to what happens in modeling applications, information needs to be restructured so that it fits into the information structure provided by the archive application.

The Port House project discussed in Pauwels et al. [2012b] was not added to any archive application, nor was a specialized archive application, such as OIKODOMOS or MACE (Fig. 2.4), used during the design and construction process by the engineering team. However, considering the design of the building (Fig. 2.1 and 2.20), one can see the difficulties of formalizing (characteristics of) the building using the information structures provided by these archive applications. For instance, MACE relies on a MACE Application Profile (AP) of the Learning Object Metadata (LOM) standard [Stefaner et al., 2007]. This AP represents the information structure used by MACE for the terms and values that could be used to describe architectural content [Arlati et al., 2008]. The list of classification terms is derived from different existing standards, among which the Getty Thesaurus [Getty Research Institute, 2012]. If the engineering team in charge for the Port House would want to add their project to the MACE archive, they would be required to describe the building in the terms provided by the MACE AP. Apart from stating that the building is an office building located in Antwerp and built around 2012, this can quickly become a task that is equally challenging as modeling the building in Revit Architecture, simply because terms in their understanding of the building do not entirely match the terms used by the archive application (the MACE AP).

2.1.3 Calculation applications

Whereas modeling and archive applications are typically aimed at bringing information together, calculation applications are typically aimed at (re-)using this information. Numerous calculation applications exist in the AEC domain: thermal analysis applications, computational fluid dynamics (CFD) simulation environments, compliance-checking applications, structural analysis applications, and so forth (Fig. 2.5). Calculation applications

Figure 2.4: Digital archive applications, such as the OIKODOMOS case repository (top) [ARC school of architecture La Salle, 2012] and MACE (bottom) [MACE Consortium, 2012], collect information that could be useful to designers and makes this information accessible via an online interface.

typically provide a complex set of rules and algorithms, which need to be closely related to incoming information, if they are ever to produce any results. In many cases, a structure is thus set up for the incoming information, so that (1) this information can easily be put into this structure, and (2) rules and algorithms can be designed and implemented according to this structure.

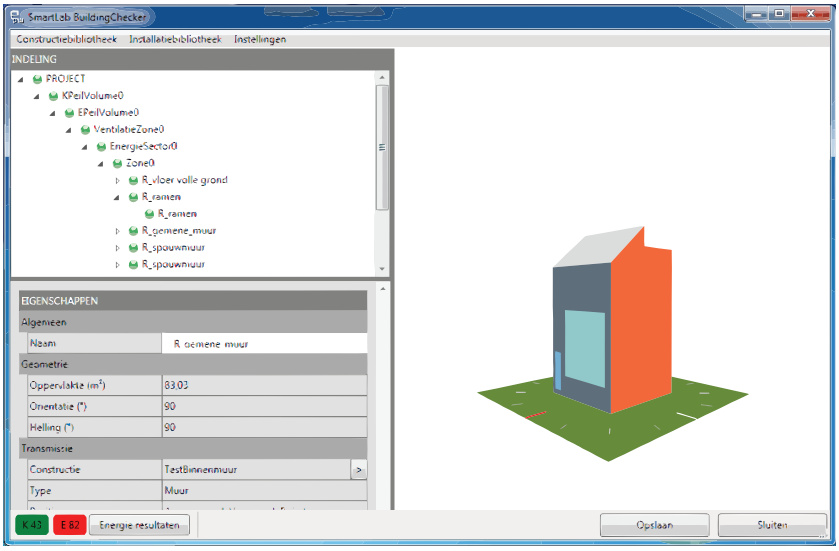


Figure 2.5: Calculation applications typically reuse available information in order to derive additional information. The UG-EPW application [Verstraeten et al., 2009], for instance, allows calculating an energy performance value for building models.

Unfortunately, the incoming information often does not follow the structure that is provided by the calculation application. If the incoming information is provided by a human user, then most often a similar situation occurs as in the modeling and archive applications: the information needs to be restructured so that it fits the information model provided by the calculation application. This often leads to choices for the human user (for instance, is something a 'column' or a 'short wall'?), leading to changing interpretations and 'miscalculations'. If the incoming information is provided by a computer application, for instance in the form of a digital 3D model, conversion routines are typically followed from the incoming information model to the information model of the calculation application. This similarly leads to changing interpretations and thus to misconversions and miscalculations.

An example of this situation in the context of the Port House can be found in the collaboration of the engineering team of Bureau Bouwtechniek with the structural engineers for the project. The engineering team used a specialized application for making structural calculations. The information structure in this applications is different from the information structure in the BIM application. Because information exchange between both information systems requires significant restructuring of information, it was decided to let both engineering teams maintain their own two separate models and to merge them only geometrically when necessary [Pauwels et al., 2012b].

2.1.4 Visualization applications

Visualization obviously plays an important role in AEC projects. However, the term 'visualization' also has many interpretations. In an AEC context, visualization most often stands for 'architectural visualizations'. An example of an architectural visualization is given in Fig. 2.1 and 2.6, showing images of buildings as they are designed and meant to be built at a certain moment in time. Architectural visualization applications are typically linked to modeling applications. In the case of 2D modeling applications, visualization software typically provides additional, more advanced modeling functionality, enabling the user to produce more compelling images of design scenes. When coupled to 3D modeling applications, visualization applications typically enable the production of animations and walkthrough movies of the building design, or still image renderings of design scenes. More advanced visualization applications are game engines and applications similarly capable of producing virtual environments that visualize the design (Fig. 2.6). Such virtual environments can be modeled and accessed in diverse ways, thereby making, for instance, the difference between Virtual Reality (VR) and Augmented Reality (AR) applications.

It can be argued that modeling, archive and calculation applications to some extent also provide 'an image of a building as it is designed and meant to be built'. This interpretation is less narrow than the traditional interpretation of the term 'architectural visualization'. In this interpretation, the Graphical User Interface (GUI) constitutes a significant component of the application. Through this GUI, the application visualizes a certain 'interpretation' of a design model, whether this be a 2D CAD drawing, a set of references in an archive application, or a 3D calculation model.

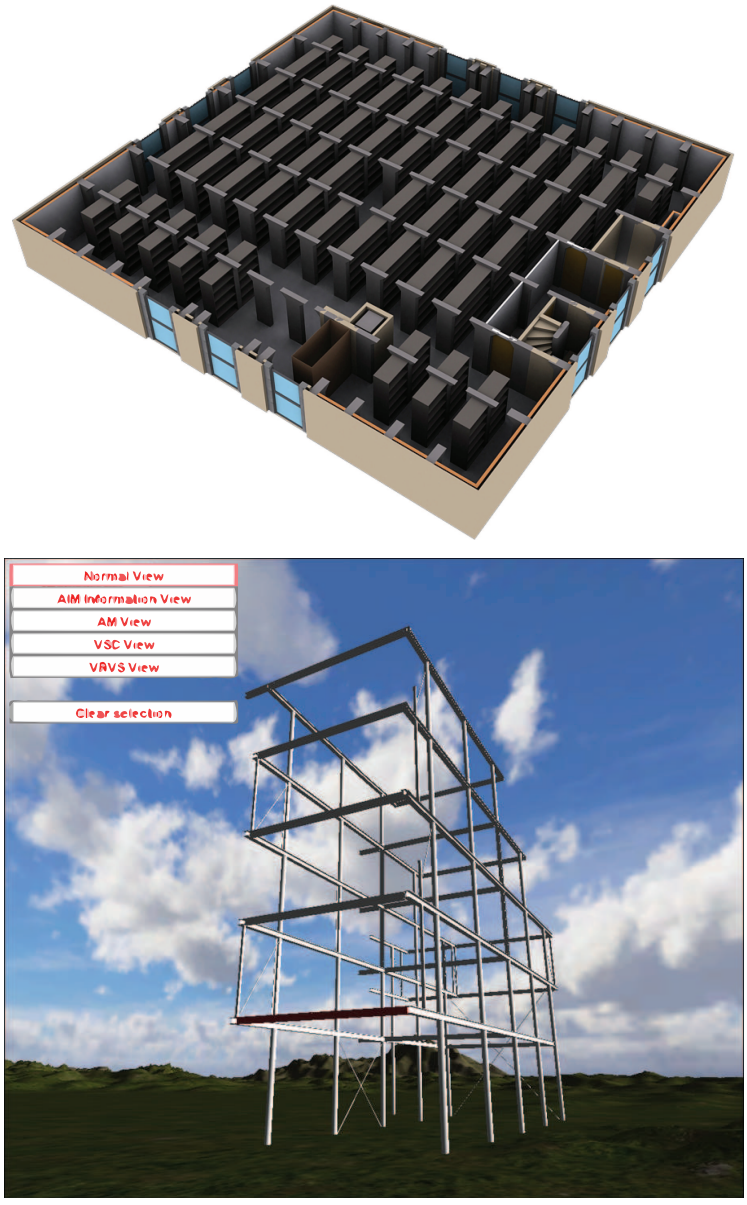


Figure 2.6: Architectural visualizations of a building in Autodesk 3DS Max and Unity [Unity Technologies, 2012], which are applications that are used in an AEC context to produce architectural visualizations.

What is common for both interpretations, however, is that the produced visualization often does not show the information that an AEC specialist needs or desires at a certain moment in time. This is especially problematic in the second interpretation, in which the visualization component is constituted by the GUI. In this case, an image of the building is provided to the end user, showing certain features of the design. A modeling application, however, does not have the amount of analysis data that a calculation application has. Consequently, these features of the building cannot be visualized in this modeling application. In the case of the Port House project in Antwerp, a large number of engineering decisions lies at the basis of the structure of the building façade [Pauwels et al., 2012b]. However, because these decisions were made using MS Excel and not using Revit, a visualization of what lies at the basis of the façade structure cannot easily be obtained, especially not in Revit. For instance, an image that indicates how the building façade is split up in diverse zones, each built up by a specific kind of modules (Fig. 2.7), is unavailable in an architectural visualization that starts only from the BIM model in Revit.

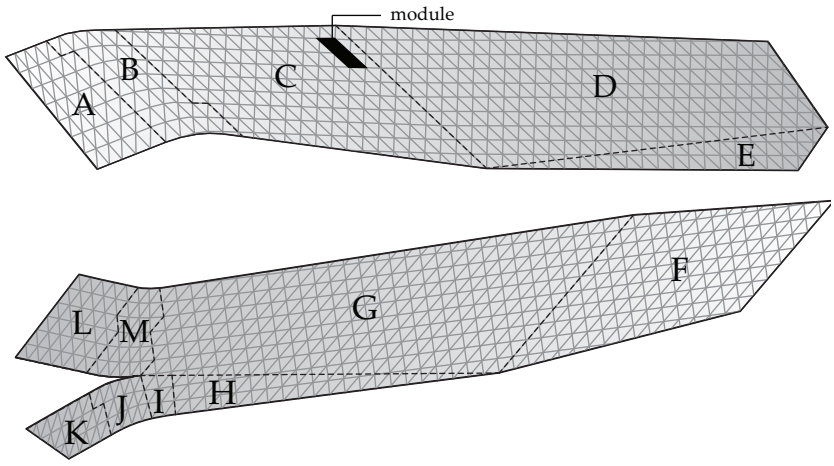


Figure 2.7: Layout of the façades of the Port House, with an indication of the diverse zones in the façade where a modular system can be applied.

2.2 The central issue: information flow

The amount and diversity of information is one of the most notable characteristics of a project in the AEC domain. Many domain experts with dif-

ferent backgrounds typically meet within the context of a building project, each of them composing a personal understanding of the building design and providing with this personal understanding a specific contribution to the project. Additionally, each of these experts relies on diverse software tools, which causes a multiplication of the number of information structures at play in a project. The following information structures are just a few of those used in a design and construction project, in addition to the information structures of the modeling, archive, calculation and visualization applications considered in section 2.1.

- 'designerly' information managed by the architect:
  How are certain elements altered by design decisions? What are the motivations behind specific design decisions? How are specific design requirements addressed in the design?

- material information managed by diverse construction partners:
  What materials are certain design elements made of? What are material characteristics of specific construction elements? How much are the building costs? What are the known advantages and disadvantages of using specific construction elements in certain contexts?

- structural information managed by structural engineers:
  Which elements are central in bearing specific user-loads? How do elements behave in their specific location? What are recommended construction techniques for specific building configurations?

Since these information structures are all part of one and the same project –a project that needs to be finished collaboratively– a lot of information flows emerge between these information structures. These information flows connect the diverse 'information managers' of the project, which are both human users and information systems. The architectural design needs to be communicated to the structural engineer, the structural engineer needs to take into account the design of the electricity engineer, compliance is needed with all kinds of regulations and standards, and so forth. Crucial in this context of continuous information flows are the *interface points* where two or more understandings come together. In these points, information is interpreted from one understanding or information structure into another, thereby making them sensitive to misconceptions or 'mistakes' because of the possible misunderstanding (Fig. 2.8).

Section 2.1 presented indications of where interface points and the corresponding misconceptions can be found for the four considered categories of information systems used in the AEC domain. Because of these misconceptions and resulting mistakes, many designers typically switch to a

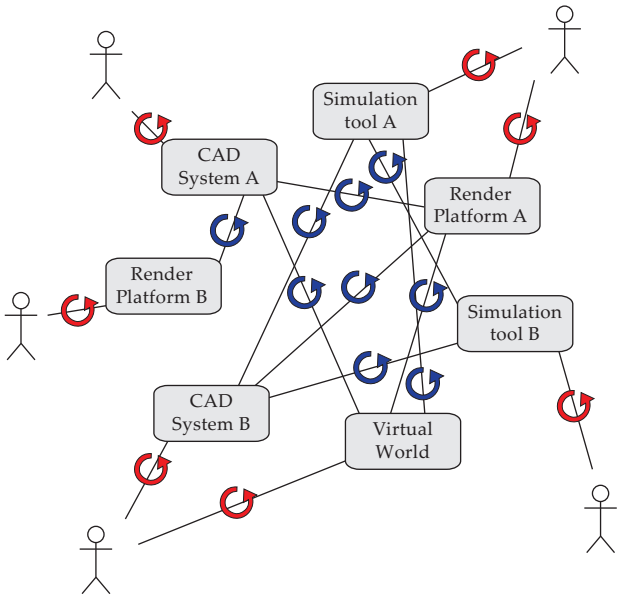


Figure 2.8: Interface points are points where information is interpreted from one information structure into the other, both between human users and information systems (in red) and among information systems (in blue).

more pragmatic approach in which they use information systems for very specific and limited purposes and 'manually' integrate results. One could say that computer-aided design (CAD) applications, for instance, are in this case used as 'computer-aided drafting' environments. The information systems are in this pragmatic software usage often combined with a lot of traditional techniques, such as paper-based sketching, simplified simulation models, and so forth.

Note that this more pragmatic approach is not necessary in all cases. Large architectural design and construction firms are to some extent able to address these issues by developing custom in-house information systems, directly tailored to the needs of the design team. Some successful examples can be named of this approach, namely the 'Digital Project' modeling application implemented and used in the office of architect F. Gehry [Gehry, 2012], and the reliance on a Specialist Modeling Group (SMG) in the office of Foster and Partners [Peters and De Kestelier, 2006]. Although Digital Project relies on CATIA modeling software, important features were added that were of particular use to the architectural design style of architect Gehry. The SMG in Foster and Partners similarly provides custom

design tools compliant with specific needs in specific design projects, leading to custom and on-demand assistance in these projects.

In the following chapters, we will investigate to what extent information system support can be improved for the designers that do not have such a specialized programming team at their command. In our investigation, we will distinguish between interface points between two information systems (displayed in blue in Fig. 2.8) and between a human user and an information system (displayed in red in Fig. 2.8). Of course, one could also consider interface points between two human users, but this is left out of this thesis since we focus here on information system support for architectural design.

As can also be seen in Fig. 2.8, the central issue of information flow can be subdivided in two subissues, namely a lack of *interoperability* among information systems and a *mismatch between the functionality* provided by information systems and the functionality expected by end users. Whereas the first issue relates to information exchange only, the second issue additionally deals with a layer of functionality. In the two following sections (2.3 and 2.4), we will proceed with a discussion of both issues separately.

2.3 Interoperability among information systems

The information flow between information systems is closely related to the notion of interoperability. This is the ability of information systems to connect their information structures and 'work together' effectively. Two levels of interoperability can be distinguished: syntactic and semantic interoperability. These terms have a long history of definitions, relations and understandings [Veltman, 2001], but the terms will be used in their traditionally used senses here. Two syntactically interoperable systems describe information using the same syntax, with syntax defined as an *"orderly or systematic arrangement of parts or elements"* [Simpson and Weiner, 1989]. Two semantically interoperable systems supposedly have the additional ability to interpret the *"signification or meaning"* [Simpson and Weiner, 1989] of the exchanged information and (re)use it. We will concentrate here on semantic interoperability.

Note that we refer here to the term 'semantics' as it is typically used in the context of information systems and software in general. In this context, 'formal semantics' are used for representing the meaning of programs in a formal structure. A particular type of formal semantics are 'denotational semantics'. In denotational semantics, a certain group of concrete expressions are used as representations of abstract objects in the real world. In Hennessy [1990] (p. 36), this is explained using the natural numbers and

the mathematical operators $+$, $\times$, $-$, div. Both the natural numbers and the given mathematical operators are abstract concepts in the real world, constituted only by one's numerous encounters with them in language, arithmetic, simple everyday life, and so forth. As indicated in Hennessy [1990] (p. 36), *"most cultures, including ours, have built up over centuries a conceptual model of the natural number [..] We are all very familiar with the natural numbers and how to reason about them."* Denotational semantics allow to represent these abstract objects using an explicit and concrete notation. *"Let us use N to represent this set of abstract objects, the natural numbers. Then there is a natural mapping or interpretation from the set of symbols, Num, to the set of corresponding abstract objects N. It maps the symbol 0 to the number* 0*, the symbol 1 to the number* 1 *and so on."* [Hennessy, 1990] (p. 36).

The information systems considered in this chapter similarly represent diverse abstract concepts in concrete terms. These terms are far more complex than the ones considered by Hennessy [1990], and include representations for abstract objects such as walls, floors, colors, lines, spheres, and so forth. These are only representations of the actual objects, which only have a meaning within their respective semantic domains. Figure 2.8, for instance, shows diverse semantic domains, including the semantic domains for CAD system A and B, the Virtual World, the Simulation Tool A and B, and so forth. The interoperability issue considered in this section is constituted by the lack of an appropriate mapping between two such domains. This mapping is known as the semantic function between two semantic domains.

Below, we look into diverse strategies that can be used to produce mappings or semantic functions between semantic domains in the AEC domain, and thus to address the interoperability issue.

2.3.1 Sharing information in the wild

Information used within a design and construction project can be described in many ways, with both a varying syntax and varying semantics. Additionally, this information is so diverse that no single information structure can describe it all. This results in a large set of specialized information structures between which conversions are inevitable. This naturally evolves into the situation shown in Fig. 2.9, in which information is being converted from one information structure into the other as needed.

The actual connection between two information systems often looks as is indicated in Fig. 2.10, with every transition between two information systems consisting of at least two interface points between each of the information systems and one exchange file format. For example, the .DWG

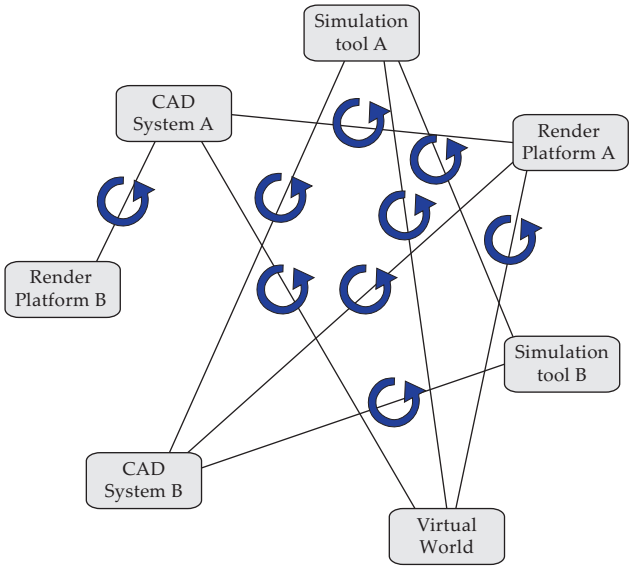


Figure 2.9: The traditional information flow between information systems: sharing information 'in the wild'.

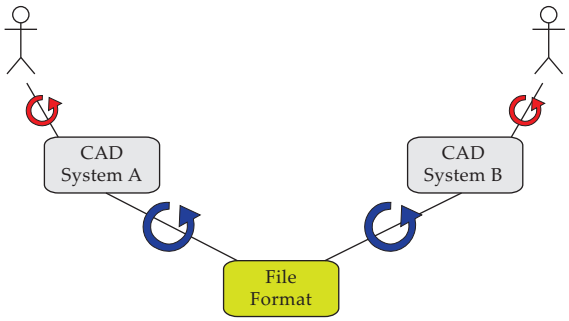


Figure 2.10: The information flow between two CAD systems using one file format contains two interface points in which information needs to be converted.

file format is an often used file format to communicate between diverse Autodesk modeling applications [Autodesk, 2012a]. In this situation, the interface points are materialized by the import and export functions of the applications at hand. Both the import and the export function constitute a mapping between the information structure of the application and the information structure of a certain file format.

Typically, an application is able to save its information into a proprietary file format without many problems. But additionally, the application is required to export to and import from diverse other file formats. These file formats typically are the proprietary file formats of other software applications in the AEC domain. For reference, Fig. 2.11 gives an overview of the possibilities of going from Autodesk AutoCAD 2011 to Autodesk 3dsMax 2011, using only one file format.

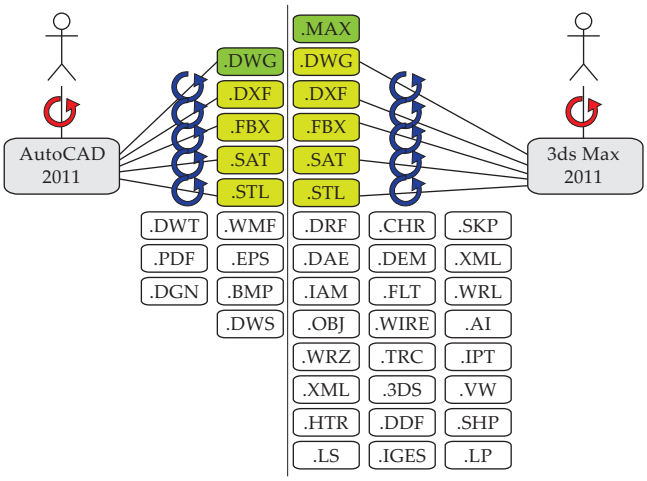


Figure 2.11: The five possible information flows between Autodesk AutoCAD 2011 and Autodesk 3ds Max 2011 using one file format (.DWG, .DXF, .FBX, .SAT, .STL). Both programs' native file formats (.DWG, .MAX) are indicated in green, possible exchange formats are indicated in yellow (.DXF, .FBX, .SAT, and so forth).

Some file formats contain animation data, some describe building components, others are used for interactive web applications, and so forth. Because a different part of the AEC domain is described in each of these file formats, each file format tends to use a partly unique structure with an equally unique syntax and semantics, making it impossible to create a complete and exact mapping between an application and any of those file formats. Such import and export functions are used anyway, resulting in a loss of information. The lost information needs to be remodeled, lead-

ing further in the process to errors and limitations in the design conception stage and to inefficiency due to the required remodeling efforts [Gallagher et al., 2004].

In some cases, additional conversions between file formats are needed, as is shown in Fig. 2.12. These additional conversions are either realized by a transition through another application, or by dedicated and freely available, but in many cases also incomplete, conversion tools. These tools do exactly the same as the import and export functions discussed before, namely mapping between diverse information structures, only in this case the mapping occurs between file formats only.

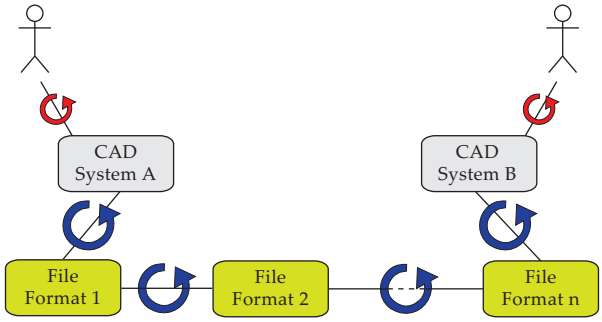


Figure 2.12: The information flow between two CAD systems using multiple file formats contains multiple interface points in which information needs to be converted. The number of interface points depends on the number of file formats used.

These extra conversion steps increase the number of interface points and thus result in a further loss of information, equally leading to errors and limitations in the design conception stage and to inefficiency due to the required remodeling efforts [Gallagher et al., 2004]. Such extra conversion steps are typically used when an application cannot export to or import from a certain file format, most often because the application provides a notably different functionality. For example, the transition between a 3D modeling environment in architecture, such as AutoCAD [Autodesk, 2012a], and a game engine environment, such as Unity [Unity Technologies, 2012], requires several file format transitions. But also a transition over the SketchUp application when going from AutoCAD 2011 to 3ds Max 2011 opens up several possibilities, as is shown in Fig. 2.13.

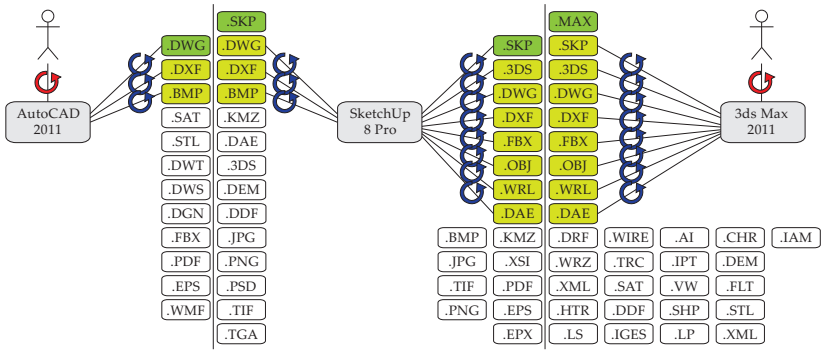


Figure 2.13: The possible information flows between Autodesk AutoCAD 2011, Google SketchUp 8 Pro and Autodesk 3ds Max 2011. The programs' native file formats are indicated in green, possible exchange formats are indicated in yellow.

2.3.2 The remodeling effort

The remodeling effort strategy, which is shown in Fig. 2.14, is a rather pragmatic and ad hoc approach towards interoperability. Instead of trying to use file exchange mechanisms (conversion, import/export), which typically result in a certain loss of information, information is exchanged between the users themselves, who are in charge of their own versions of the design model.

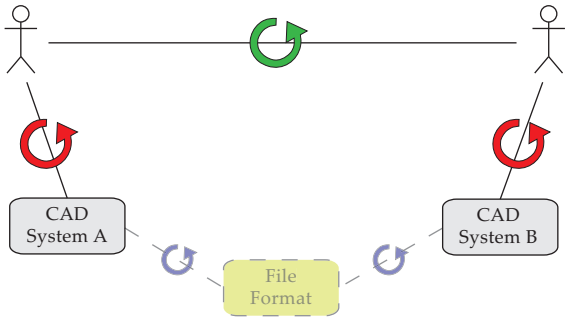


Figure 2.14: In the remodeling effort strategy, information is primarily exchanged between users, possibly with additional reliance on file exchange mechanisms.

This approach is often combined with the approach of sharing information in the wild (Fig. 2.9). As was also indicated in section 2.3.1, this approach results in an important amount of information loss when a de-

signer switches between different applications and file formats. Even when converting back to the original file format, the loss of information is irreversible. Therefore, the information needs to be remodeled. However, manually remodeling information similarly results in a loss of information [Gallagher et al., 2004], both in the communication between the human users and between the user and the application(s) in which the information is modeled. This remodeling approach does not solve the issue of interoperability, but just puts the end user back in charge.

2.3.3 Kernel-level interoperability

Another approach, suggested in the domain of 3D information exchange, is kernel-level interoperability. Most of the applications in the AEC domain rely on a 3D modeling kernel. As is indicated by Gerbino [2003], this kernel is responsible for storing and organizing the basic geometric shapes and model topologies used by that application. Some well-known kernels used by CAD applications are ACIS (.SAT file format), Parasolid (.X_T file format) and Open Cascade (.CSFBD file format). A CAD application thus provides a whole range of functionalities that rely in their foundations on the functionalities offered by the kernel. The CAD information structure might thus be considered as an extension of the more basic information structure of the 3D kernel. When two applications rely on the same kernel, they essentially have the same basis underneath their information structures.

Kernel-level interoperability relies on this common basis to optimize information exchange between these information systems. This approach might be of certain use in a pure 3D context. In such a context, it is advisable to exchange 3D information between applications with a common kernel in the file format of this kernel (.SAT, .X_T, .CSFBD). In this case, the 3D information is brought back into its basic form, making it understandable for the other application. In the other application, the 3D model can be reconstructed into its more advanced description. However, the original advanced description is not communicated, so there is a certain loss of information in the communication. The application into which the 3D model is imported, is supposed to reconstruct this advanced description from the kernel-level description solely. This approach might be feasible to some extent for pure 3D information, but it is highly unpredictable in an AEC project, because this project involves more detailed feature information, such as wall parameters, floor types and attributes, and so forth.

Furthermore, it does not work well between applications that use a different 3D modeling kernel, which is often the case in the AEC domain

[Gerbino, 2003]. In this case, kernel-level information exchange is just as reliable as any of the other file formats, which is also indicated in Fig. 2.11 for the ACIS kernel (.SAT file). In conclusion, this approach can merely be considered as a part of the approach of sharing information in the wild, which is shown in Fig. 2.9.

2.3.4 The centralized information structure

One of the latest approaches gaining significant support in the AEC domain is the Building Information Modeling (BIM) approach, in which one central 3D building model is to be used as a centralized information structure by several applications [Eastman et al., 2008] (Fig. 2.15). All information is stored in a central BIM model, which can be accessed from within diverse other applications in the AEC domain. Since all information is stored in one central model, all this information is always available for all users. Changes made to the design are applied to and stored in the BIM model, thereby making them directly available to other users.

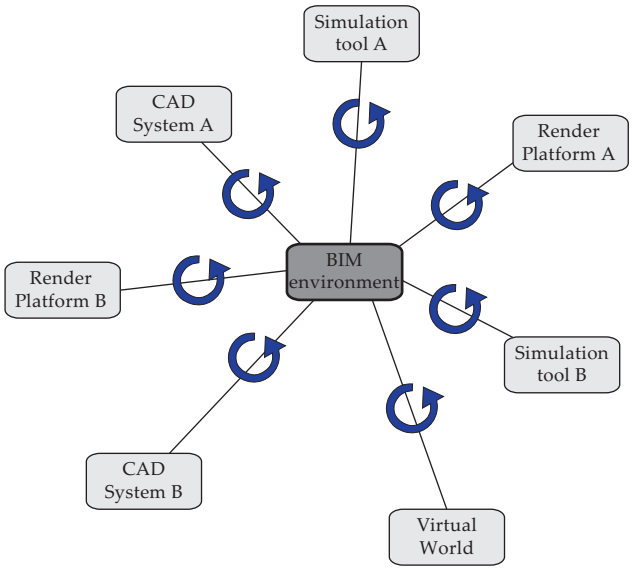


Figure 2.15: When relying on a centralized information structure, information exchange between information systems is based on a central building information model.

Although this approach appears to eliminate some interface points, several such points persist in this approach (Fig. 2.15). These interface points are, however, seldom included in overviews of this BIM strategy [Graphisoft, 2012]. This gives the impression that information can perfectly be exchanged with any of the surrounding applications, which is not the case in any of the existing BIM applications [Jeong et al., 2009, Pauwels et al., 2009c, Pazlar and Turk, 2008, Plume and Mitchell, 2007, Verstraeten et al., 2008].

The schema in Fig. 2.15 appears to suggest that one can build a central information structure that is capable of describing *all* the information possibly needed in any of the applications used in an AEC project. This suggestion is also made within diverse research initiatives towards a 'standard' or 'neutral' information structure for all building information [Liebich et al., 2012]. Examples of such suggested standards include not only proprietary industry standards, such as .DXF, .FBX, .IGES, or .DWG, but also 'neutral' formats, such as .STEP, .IGES, .X3D, or .IFC. Over time, however, these standards merely tend to turn into yet another file format the user needs to convert information to or from, and the actual conversion issue is not solved. Both the results from the BIM approach and the results from the usage of standards [Jeong et al., 2009, Pauwels et al., 2009c, Pazlar and Turk, 2008, Plume and Mitchell, 2007, Verstraeten et al., 2008] indicate that it is not possible to rely on one central information structure that is capable of describing *all* building information. The centralized information structure as depicted in Fig. 2.15 is thus not feasible. In reality, the central information structure is just one of the many available information structures (Fig. 2.16).

2.3.5 The software suite strategy

The software suite strategy might be considered as a mix of the kernel-level interoperability approach and the centralized information structure approach. In this strategy, a specific software suite from one product vendor is used by as much actors in the AEC project as possible (Fig. 2.17). An example is the Autodesk software suite, which includes applications such as AutoCAD, Revit and 3ds Max. This strategy assumes that the applications within this suite are all implemented using a similar 3D modeling kernel and similar top-level information structures. Because this results in better chances for understanding each other, this might be a practical approach towards addressing the interoperability challenge.

On the other hand, this approach limits the user in choosing the appropriate application for the task. As soon as one wants to exchange in-

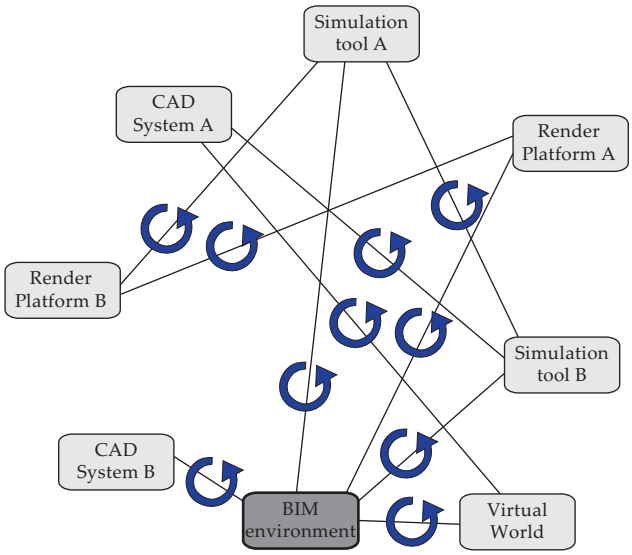


Figure 2.16: When relying on BIM software, 'standard' file formats or any other centralized information structure, this structure is in reality just used as one of the many available information structures.

formation with information systems outside the product suite, the original interoperability challenge returns (Fig. 2.17). And even if one sticks to the information structure provided by the software suite, this approach is essentially identical to the first approach of sharing information in the wild (Fig. 2.9), but with a limited number of information structures.

2.3.6 The linked data approach

A last approach is to separate the actual data from the applications they respectively reside in: the linked data approach. This approach has been suggested several times over history in diverse colors, forms and names. The Windows 'Object Linking and Embedding' (OLE) technology, for instance, enables linking and embedding information from one application into the other, for instance, Microsoft Excel and Microsoft Word. By doing so, the information structures or object models of the information systems are linked on a data level, as is shown in Fig. 2.18, making the information sharable between the applications. As soon as one wants to use this information in an application outside this web of linked data, however, the

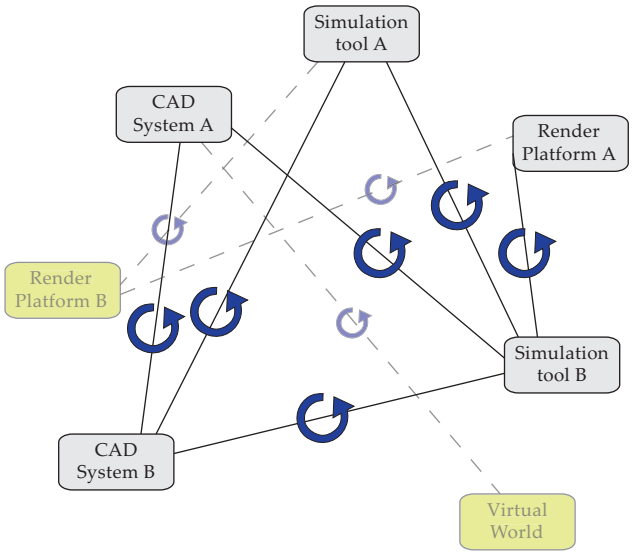


Figure 2.17: The software suite strategy includes several 'preferred' information flows, namely those between applications of the same software suite.

situation is back to what it was before: not interoperable. This approach thus works more or less like the software suite strategy depicted in Fig. 2.17, only implemented more on a data level.

More recent examples in which this linked data approach is implemented, can be found in the semantic web domain [Berners-Lee et al., 2001]. Using semantic web technologies, one is able to describe information in a directed labeled graph using a common language. Continuously extending this directed labeled graph results in a globally interconnected semantic web, or a linked data cloud [Bizer et al., 2009, Cyganiak and Jentzsch, 2011], which is directly connected to the ontologies or description structures structuring its information. As such, these technologies allow one to combine information models used in diverse information systems, with respect for the inherent semantics and syntax of each of these subgraphs (Fig. 2.18).

A semantic web approach was already suggested to improve the interoperability of CAD information, for instance by Abdul-Ghafour et al. [2008]. The authors indicate how semantic web technologies allow the combination of information from several different knowledge domains, enabling a seamless coupling of 3D information with non-geometric informa-

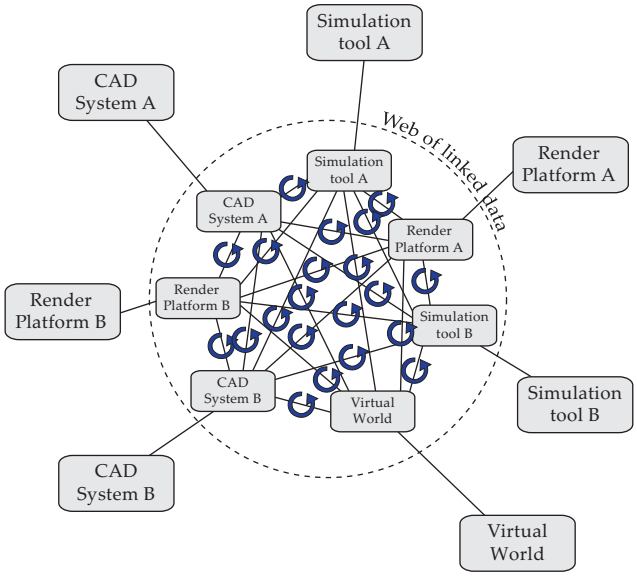


Figure 2.18: In the linked data approach, information is linked on a data level. This results in a web of linked data that is accessible for any application that wants to use it.

tion, such as design intent and domain-specific product features. Similar suggested approaches relying on semantic modeling of product information, not necessarily targeting improvements regarding the interoperability issue, can be found in Abdul-Ghafour et al. [2007], Böhms et al. [2009a,b], Kraft and Nagl [2007], Yang and Zhang [2006]. Pauwels et al. [2010a] similarly presents how semantic web technologies enable the integration of architectural design information with general AEC and 3D information available through the IFC schema.

Two important reasons why this approach might be better compared to the other approaches, is (1) that semantic web technologies rely on a common language for describing information, namely the Resource Description Framework (RDF) [Manola and Miller, 2004], and (2) that semantic web technologies appear to be deployed on a global scale [Bizer et al., 2009, Cyganiak and Jentzsch, 2011]. Consequently, information that would typically be unavailable in the software suite strategy (Fig. 2.17) has a notably higher chance of being incorporated in the linked data approach, making this currently one of the most promising approaches.

2.4 Functionality mismatch between information systems and end users

As indicated in section 2.2, there is a second element in the central issue of information flow, apart from the information flow among information systems. Namely, there is also a mismatch between the functionality provided by information systems and the functionality expected by end users (Fig. 2.8). Not only is the information flow problematic in these interface points, also the functionality layer appears to be problematic.

Section 2.1 already showed some example problematic interface points. Functionalities provided by modeling applications are either 'not enough and too simple', or 'too much and too complex'. The functionality provided by simulation applications is 'not correct' or 'irrelevant'. The visualization produced by visualization applications 'does not communicate the required information'. And archive applications typically contain only the information one 'does not need'. We will not look into these issues in too much further detail, but instead we will look at the parallel with the lacking interoperability among information systems that was outlined in section 2.3, and we will look into some improvements to this issue that might result from a linked data approach as suggested in section 2.3.6.

2.4.1 The parallel with the lack of interoperability

For all application types outlined in section 2.1, a similar problem occurs in the interface points between human users and information systems: the information presented by the system does not conform to the needs and/or desires of the end user. It is as if two different information models or semantic domains are maintained, by the human user and by the information system, and both models do not match. As well in the mind of the architectural designer as in the information structure of the information system, an information model is maintained for the design situation at hand. The resulting functionality, which is in more concrete terms human interaction and output from the information system, respectively, is based on this information model. In many cases, the underlying information model is notably different, and, because this information model lies at the basis of the provided functionality, also the resulting functionality is different. The resulting 'misunderstandings' thus constitute the functionality mismatch earlier outlined.

Clear examples supporting this argument can be found in the interaction between designers and modeling applications (AutoCAD, SketchUp, 3dsMax, Rhinoceros, etc.). Each modeling application provides a specific

functionality to the designer, and relies on an application-specific information model to achieve this specific functionality. We have seen how this results in problems when information is to be exchanged between these modeling applications (indicated in blue in Fig. 2.19). But a designer similarly relies on a certain understanding of a design situation, which might be simplified and represented as an information model of its own in this section. This information model in the designer's mind differs at least as much from the information model in the modeling application. For instance, certain architects mainly understand the design of a building in terms of historical references and architectural theories. These concepts are seldom included in modeling applications such as the typical 2D CAD environments, resulting in a mismatch of functionality. Whereas the designer wants to model the design in terms of historical references and architectural theories, the modeling application only allows to use simple lines and points. And even when similar concepts appear to be present in the application and in the mind of the designer, these concepts often have different connotations in both contexts. In this case, the functionality mismatch issue is somewhat lessened, but remains present nonetheless. For the Port House example mentioned earlier [Pauwels et al., 2012b], a window element as understood by the engineering team cannot be described using the standard information structure for describing windows in Revit (see also Fig. 2.3). And even when using 'generic models' to describe one of the windows in the Port House, the geometric shape of the window also does not entirely match the shape as it is understood by the engineering team. In this case, concessions can be made by the engineering team in function of the desired end product. An analogous discussion can easily be made for calculation applications, archive applications and visualization applications.

We want to clearly distinguish the considered functionality mismatch issue from issues related to information visualization. Once the information is available that is required or desired by the end user, namely, there are still diverse ways available to visualize and present this information to the end user. As shown in Fig. 2.20 (left), one might choose to rely on the same central source of information for producing diverse visualizations of the same information for different end users or end user groups. This is one of the issues that could be considered in the specialized domain of information visualization [Card et al., 1999, Chen, 2004, Purchase et al., 2008, Spence, 2006, Ware, 2004] and to some extent also in the domain of information aesthetics [Lau and Vande Moere, 2007, Manovich, 2012]. The main goal of information visualization is to *"amplify human cognition through the use of computer-supported, interactive, visual representations of abstract data"*

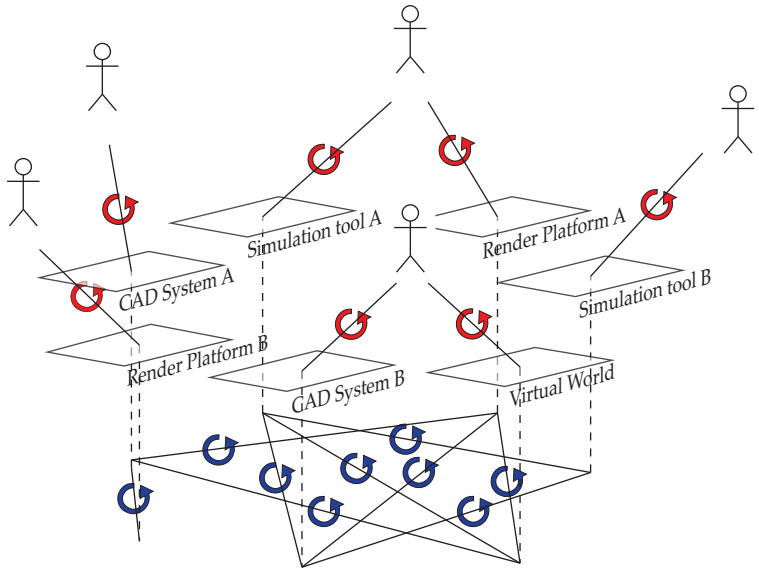


Figure 2.19: The issue of information flow, as outlined in Fig. 2.8, can be subdivided in an interoperability issue among information systems (bottom) and a functionality mismatch issue between information systems and users (top).

[Card et al., 1999]. These techniques can most likely help addressing the functionality mismatch issue. But, first and foremost, the correct information needs to be available in the correct syntactic and semantic structure. As indicated before, this should be the information and the corresponding structure desired and required by the end user. So, an environment is apparently needed in which diverse interlinked information models are available, each representing the understanding of a design situation by a certain designer (right in Fig. 2.20). These information models should be available to corresponding end users through an interface, and should be linked to other information models so that information exchange is possible.

As an example, let us reconsider the architect that desires to model a design mainly in terms of historical references and architectural theories. It is not enough to use information visualization techniques for adapting a central information structure to the needs of this end user (left in Fig. 2.20). First, a separate information structure is needed that can be used to represent the historical references and architectural theories that the designer wants to use, possibly linked to other information models. This informa-

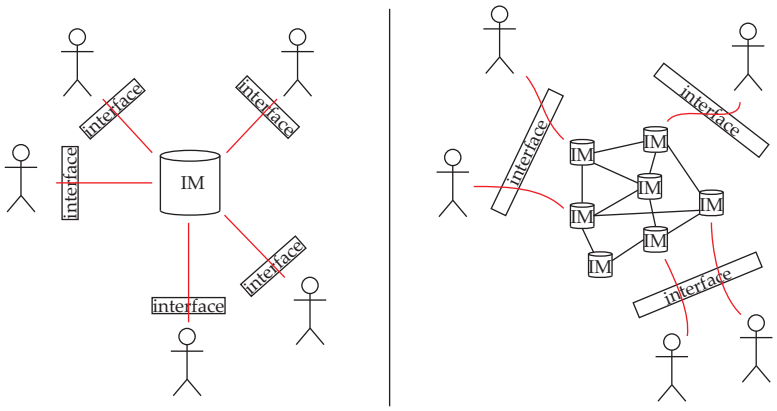


Figure 2.20: Visualizing a central information model (IM) in diverse ways to end users (left) versus providing the information models to the users that require them (right).

tion structure serves as a separate semantic domain (see Hennessy [1990]) that enables representing the considered abstract objects following the designer's needs. Afterwards, information visualization techniques may be deployed to produce an optimal visualization of this information. If such a system is created for all partners in an AEC project, a situation as depicted in Fig. 2.21 results. Diverse interlinked information models are available in a web of linked information, each information model following the information structure desired by specific end users. Designers can then get access to information structured according to their understanding and the application might consequently provide the functionality required by the designers.

2.4.2 Improvements anticipated in a linked data approach

In section 2.3, the finally suggested linked data approach based on semantic web technologies came out as a promising approach towards the issue of interoperability among information systems. Semantic web technologies (1) rely on a common language for describing information and (2) appear to be deployed on a global scale (section 2.3.6). Consequently, chances are higher that information models in a specific application are linked to information models used in other applications. As such, they might at least be able to form an improved solution towards the encountered issue of interoperability.

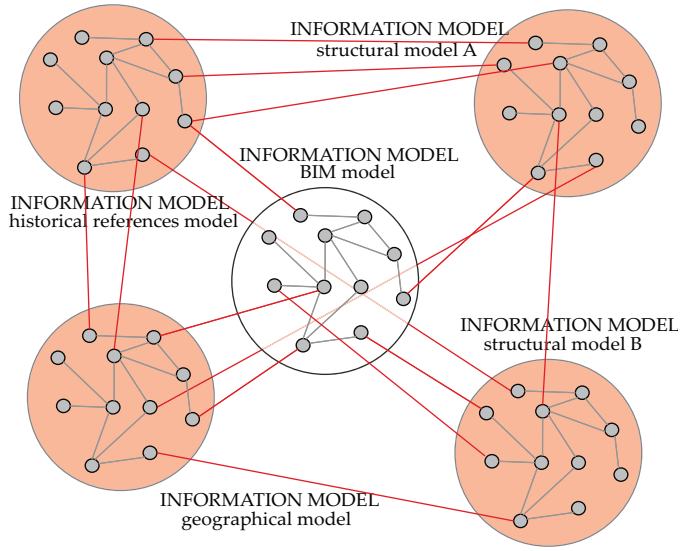


Figure 2.21: Linking diverse information models might not only improve interoperability among information systems, but might also enable the implementation of applications tailored to the needs and desires of specific users or user groups. As such, this image corresponds to the central part shown in Fig. 2.20 (right). For instance, a structural model that corresponds to the understanding of a specific engineering team (structural model A) could be linked to available other information models, so that applications can provide the appropriate functionality to this engineering team and maintain the link to other understandings of the considered design situation (historical references model, BIM model, structural model B, geographical model).

Considering the parallel documented in section 2.4.1, this linked data approach might result in similar improvements for the functionality mismatch issue. Analogously, the linked data approach might enable architectural designers to model their understanding of a design situation independent of the information models available in modeling, calculation, archive or visualization applications. The resulting information model is then accessible to applications, which can use these information models to provide the functionality required by the end user (right in Fig. 2.20).

We will briefly look into the 'CultureSampo' project as an example project in the domain of cultural heritage [Hyvönen et al., 2005, 2009a,b, Mäkelä et al., 2012]. This project similarly relies on a linked data approach with semantic web technologies for combining diverse information models and that provides diverse services and applications on top of this information, each time using a specific part of the information. Similar to the AEC

domain, the cultural heritage domain also encompasses very diverse kinds of information. This is illustrated by the following passage in Mäkelä et al. [2012]: *"As an example, in Finland the Finnish National Gallery holds a painting by the painter Akseli Gallen-Kallela, depicting a scene from the Finnish national epic, Kalevala, collected by Elias Lönnrot. The original poems are available as a database provided by the Finnish Literature Society. There is also a separate database containing further information on each passage, actor and imaginary place of the epic poem. In addition, the National Biography contains biographical information about Lönnrot and Gallen-Kallela and 6000 other famous Finnish authorities, whose life stories may be mutually intermingled. At the same time, the Agricola-network of Finnish historians holds a database of historical events, some pertaining to Kalevala, and others to Gallen-Kallela and Lönnrot. Yet further, the international Union List of Artist Names by the Getty foundation contains structured information on other people Akseli Gallen-Kallela worked with as well as his roles in society. The video archives of the Finnish Broadcasting Company contain videos related to all of these themes. In addition, all the sources contain lists of related places, times and content keywords which further link the material to museum collection objects, photographs, historical buildings, and so on. As a final source of additional information, Wikipedia contains further peer-curated information on nearly all aspects of life, given that they are notable enough."* One and the same element, in this case the person Gallen-Kallela, can thus be considered from very diverse perspectives, similar to the way in which a design situation or a building can be considered from very diverse perspectives. Typically, each perspective is described in a bounded environment. In the case of Gallen-Kallela, these bounded environments are distinct databases, each managed by a different institute, whereas in the case of a design context, these bounded environments are the information models presented in Fig. 2.20 (right) and 2.21.

In the CultureSampo project, these databases were integrated using semantic web technologies, resulting in a graph that combines the available information models, similar to how it was presented in section 2.3.6 (Fig. 2.18). The resulting 'knowledge base' is then accessible from within various applications, among which a WWW browser, as indicated in Fig. 2.22. As such, information is integrated from over thirty organizations [Mäkelä et al., 2012].

After all information was integrated with semantic web technologies, applications were built on top of this information. The main application that resulted from this effort, is the online semantic portal 'CultureSampo' itself. This portal provides diverse perspectives on the underlying web of information. As such, it can be compared to the rightmost schema in Fig. 2.20: diverse information models are combined in a linked data approach,

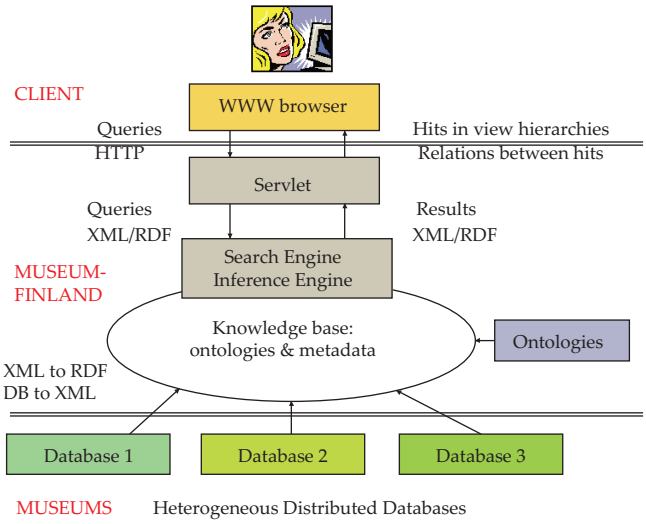


Figure 2.22: Architecture of the semantic portal 'MuseumFinland' as shown in Hyvönen et al. [2005]. Local database contents are merged and made available for query access, which can be used by diverse applications and users.

and diverse user interfaces are presented to the end user, each time presenting specific information tailored to the requirements of the end user. Examples of generated user interfaces can be found in Mäkelä et al. [2012]. This includes visualizations of selected cultural heritage artifacts in a timeline interface or a geographical map interface, for instance. But also very specific interfaces can be generated. For instance, an information model was added that allows the representation of historical areas related to their borders and their artifacts [Mäkelä et al., 2012]. An interactive interface can be built on top of this information, as shown in Fig. 2.23, allowing exploration by the end user. Mäkelä et al. [2012] similarly indicates how the 'knowledge base' in Fig. 2.22 was extended with additional related information models by the BookSampo project [Mäkelä et al., 2011]. This project includes information models about Finnish fiction literature and links this to content in the CultureSampo knowledge base. Information is thus reused in different contexts by different users using different functionality.

This project primarily focuses on the integration of heterogeneous information sources and providing relatively simple views on parts of this information. Only to a limited extent is application functionality also adapted to the actual content of the information: maps are used for pre-

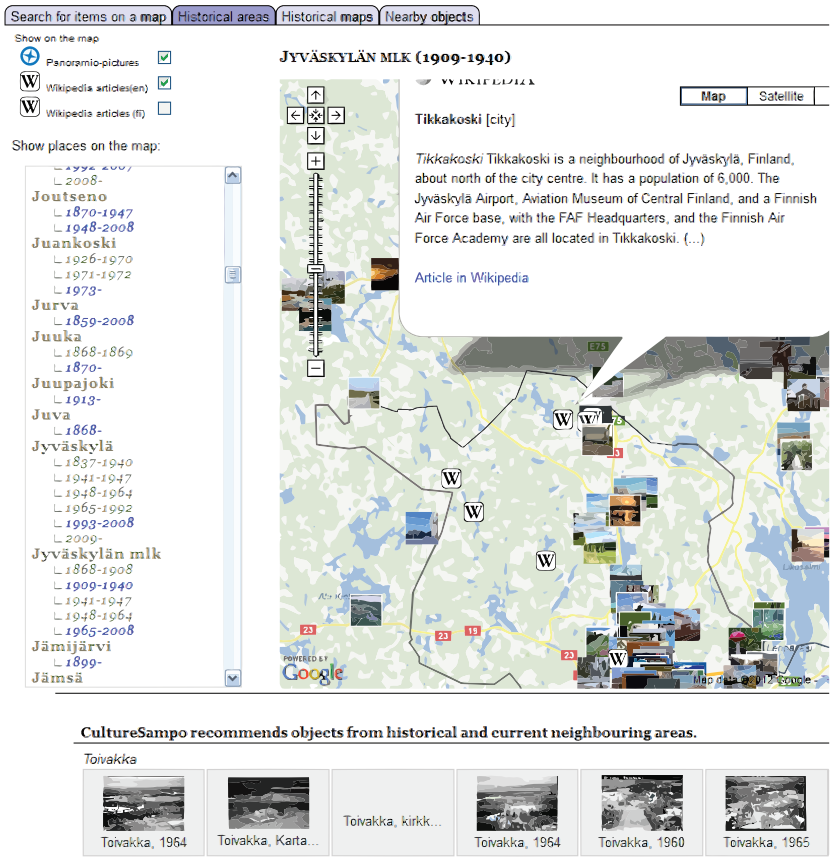


Figure 2.23: Historical borders as viewed in 'CultureSampo' [Mäkelä et al., 2012] (source: CultureSampo [2012]).

senting geographical information, timelines are used for presenting time-based information, and so forth. Nevertheless, it seems at least feasible to extend this approach to AEC information and to allow the use of specific information model(s) to produce an information system tailored to the requirements and desires of specific designers or groups of designers.

2.5 Conclusion

We have seen in this chapter how applications in the AEC domain often provide only a limited support to partners in a design and construction project due to a malfunctioning information flow. This malfunctioning in-

formation flow can be subdivided in two main issues: a lacking interoperability among information systems and a functionality mismatch between information systems and end users (Fig. 2.19). Notwithstanding the significant amount of effort put into the design and implementation of applications for the AEC domain, these issues return time and again in the evaluation of software usage in the AEC domain.

We have looked into strategies for addressing the interoperability issue, resulting in a brief discussion of the following strategies:

- Sharing information in the wild
- The remodeling effort
- Kernel-level interoperability
- The centralized information structure
- The software suite strategy
- The linked data approach

The linked data approach is suggested as one of the most promising strategies for addressing interoperability issues, mainly because (1) these technologies rely on a common language based on a logical foundation for describing information and because (2) these technologies appear to be deployed on a global scale. The latter reason is important, because, no matter how standard or logical a language might be, it needs to be used by information systems to enable information exchange among this group of information systems. The larger the group of information systems using a language, the more information systems can exchange information.

Next in this chapter, we have made a parallel between the functionality mismatch issue and the interoperability issue. It was outlined how the information presented by information systems does often not conform to the needs and/or desires of the designer. To address this issue, the information model maintained by the designer, which includes the terms and concepts typically used by this designer, should to some extent reflect in the information structure used by the information system. The functionality mismatch issue thus translates into an interoperability issue, in the sense that designers should be able to represent the information structure they are using and relate this to existing information structures, so that applications can take the information model of the designer into account when providing certain functionality.

Following the parallel between the functionality mismatch issue and the interoperability issue, a linked data approach based on semantic web

technologies might thus to some extent also be used to address the functionality mismatch issue. An indication is given of anticipated improvements by documenting a recent similar project relying on a linked data approach to present cultural heritage information to specific users and/or groups of users.

3

A linked data approach for information exchange in the AEC domain

IN chapter 2 it was concluded that a linked data approach based on semantic web technologies might allow addressing the interoperability issue in the AEC domain (section 2.3), because they allow to explicitly and unambiguously connect information deployed in diverse applications. This approach might additionally generate improvements regarding the functionality mismatch issue (section 2.4). In this chapter we pursue this line of thinking and show how this approach was tested in this research project for the AEC domain. Before looking into concrete examples for the AEC domain, we will start with an overview of semantic web technologies.

3.1 Semantic web technologies

In this section, an overview is given of semantic web technologies. Because extensive documentation of these technologies exists elsewhere [Berners-Lee et al., 2001, Brickley and Guha, 2004, Grant and Beckett, 2004, Manola and Miller, 2004, McGuinness and van Harmelen, 2009, W3C, 2012], this overview will be kept brief and limited to the concepts needed for understanding the following sections.

The semantic web was conceived and suggested by Berners-Lee et al. [2001] as the successor of the existing World Wide Web (WWW). In this semantic web, all information would supposedly be described in a language that can be 'understood' by computer applications. Because the WWW contains information about almost any possible concept in the world, the language describing this information can not follow one domain-specific schema. Instead, a flexible and generic language is needed that allows one to describe and easily combine information from very different knowledge domains.

Therefore, the semantic web was conceived as a semantic network [Shapiro, 2003] in which diverse semantic domains can be represented and combined using directed labeled graphs (Fig. 3.1). A directed labeled graph graph simultaneously represents several binary relations, and the labels on the arrows of the graph identify the particular relation. Each node in such a graph thus represents a concept or object in the world and each arc in this graph represents the logical relation between two of these concepts or objects. A graph can be constructed using the Resource Description Framework (RDF) [Manola and Miller, 2004] and the Web Ontology Language (OWL) [McGuinness and van Harmelen, 2009], which have a basis in description logic (DL) [Baader and Nutt, 2003, McGuinness and van Harmelen, 2009]. The graph is thus formed by a set of logic-based declarative sentences and, in total, it represents a specific semantic domain as it is understood and explained by Hennessy [1990]. By describing information in a single directed labeled graph, a uniform representation of information is targeted, making information reusable by both humans and computer applications.

Today, the development of the semantic web is mainly led by the World Wide Web Consortium (W3C) [W3C, 2012], strongly supported by actors stemming from various domains, including both scientific research institutes and industrial partners. According to the W3C, the semantic web nowadays consists of a web of 'linked (open) data (LOD)' [Bizer et al., 2009] that is superseding the borders of individual applications and hence is interconnected through links between identical or related entities (Fig. 3.2). According to Bizer et al. [2011], the LOD cloud in Fig. 3.2 contains 31,634,213,770 RDF statements, where each statement can be understood as one of the arcs shown in Fig. 3.1.

3.1.1 Resource description framework

Semantic web technologies rely on RDF as a data model for representing information in graph structures [Manola and Miller, 2004]. These graph

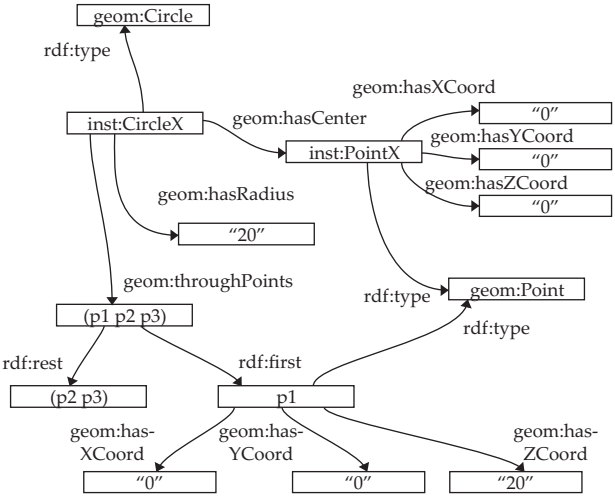


Figure 3.1: A directed labeled graph representing three-dimensional information. In this case, a circle (`inst:CircleX`) is described (1) by a center point (`geom:hasCenter`) and a radius (`geom:hasRadius`), and (2) by three points through which the circle passes (`geom:throughPoints`).

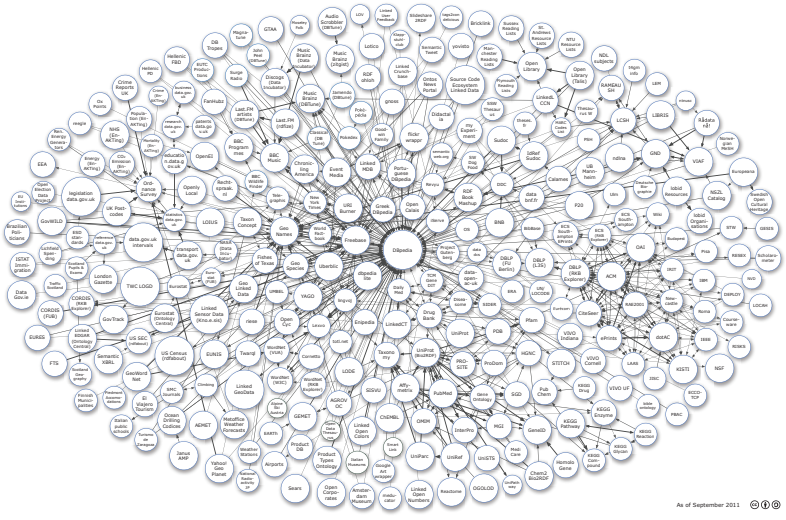


Figure 3.2: The Linking Open Data cloud diagram, by Cyganiak and Jentzsch [2011] (original image from Cyganiak and Jentzsch [2011]).

structures are generally referred to as RDF graphs. RDF graphs can thus be considered the representations of data that follow the RDF data model. Within the semantic web domain, RDF was soon considered a standard language to represent any possible kind of information, not limited to the WWW. Information thus implicitly became interoperable between different environments, whether these environments be web pages, complete software environments, or anything else [Manola and Miller, 2004].

An RDF graph is constructed by making conjunctions, i.e., applying a logical `AND` operator to a list of logical statements representing concepts or objects in the world and their relation. The directionality of the RDF graph is realized through the 'RDF triple' structure (Fig. 3.3), and the graph is labeled with Unique Resource Identifiers (URIs) for both concepts and relations. Every subject, predicate and object in an RDF graph is uniquely defined through this URI. When two identical URIs are found, their semantics are considered identical as well.

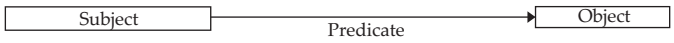


Figure 3.3: The 'triple' form of an RDF statement: subject - predicate - object.

An RDF graph can be expressed in diverse textual representations that each follow a specific syntax. This is similar to how a data model can be serialized into diverse textual representations. Syntaxes used for RDF graphs are RDF/XML, N-Triples [Grant and Beckett, 2004], SPARQL [Prud'hommeaux and Seaborne, 2008], Turtle [Beckett and Berners-Lee, 2011] and Notation-3 (N3) [Berners-Lee and Connolly, 2011]. We use the N3 syntax because it focuses on human readability and the possibility to express both data and logic in the same language [Berners-Lee and Connolly, 2011]. The N3 syntax additionally provides one of the highest levels of expressiveness within the semantic web domain [Berners-Lee and Connolly, 2011], although certain information can still only be expressed in RDF/XML and/or SPARQL (Fig. 3.4).

Fig. 3.5 illustrates how a window may be represented using N3. The first two statements define two 'prefixes' (`inst:` and `ont:`) that enable the abbreviation of the URIs used in the RDF statements below. Using the `inst:` prefix for instance, one can reconstruct the URI of the resource `inst:WindowX` into <`http://smartlab.elis.ugent.be/aimontologies/inst/SLwindow#WindowX`>. The lower lines describe four statements related to the concept `inst:WindowX`, concatenated by a dot ('`.`') as a representation of the logical `AND` operator. The predicates `ont:-`

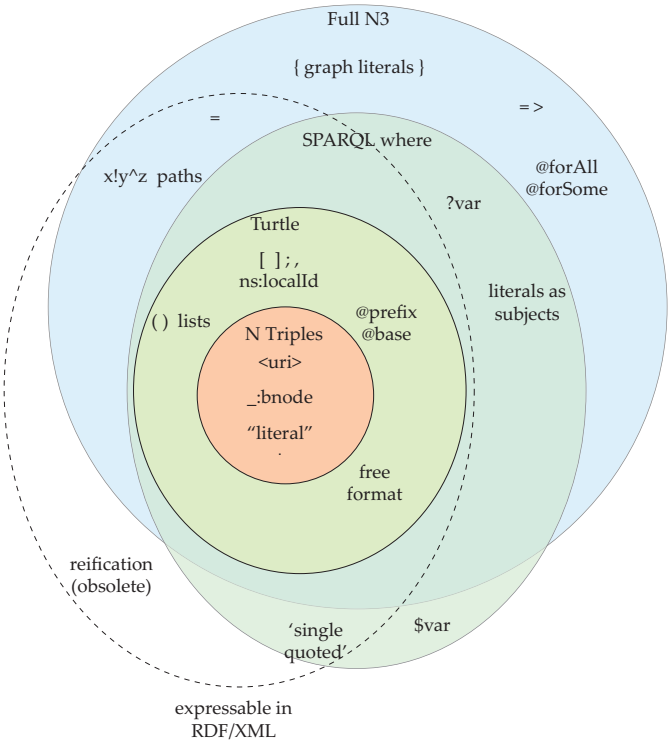
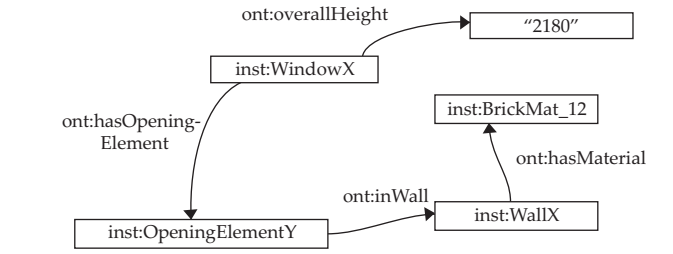


Figure 3.4: Venn diagram summarizing the expressiveness of the different syntaxes for RDF graphs (original image from Berners-Lee [2005]). The diagram shows that the N-Triples syntax is the least expressive, followed by Turtle and SPARQL. Full N3 is one of the most expressive syntaxes for describing information.




```
@prefix  inst:  <http://smartlab.elis.ugent.be/aimontologies/inst#> .
@prefix  ont:  <http://smartlab.elis.ugent.be/aimontologies/ont#> .

inst:WindowX  ont:hasOpeningElement  inst:OpeningElementY .
inst:WindowX  ont:overallHeight  "2180.00"^^xsd:double .
inst:WindowX  ont:inWall  inst:WallX .
inst:WallX  ont:hasMaterial  inst:BrickMat_12
```

Figure 3.5: An RDF graph representing a window (`inst:WindowX`), both in a graphical (top) and a textual N3 representation (bottom). The window has in this example two properties, namely the overall height of the window (`ont:overallHeight`) and the opening element in which the window is placed (`inst:OpeningElementY`). The graph additionally describes how the opening element is made in a wall (`inst:WallX`) that has a specific brick material (`inst:BrickMat_12`).

`hasOpeningElement` and `ont:overallHeight` label the logical relations attached to the `inst:WindowX` concept.

Information represented in this RDF graph can be queried with the Simple Protocol and RDF Query Language (SPARQL) [Prud'hommeaux and Seaborne, 2008]. SPARQL queries are similar to simple SQL queries. SPARQL queries, however, additionally benefit from the deployed triple structure, both in the RDF graph and in the SPARQL queries. This, namely, allows intuitively constructing semantically more complex queries.

3.1.2 Ontologies

RDF graphs can be given an improved semantic structure using RDF vocabularies or ontologies. The most basic elements to describe such ontologies are available in the RDF Schema (RDFS) vocabulary [Brickley and Guha, 2004]. RDFS enables the specification of classes, subclasses, data types, and so forth. This specification is done through RDF statements that deploy concepts from the RDFS vocabulary. Examples can be found in Fig. 3.6, specifying for instance that the `ifc:IfcWindow` class is a subclass of the `ifc:IfcBuildingElement` class.

More expressive elements to describe ontologies are available within the Web Ontology Language (OWL) [McGuinness and van Harmelen, 2009], which uses RDFS concepts as a subset. The RDF graphs constructed with OWL concepts are called OWL ontologies, and they can be used as an available vocabulary when making other, more complex RDF statements. Such an OWL ontology can be compared to the IFC schema in EXPRESS [International Organization for Standardization, 2004, Liebich et al., 2012, Scheck and Wilson, 1994], only using OWL/RDF as a language instead of EXPRESS. An OWL ontology may include representations of classes, properties and their instances. RDF resources can then 'belong to a specific class' of an OWL ontology (Fig. 3.6). By extending RDF with this OWL functionality, virtually any kind of information can be represented.

When using OWL and RDFS concepts in an RDF graph, one can infer new information through a standard reasoning process. For instance, when a resource is part of a subclass, a reasoning engine can automatically infer that the resource is also part of the superclass. Some of the concepts in OWL enable the specification of more elaborate rules. The RDF graph in Fig. 3.7, for instance, describes that for a concept to be an instance of the `ont:ResidentialBuilding` class, it always requires some values from the `ont:Bedroom` class for the property `ont:hasBedroom`. This RDF graph thus represents the rule '`IF` something has at least one bedroom, `THEN` it is a residential building'.

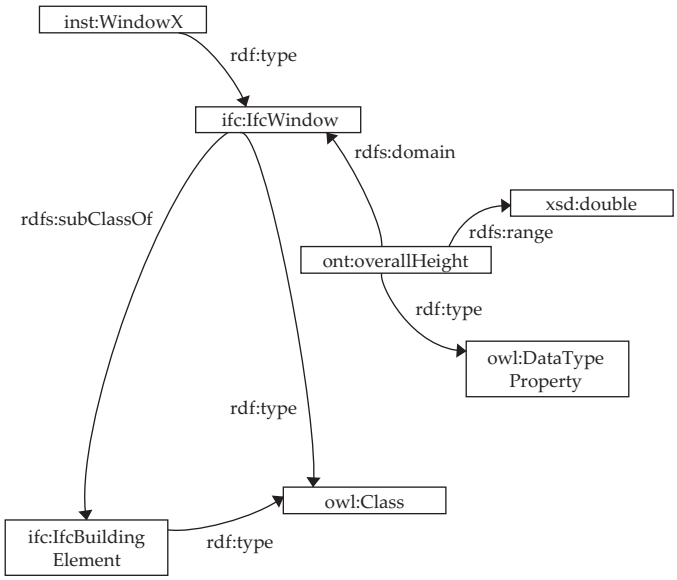
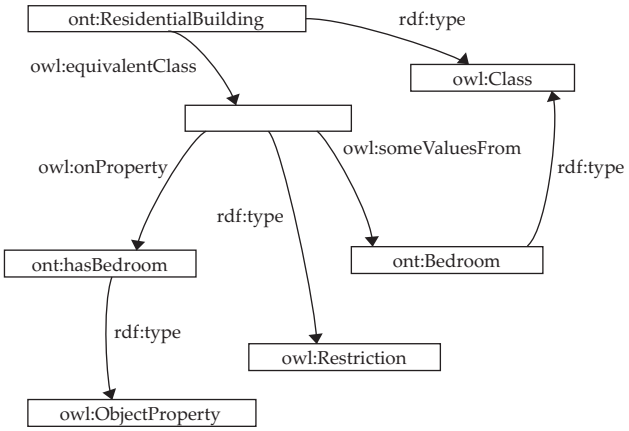


Figure 3.6: An OWL ontology describing an `ifc:IfcWindow` class. The `ifc:IfcWindow` class is here described as a kind of building element (`ifc:IfcBuildingElement`). The `ifc:IfcWindow` class can have a `ont:overallHeight` property (see Fig. 3.5), which should be expressed by a double precision floating point number (`double`).

3.1.3 Rule languages

The available RDFS and OWL concepts enable only a specific level of reasoning. In many cases, the combination of such concepts with a more dedicated rule language is desirable for the description of more complex rules. Using a specific rule language, one is able to define custom rules and subsequently use them in a rule-based reasoning process. Several rule languages have been developed to express such rules. Three of the most notable initiatives in the semantic web domain are the Semantic Web Rule Language (SWRL) [Horrocks et al., 2004], the Rule Interchange Format (RIF) [Rule In-




```
ont:ResidentialBuilding  rdf:type  owl:Class .
ont:ResidentialBuilding  owl:equivalentClass  _:125 .
_:125  rdf:type  owl:Restriction .
_:125  owl:onProperty  ont:hasBedroom .
_:125  owl:someValuesFrom  ont:Bedroom .
ont:Bedroom  rdf:type  owl:Class .
ont:hasBedroom  rdf:type  owl:ObjectProperty
```

Figure 3.7: A set of RDF sentences describing a simple rule using OWL concepts. The graph contains one 'blank node', identified by the node identifier `_:125`. This node identifier is only used in the serialization into the N3 syntax. This identifier must not be used as a URI and consequently does not appear as a named label in the RDF graph [Berners-Lee and Connolly, 2011].

terchange Format (RIF) Working Group, 2012b] and N3Logic [Berners-Lee et al., 2008].

SWRL was proposed as one of the first semantic rule languages in Horrocks et al. [2004]. It is based on a combination of OWL [McGuinness and van Harmelen, 2009] and RuleML [The Rule Markup Initiative, 2012]. Both an abstract and an XML concrete syntax were proposed to describe rules. A similar approach was picked up by the RIF working group while constructing RIF [Rule Interchange Format (RIF) Working Group, 2012b]. RIF is similar to the SWRL effort in the sense that it allows rules to be expressed both in XML as a normative concrete syntax and in a more human-readable abstract syntax. N3Logic is an alternative approach to SWRL and RIF, proposed by Berners-Lee et al. [2008]. Unlike SWRL and RIF, this third rule language is based on N3 as its normative syntax, which was presented as a readable alternative for RDF/XML by Berners-Lee and Connolly [2011].

These rule languages aim to do for logical information what RDF does for data: to provide a common data model to express globally sharable information on logic. They enable information with a far greater expressive power, such as rules, to be shareable similar to the way in which RDF is sharable. Having one rule represented in such a rule language, any other agent should be able to process this rule and possibly apply it within a completely different situation and environment.

Examples will be documented here in N3Logic. A rule in N3Logic contains hypothetical 'formulae'. Every hypothetical formula thereby uses curly brackets in its syntax to describe a subgraph, i.e. a logical conjunction of the statements between the curly brackets [Berners-Lee et al., 2008] (Fig. 3.8). Following the syntax of an RDF triple statement, a rule may describe that `IF` one hypothetical formula is true, `THEN` the other hypothetical formula is also true. In addition, an N3Logic rule references the concepts required in the rule using the URIs of these concepts, optimally abbreviated with prefixes.

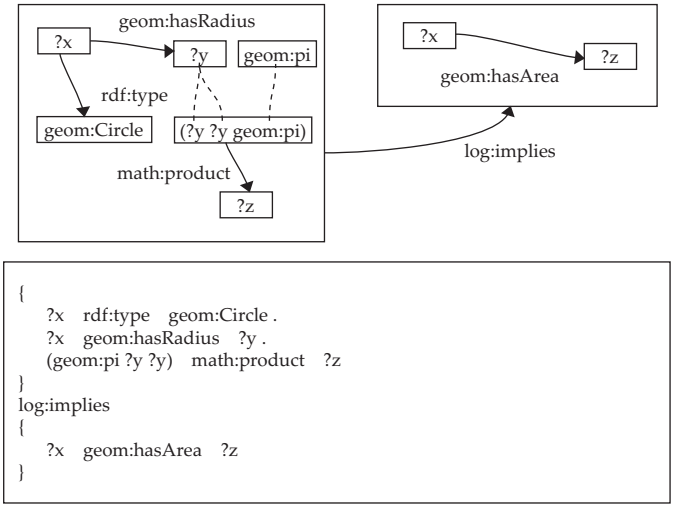


Figure 3.8: N3Logic rule in its normative N3 notation. The rule in this case described how one can infer the area of the circle (`geom:hasArea`) using the radius of the circle (`geom:hasRadius`), π (`geom:pi`), and a multiplication (`math:product`).

An example rule in N3Logic is given in Fig. 3.8. This rule specifies that `IF` an entity can be found that is of type `geom:Circle`, `AND` that has a `geom:hasRadius` property that may be reused in a calculation $\pi \times \rho^2$, `THEN` that entity has a `geom:hasArea` property pointing to the result of this calculation (`?z`).

3.1.4 Reasoning engines

Using rules and RDF graphs, a semantic reasoning engine may infer information that was not explicitly available before. These reasoning engines typically return a conclusion consisting of a set of newly inferred facts or a proof that a statement holds or not. An elaborate comparison of existing reasoning engines is out of scope here. Instead, only a brief description is given for two of the most significant reasoning engines. Additional information can be found elsewhere [Berners-Lee, 2009, Berners-Lee et al., 2003, De Roo, 2012a,b, Rule Interchange Format (RIF) Working Group, 2012a].

The Closed World Machine (CWM) reasoning engine was the first reasoning engine built for processing N3Logic and RDF graphs. It is developed as a *"general-purpose data processor for the semantic web"* [Berners-Lee, 2009] and is capable of loading resources, applying rules and outputting results in one of the several syntaxes available for RDF. CWM is a typical forward-chaining reasoning engine, meaning that it starts from the available facts and uses the available inference rules to infer new facts, after which these facts are added to the initial facts base and the reasoning process starts all over until there is nothing more to infer. The benefit of using a forward-chaining reasoning engine is that a lot of new information is generated, which could be useful for certain automation purposes. On the other hand, in many cases far too much unnecessary information is inferred, forcing even the smallest queries to go through an unnecessary long inference process.

A backward-chaining reasoning engine, on the other hand, starts from a set of SPARQL(-like) queries. It searches for the concepts contained in the given query both among the supplied RDF graphs and among the supplied inference rules. Only those rules that contain the required concepts in their `THEN` clause are taken into account [Berners-Lee et al., 2008]. If the `IF` clause of a needed rule is not known to be true, it is added to the overall query, and the search process continues. Reasoning is consequently limited to a minimum and occurs on-demand. The Euler Yap Engine (EYE) is one of the most notable reasoning engines that is based on N3Logic and RDF graphs and implements a backward-forward-backward chaining reasoning process [De Roo, 2012a,b]. This reasoning engine relies on Yet Another Prolog (YAP) [Costa et al., 2011], which is a Prolog [Colmerauer, 1993] system that has been under constant development since the mid-eighties.

Reasoning engines similar to CWM and EYE exist and are under development for SWRL and RIF [Rule Interchange Format (RIF) Working Group, 2012a]. When planning the design and implementation of an effective and widely used reasoning environment, one should take into account differences between these reasoning engines concerning efficiency, resource allo-

cation, and so forth. Test cases documented in this chapter typically rely on the reasoning functionality provided by EYE, mainly because of its backward chaining capabilities.

3.2 A framework that combines information models

As a part of research at UGent SmartLab [Pauwels et al., 2009b, 2011a], an architectural information modeling (AIM) framework was suggested with the aim of combining diverse information models within the AEC domain and providing a diverse set of applications on top of these information models (Fig. 3.9). As such, this framework provides a good context for testing the proposed linked data approach based on semantic web technologies (section 2.3.6). In this framework, architectural designers have the ability to describe all information that is part of an architectural design process into one semantic web of architectural information (lower layer in Fig 3.9). This web of architectural information provides information that can be used in various contexts for various purposes (upper layer in Fig. 3.9).

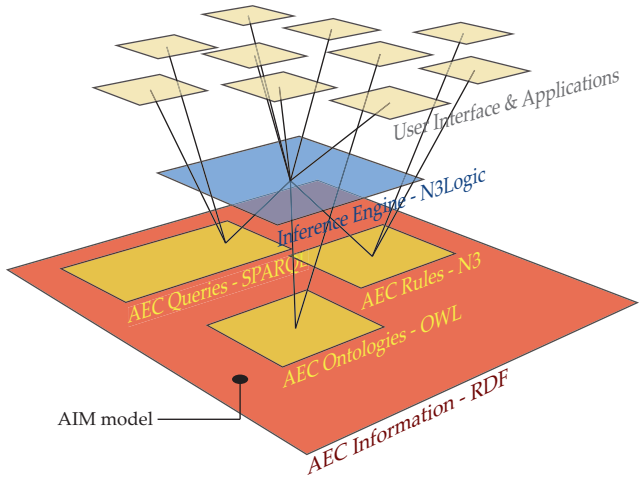


Figure 3.9: Diverse layers of additional functionality can be built on the common layer of information (see Fig. 2.18) that includes an AIM model. As such, the same source of information can be used in various contexts and applications.

Central in the web of information is one specific information model, namely the AIM model that is being modeled by the designer. This model forms a distinct semantic domain, namely, the representation of concepts, objects and relations used by a designer for a specific design situation. In terms of the example in section 2.4, this AIM model contains a representation of the historical references and architectural theories that are used by the designer to describe the design. Because this AIM model is modeled as a part of the global web of linked data (section 2.3.6), designers theoretically have the ability to model their interpretation of a design situation in a separate semantic domain, and link this semantic domain to the various other semantic domains available in the web of linked data.

In the context of the AIM framework, three initial ICT components were distinguished that could be developed on top of the central web of architectural information with the AIM model (Fig 3.9): an architectural memory component (AM), a virtual simulation and calculation component (VSC), and a Virtual Reality - Visual Simulation component (VRVS) (Fig. 3.10). These components and the central AIM model more or less align with the four application types outlined in section 2.1: modeling applications (AIM), archive applications (AM), calculation applications (VSC), and visualization applications (VRVS).

3.2.1 Architectural Memory (AM)

The AM component should be understood as a global repository of general and specific knowledge concerning architecture and construction [Heylighen, 2007]. This AM component does not only include references to buildings visited, readings and stories about buildings, and remembered design and construction processes. It also contains references to cultural backgrounds, such as ideas about how spaces should be used, opinions on the effect of color to the feeling of a space, and so forth [Cross, 1982, Heylighen and Neuckermans, 2000]. In terms of our earlier discussion in chapter 2, this component adds diverse information models to the global web of information, thereby capturing various interpretations and experiences of all kinds of concepts and objects (Fig. 2.21). This component furthermore lets a designer link the central AIM model (the designer's interpretation) to these information models. This should eventually serve as the basis for improvements in the interoperability issue and perhaps also in the functionality mismatch issue.

Figure 3.11 shows to what this might lead for an example model of a casino when relying on semantic web technologies. An interpretation of a certain design element of this casino is represented as a specific concept in

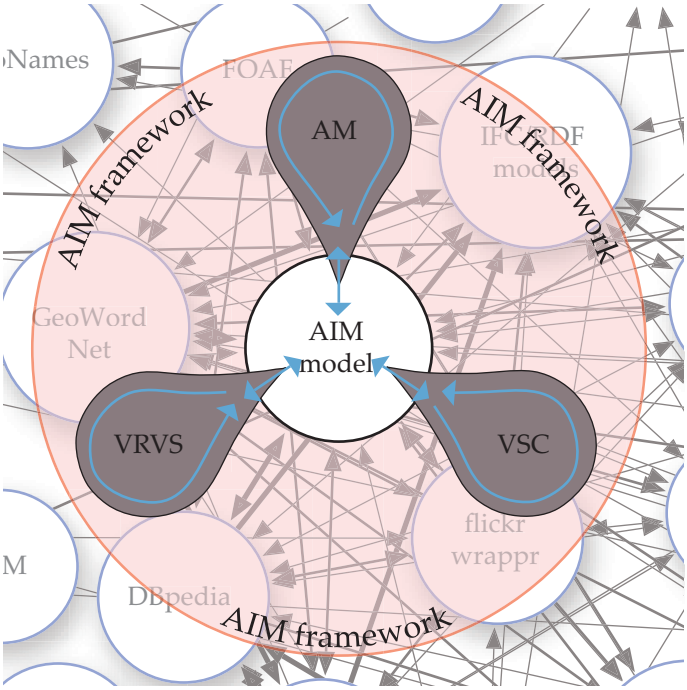


Figure 3.10: The AIM model lies at the center of the AIM framework. This model is linked to various other information models in the web of linked data (Fig. 3.2). The information in this web of data can be used by the AM, VRVS and VSC components through the AIM model.

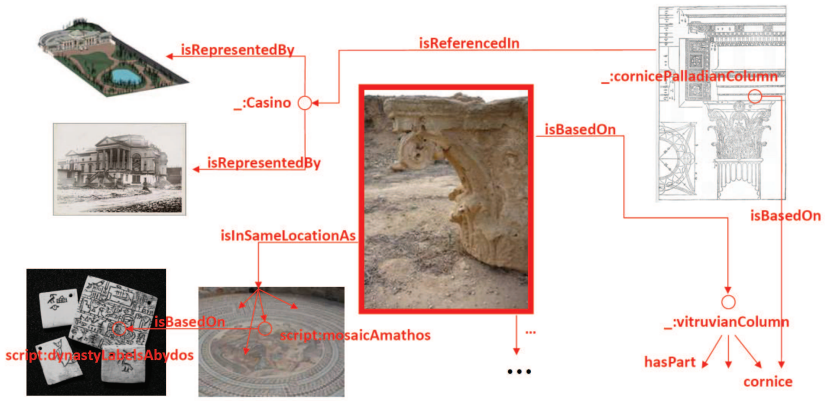


Figure 3.11: The AM component allows to retrieve relevant information in the web of data through concepts that are part of the central AIM model (`_:Casino`) [Pauwels et al., 2008, 2009b].

an RDF graph (`_:Casino` in Fig. 3.11). This concept is linked to available resources, such as a 3D model or a photograph, for instance. Furthermore, the RDF graph can be enriched with references to existing interpretations of other resources, such as the resources shown in Fig. 3.11.

Using SPARQL, the resulting RDF graph can then be queried to find, for example, similar features in other parts of the AIM model or in RDF graphs in the AM component [Pauwels et al., 2008, 2009b]. For a column capital, for instance, references can be found to objects that are located in the same area, or to objects that have similar geometric parameters or construction characteristics, and so forth (Fig. 3.11).

3.2.2 Virtual Simulation and Calculation (VSC)

The VSC component enables making simulations and calculations based on the information in the central web of information with the AIM model. Main improvements of the linked data approach should be the improved level of interoperability between calculation applications in this VSC component and available modeling applications using this web of information (Fig. 3.9). An additional improvement of using a linked data approach for this component is the possibility to use a lot more information than is typically available in calculation applications. Example calculations or simulations are the calculation of the load-bearing capacities of a column, the energy performance level of a space within a context of surrounding architecture and constructions, a calculation of the similarity level of diverse archi tectural elements, and so forth. A third possible enhancement generated by the usage of semantic web technologies, should be the possibility to deploy rules in a semantic rule language and a logic-based reasoning engine. Because both the information representation language (RDF) and the semantic rule language (N3Logic) rely on description logic (DL) [Baader and Nutt, 2003], a calculation application can presumably be implemented declaratively using DL principles [Pauwels et al., 2011c]. This enables a language-driven and more generic approach than is currently available in similar existing applications in the AEC domain [Eastman et al., 2009].

3.2.3 Visual Simulation - Virtual Reality (VRVS)

The VRVS component enables making architectural visualizations using the information that is available in the AIM model. Considering that a visualization component is also present in modeling, archive, and calculation applications (see section 2.1.4), one might to some extent use (parts of) this VRVS component to start implementing GUIs for the complete AIM framework, including the AM and the VSC components. In this thesis, however,

we will only look into the possibilities of using a linked data approach in the production of architectural visualizations. And also in this context, we will not look into diverse techniques for producing architectural visualizations, let alone of visualizations in general. Instead, we will mainly investigate to what extent the information that is required or desired by the designer can be made available in the visualization environment.

The usage of a linked data approach with semantic web technologies has certain advantages in the VRVS component. Because the web of information at the foundations of the AIM framework includes an AIM model that reflects the information structure with terms and concepts used and understood by an end user, architectural visualizations can be tailored to the needs of this end user. An additional advantage of using standard semantic web technologies, is that they allow reusing equally standard tools in combination with the information represented in the semantic web. Similar to the adoption of existing logic-based reasoning engines in the VSC component, the VRVS component may adopt a generic visualization technique developed to visualize RDF graphs. An example standard front end is the Pubby linked data interface [Cyganiak and Bizer, 2012] (Fig. 3.12). One can of course also develop custom in-house visualization tools. For the production of architectural visualizations in the AIM framework, it would be interesting to rely on existing technologies available for three-dimensional visualizations, including Virtual Reality (VR) and Mixed Reality (MR) technologies, for instance. This could eventually result in architectural virtual environments that can be explored by an end user and that present information in the information structure of this end user, which aligns with our initial purpose as documented in section 2.1.4 and 2.4.

3.3 Improving information exchange in the AEC domain

In this section, we investigate to what extent information that is deployed in diverse applications can be connected with respect for the inherent syntax and semantics of the information. Context of this investigation is the RDF graph in the lower level of the AIM framework (Fig. 3.9). By doing so, it is assessed to what extent the interoperability issue in the AEC domain (section 2.3) is addressed.

A semantic web approach was already suggested to improve the interoperability of CAD information by Abdul-Ghafour et al. [2008]. Using semantic web technologies, one should be able to represent 3D information, which is otherwise contained in a file according to a specific infor-

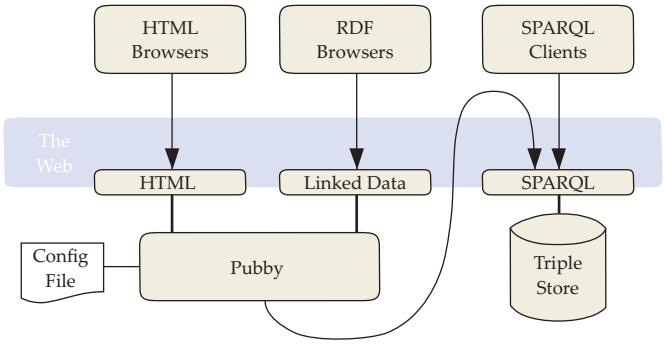


Figure 3.12: "Many triple stores and other SPARQL endpoints can be accessed only by SPARQL client applications that use the SPARQL protocol. It cannot be accessed by the growing variety of Linked Data clients. Pubby [Cyganiak and Bizer, 2012] is designed to provide a Linked Data interface to those RDF data sources." (original image from Cyganiak and Bizer [2012]).

mation structure, in a directed labeled graph, thereby describing its syntax and semantics in basic logical terms. This graph representation allows the expression of relations between different information structures. Abdul-Ghafour et al. [2008] indicates how this allows the combination of information from several different knowledge domains, enabling a seamless coupling of 3D information to non-geometric information, such as design intent and domain-specific product features.

Whereas these researches give a general overview of how semantic web technologies might form an appropriate alternative track to enable the interoperability of information between different application domains in the AEC field, they do not really handle the combination of alternative representations for the *same* information. Many important interoperability issues are, however, typically caused by the presence of such alternative representations for the same information. It is argued in this section that, additionally, a way should be found to connect these diverse ways of describing the same information directly and consistently, a goal that may be achieved using semantic web technologies. The former goal, connecting representations of information stemming from very diverse application domains, will be dealt with in section 3.3.1. The latter goal, connecting representations of nearly the same information, will be dealt with in section 3.3.2. A more detailed discussion of these two sections can be found in Pauwels et al. [2010a, 2011b].

3.3.1 Combining representations of different information

In diverse case studies, it was investigated how the issue of interoperability could be improved using semantic web technologies [Pauwels et al., 2008, 2009b,c, 2010a, 2011b]. We will look into the case study that was documented in Pauwels et al. [2010a], in which information is modeled for a building in Antwerp, Belgium, that was designed by architects R. De Meyer and F. Van Hulle (Fig. 3.13). An RDF graph was created for this design context with information represented with the IFC ontology [Liebich et al., 2012] and with an AIM ontology [Pauwels et al., 2010a].

Figure 3.13: The building in Antwerp includes a store on the ground level, and three flats on the four upper floors. The location of the three flats has been subject to study (right) during the initial design process (image courtesy by R. De Meyer and F. Van Hulle).

The IFC ontology is used as an EXPRESS schema by BuildingSMART [BuildingSMART International, 2012, International Organization for Standardization, 2004, Liebich et al., 2012, Scheck and Wilson, 1994] for describing information that is typically used in construction industry, and is thus usable by corresponding applications in construction industry. In this research project, this IFC ontology in EXPRESS is partially converted into an OWL ontology, similar to how it was suggested by Beetz et al. [2009]. With this ontology, an online web service is built and maintained through which file-based IFC models can be uploaded for conversion into IFC/RDF graphs [UGent Multimedia Lab, 2012a]. A file-based IFC model can be obtained by modeling the design context (Fig. 3.13) in a BIM application, such as Revit Architecture 2011, and exporting this model to an IFC file. The IFC/RDF graphs resulting from our IFC-to-RDF conversion service [UGent Multimedia Lab, 2012a] are made available online through a SPARQL endpoint for query access [UGent Multimedia Lab, 2012b].

The AIM ontology is modeled as an OWL ontology and used by an architectural designer to model the design at hand in an AIM model. In this case, this AIM ontology enables the representation of architectural design concepts, such as style information, history information, design intent information, and so forth. This ontology is constructed within this research project and is solely meant for investigating the applicability of semantic web technologies within the AEC field and can by no means be considered final. Similar efforts in which designers are able to model their interpretation of a design situation, can be found in Abdul-Ghafour et al. [2007], Böhms et al. [2009a,b], Kraft and Nagl [2007]. Using the AIM ontology, one is able to express design entities in an RDF graph, thereby including distinct knowledge domains (Fig. 3.14).

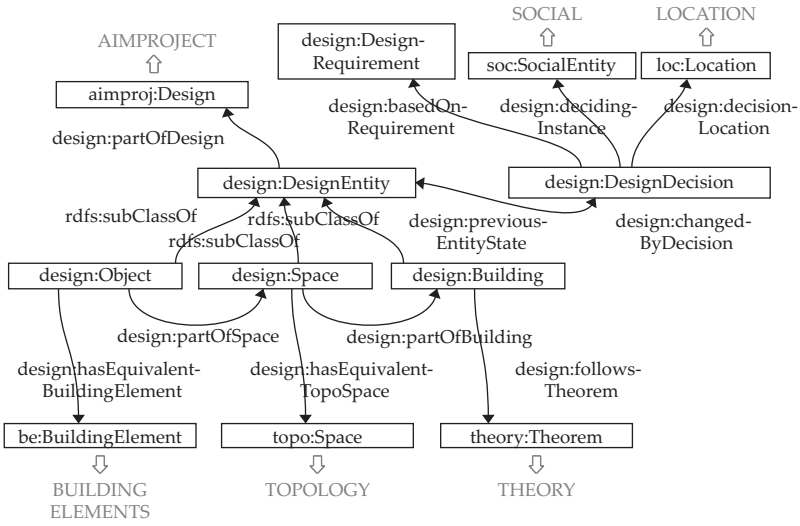


Figure 3.14: Graph overview of the design ontology that was used for this AIM model. This design ontology allows describing diverse design entities (`design:Object`, `design:Space`, and `design:Building`) and related design decisions. Concepts in this design ontology can be linked to other information (social, location, and so forth).

Using these ontologies, an example RDF graph was built for the considered design situation (Fig. 3.13). This RDF graph combines diverse representations, including an IFC/RDF graph, a design topology graph, and a design structure graph. As such, this example shows how information stemming from different application domains can be combined, with respect for the inherent syntax and semantics of each interpretation.

As an example, a small fraction of the design structure graph is given in Fig. 3.15. This graph represents a part of the steel structure of the design. Every `design:Object` instance (e.g. `inst:Beam_1`) is linked to its equivalent `be:BuildingElement`, thereby explicitly connecting design properties to construction type properties, IFC properties, geometric properties, and so forth. Figure 3.15 further illustrates where to include a representation of the geometric representation (`ifc:representation`) and the 3D placement (`ifc:objectPlacement`) of a building element, or how one may describe a truss consisting of two girders and two columns.

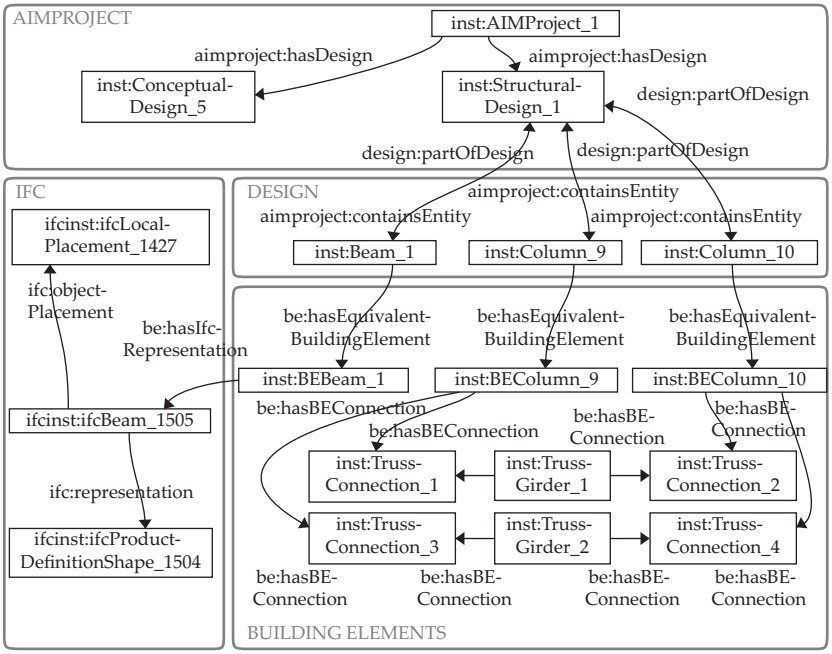


Figure 3.15: Part of an RDF graph illustrating how a steel construction may be described as an aggregation of columns and beams. In this graph, equivalent instances in diverse contexts (`inst:Beam_1`, `inst:BEBeam_1`, and `ifcinst:ifcBeam_1505`) are linked together, thereby explicitly connecting diverse properties of the same object.

Other information has been represented as described above, including geometric properties, theoretical information, and so forth. Information that is not considered a direct part of the AEC domain may be connected to this graph as well [Pauwels et al., 2011a]. This may include, for instance, geographical information (e.g. GeoNames [GeoNames, 2012]), people and organization information (e.g. FOAF [Brickley and Miller, 2010]) or expert material information (e.g. MATOWL [Zhang et al., 2009]).

Figure 3.15 indicates how one can represent the same concepts and objects according to diverse interpretations and link these representations so that this information may become appropriately reusable or interoperable. The same concept, namely, one of the beams in the design context, is described through multiple concepts, namely `inst:Beam_1`, `inst:BE-Beam_1`, and `ifcinst:IfcBeam_1505`. These concepts are linked in the RDF graph. As such, semantic web technologies allow to combine various information models. An application on top of this web of information may then rely on information stemming from one of these models as required. For the example in Fig. 3.15, a BIM environment may rely solely on the information represented by the IFC/RDF graph, whereas a simulation environment may rely on parts of the building elements graph.

3.3.2 Combining representations of the same information

In the example in the previous section, graphs were combined that represent information stemming from very diverse application domains. In many cases, however, information models in the AEC domain lie much closer together and tend to represent nearly the same information. This is most often illustrated in the context of 3D geometry, as was also indicated in Pauwels et al. [2011b]. The same geometry can be represented in many different ways corresponding to the context in which it is used. One information structure may represent a sphere, for instance, through its center and radius, whereas an other information structure represents it through a circular arc and a central axis, or maybe through a triangular mesh. As is shown in Fig. 3.16 for a simple box-shaped wall, this information can be combined into one semantic web. Changing part of this information, however, may lead to inconsistencies in the combined representation or web of information. For the example in Fig. 3.16, a change in the x-coordinate value of one of the vertices in the box-shaped wall (`inst:XCo1`) is not easily translated back into a change in the sizes of the box-shaped wall recorded with the X3D information structure (`x3d:sizeX`, `x3d:sizeY`, and `x3d:sizeZ`).

This issue may be addressed by relying on rules and a reasoning engine to infer the duplicate information on-demand [Pauwels et al., 2011b]. In this case, one geometric representation is available in an RDF graph and representations following a different schema are generated on demand by the reasoning engine. One may, for instance, consider how an application stores geometry in an IFC/RDF graph. When combined with the appropriate rule set, namely one that enables the inference of the same geometric information following the STL ontology, one automatically obtains also this

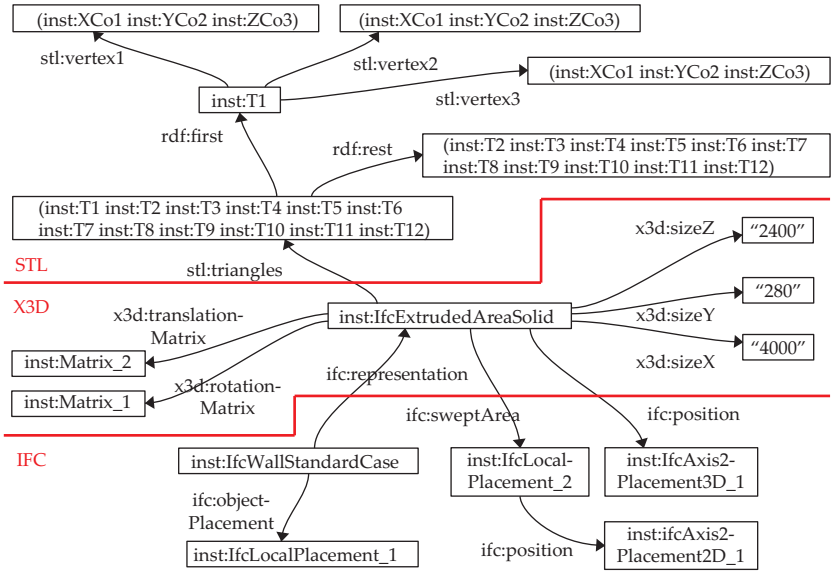


Figure 3.16: Part of an RDF graph illustrating how the geometry of a box-shaped wall can be described according to the IFC information structure (extruded area solid), the X3D information structure (rotation and translation matrix) and the STL information structure (mesh).

second representation. An application requiring the input of geometry in the STL schema can rely on this inferred STL/RDF graph. Changes made to the geometry in this STL-based application can be stored in a separate RDF graph. A second set of STL-to-IFC conversion rules provides interoperability of information back to the original IFC-based application. This approach could similarly be applied to other, non-geometric information. A brief overview is given here of how this was tested for converting IFC information into X3D and STL information (Fig. 3.17). More elaborate details about this rule-based conversion approach can be found in appendix A and in Pauwels et al. [2011b].

IFC to IFC/RDF

The IFC to STL conversion test is started from a simple 3D model created in a standard BIM environment, namely Autodesk Revit Architecture 2010. This model only contains four connected wall objects with a box-like shape. From the BIM environment, an IFC description is exported using the standard available export functionality provided by Autodesk. The resulting

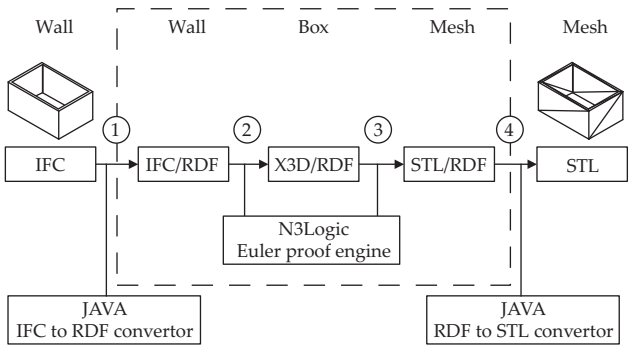


Figure 3.17: Overview of the process followed for converting IFC information into X3D and STL information.

IFC description describes the model as a combination of four `IfcWallStandardCase` concepts with both purely geometric (such as position, representation, and so forth) and extra product information (such as material, cost, and so forth) attached. This IFC description is available in an IFC file and follows the EXPRESS schema of IFC. Figure 3.18 shows an extract of this IFC representation.

```
#113=IFCLOCALPLACEMENT(#38,#112);
#114=IFCCARTESIANPOINT((7000.,0.));
#115=IFCPOLYLINE((#4,#114));
#116=IFCSHAPEREPRESENTATION(#27,'Axis','Curve2D',(#115));
#117=IFCCARTESIANPOINT((3500.,-0.));
#118=IFCAXIS2PLACEMENT2D(#117,#12);
#119=IFCRECTANGLEPROFILEDEF(.AREA.,$,#118,6999.999999999997,199.
9999999999978);
#120=IFCAXIS2PLACEMENT3D(#3,$,$);
#121=IFCEXTRUDEDAREASOLID(#119,#120,#9,8000.000000000001);
#122=IFCPRESENTATIONSTYLEASSIGNMENT((#57));
#123=IFCSTYLEDITEM(#121,(#122),$);
#124=IFCSHAPEREPRESENTATION(#27,'Body','SweptSolid',(#121));
#125=IFCPRODUCTDEFINITIONSHAPE($,$,(#116,#124));
#126=IFCWALLSTANDARDCASE('1u0N27BfbCag9WPM5IHNzv',#33,'Basi
c Wall:Generic - 200mm:113334',$,'Basic Wall:Generic -
200mm:398',#113,#125,'113334');
```

Figure 3.18: Extract of IFC information formatted according to the EXPRESS schema of IFC.

To use semantic web technologies, the IFC information in EXPRESS first needs to be converted into an RDF graph. This conversion procedure is done using the online IFC-to-RDF web service mentioned earlier [UGent Multimedia Lab, 2012a]. Using this web service, the IFC file from

Revit was converted into an IFC/RDF graph and made accessible through a SPARQL endpoint [UGent Multimedia Lab, 2012b]. The conversion by the web service results in an RDF graph that describes the IFC concepts shown in Fig. 3.18 in a graph structure, as shown in Fig. 3.19 for one of the resulting `inst:IfcWallStandardCase` concepts. The geometry of the `IfcWallStandardCase` concepts is described in the resulting IFC/RDF graph through `IfcExtrudedAreaSolid` concepts that are each in turn described by an extruded direction, a depth and a rectangular profile defining the base profile or `swept area` (Fig. 3.19). The positions of the wall objects are described by a series of rotations and translations, all relative to each other, following the overall structure displayed in Fig. 3.20.

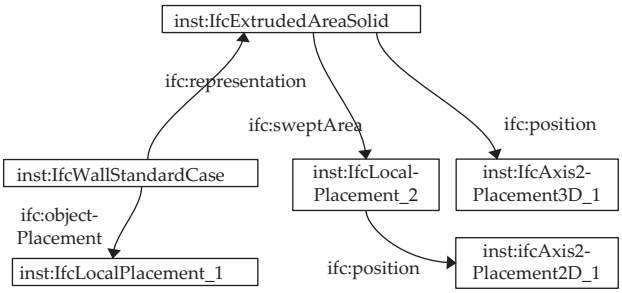


Figure 3.19: Part of the RDF graph for the `inst:IfcWallStandardCase` concept that results from the IFC-to-RDF web service.

IFC/RDF to X3D/RDF

The resulting IFC/RDF graph may then be converted into an X3D/RDF graph through a set of N3Logic rules. Starting from the IFC/RDF description, the most appropriate concepts are selected in the X3D specifications to rephrase the IFC/RDF information with. The schema of X3D does not include any notion of a wall object, let alone its product data. This information therefore needs to be disregarded in the conversion process. As 3D primitives were found to be part of the X3D schema, the descriptions of the wall objects are converted into the corresponding descriptions of X3D boxes and linked to the original IFC/RDF descriptions. A rule describing part of this conversion process is displayed in Fig. 3.21. Note that for other IFC concepts, such as doors, windows, floors, railings, and so forth, a similar process can be followed and rules can be obtained for the conversion of these IFC objects to the appropriately corresponding X3D objects. An

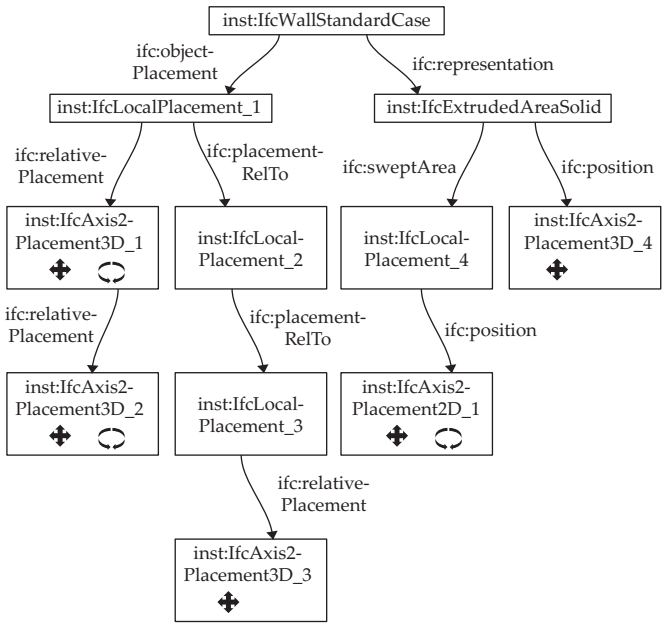


Figure 3.20: Overview of how the location of an `IfcWallStandardCase` *is described in the IFC/RDF graph as a series of relative rotations and translations.*

analogous conversion process can presumably be followed for the further conversion into an STL description.

The X3D schema describes the position of objects differently from how it is described in IFC. Instead of defining the position of a point through a whole set of relative transformations of both the position and orientation of the object (Fig. 3.20), X3D describes the location of an object through a combined translation matrix and rotation matrix relative to the world coordinate system used. These translation and rotation matrices can be inferred from the sequence of relative transformations of both the position and orientation as described in IFC. For each relative transformation, a rotation and translation matrix is calculated iteratively by the reasoning engine using specific N3Logic rules, after which another rule set in N3Logic performs the required matrix calculation on these matrices [Van Ackere and De Roo, 2010]. The eventually resulting translation and rotation matrices are added to the X3D/RDF graph.

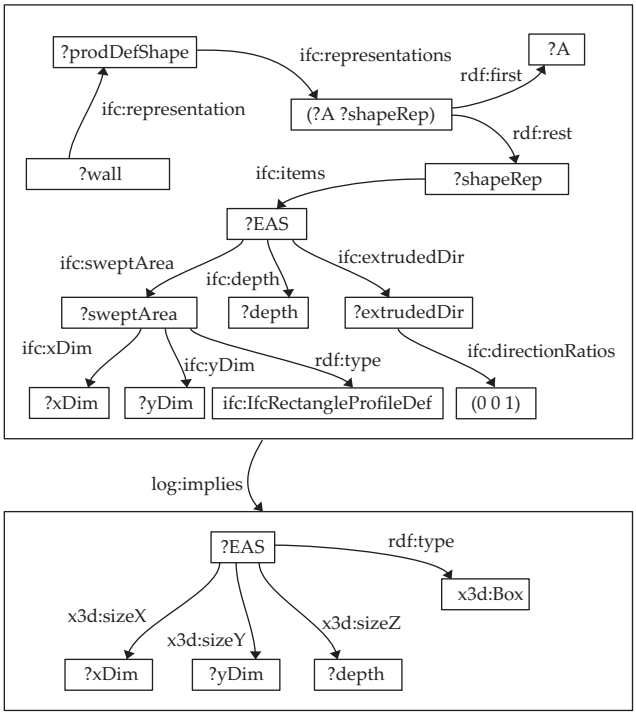


Figure 3.21: N3Logic rules creating a box description according to the X3D specifications with its additional dimensions.

X3D/RDF to STL/RDF

A third step comprises the conversion of the X3D/RDF graph into an STL/RDF graph. A similar method is followed as in the previous conversion step, starting with a search over the X3D and STL specifications for the appropriate STL concepts to be used for representing the available X3D concepts. Because the STL schema allows the description of geometry only through its triangles, the X3D/RDF box description needs to be converted into a 3D mesh description in STL/RDF.

This mesh description refers to a set of triangles, each in turn described by its normal and vertices. This is yet a different description compared to the description of a primitive box in the X3D schema. The conversion rules go through a whole range of matrix calculations to obtain these coordinates and convert these in a correct description of all the triangles in the mesh. One of these rules is given for reference in Fig. 3.22.

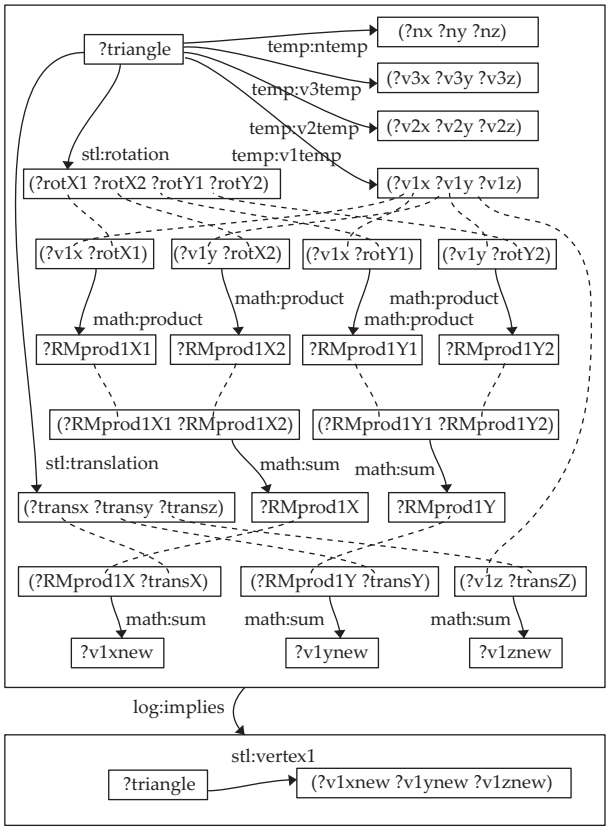


Figure 3.22: N3Logic rules describing the recalculation of the normal vector coordinates and of the vertex coordinates of one triangle in the STL mesh.

STL/RDF to STL

In a final phase, the STL/RDF graph is converted into an STL file, using a JAVA rewriting application. Because the STL/RDF ontology follows the syntax and semantics of the schema of STL, an easy one-to-one mapping can be followed. Importing the resulting STL file in a 3D modeling application, such as Rhinoceros 3D, indicated a correct conversion of the 3D information. Tests with several other simple models indicated an equally successful process, as long as the concepts with which they were described are incorporated in the rule set.

What is important here, is the graph that can be obtained simply by the combination of the initial graph in IFC (Fig. 3.19) and the N3Logic rules.

Both the X3D and the STL graph can be created on-demand, simply by running the reasoning engine and querying for the required information in the required format. This implies that no explicit and static links have to be set up between different information models, as is the case in Fig. 3.16. In a sense, the alternative representations are created on-demand and can be linked to the original, as indicated in Fig 3.23.

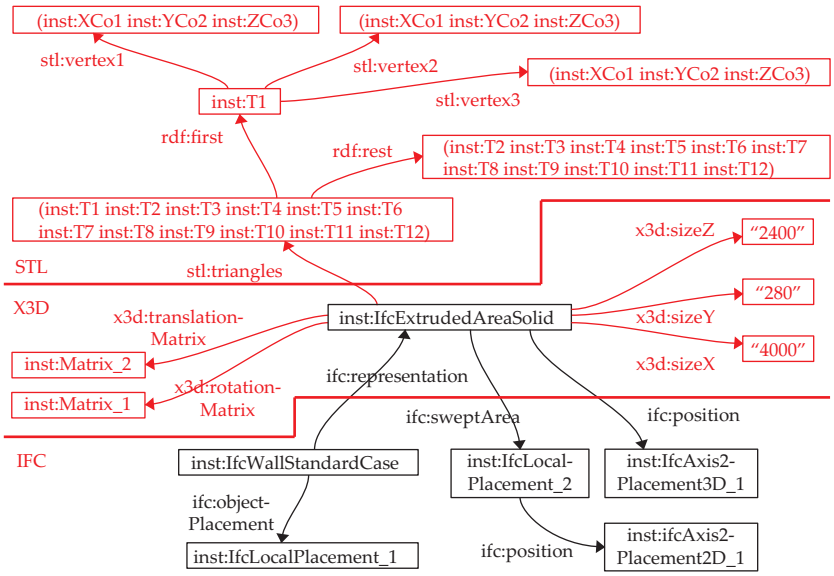


Figure 3.23: Starting from the RDF graph as shown in Fig. 3.19, the indicated alternative representations of the same information can be generated on-demand.

3.4 Applications on top of the web of AEC information

Section 3.3 indicates how an improved level of interoperability can be obtained for information models used in the AEC domain when relying on semantic web technologies. This not only includes information models used by applications, such as IFC models, but also information models that represent the semantic domain or partial understanding of a designer for a design situation (for example, AIM models). These information models can thus be combined into one global web of information, whether this be achieved by explicit describing and linking information (section 3.3.1) or by using rules that enable an on-demand conversion of information mod-

els (section 3.3.2). In this section, we will look at the diverse applications that can be implemented on top of this web of information (Fig 3.24), in order to find out to what extent the functionality mismatch issue outlined in section 2.4 is addressed with this linked data approach.

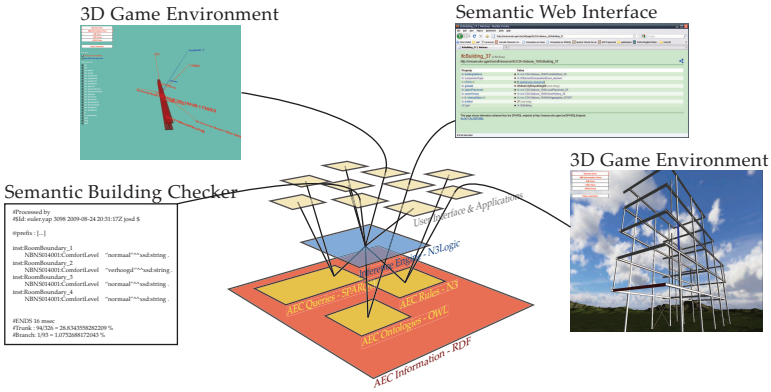


Figure 3.24: Applications on top of a central web of information.

3.4.1 Modeling applications

Modeling applications allow to model information concerning a design into information models. These modeling applications hereby rely on the available ontologies, such as an IFC ontology, a Revit Architecture 2011 ontology, an STL ontology, and so forth. Ideally, however, designers model their designs using their own concepts and interpretations, relying on their own AIM ontology, and link the resulting information models to other information models that follow these other ontologies. In other words, the main targeted result of solving the functionality mismatch issue for modeling applications, is that designers are able to represent a design in the application using their own structure (AIM ontology), with the modeling application able to use this representation to some extent.

The feasibility of this approach was tested in this research project for the Book Tower[1] in Ghent, Belgium. More information about this research project can be found in appendix B. In this case study, information concerning this building was modeled in Revit Architecture 2011 (Fig. 3.25), thus choosing to model the design of the building in the ontology used by the Revit application.

[1]The term 'Book Tower' is the literal translation of 'Boekentoren' (Dutch), and refers to the library tower of Ghent University.

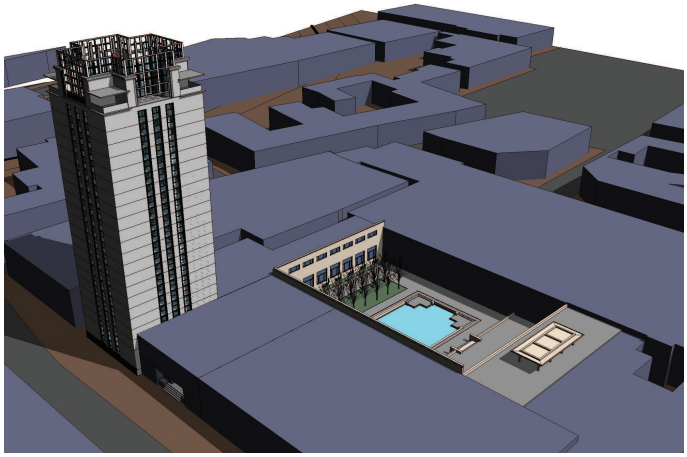


Figure 3.25: The Book Tower in Ghent, Belgium, modeled in Revit Architecture 2011.

This case study showed that the above 'ideal' situation is not yet realized. Namely, when modeling the building in Revit, the designer is confined to the ontology used by Revit. For instance, Fig. 3.26 gives an indication of how the outer walls of the Book Tower were modeled, which was not completely identical to how it was built in the 1930's. The inner side of the wall (`wall 1`) was modeled separate from the outer side of the wall (`wall 2`), because the two walls were also constructed as separate walls in reality, with the outer wall cast in concrete together with the column structure, and the inner wall as a later addition. However, since the Revit ontology allows windows to be only in one wall at the same time, a separate wall was modeled just for this window (`wall 3`). This is not the best representation for this construction, but it aligns with the Revit ontology.

By using semantic web technologies, on the other hand, one is able to build an AIM ontology specifically for this building at this specific time and place. Through a modeling application that takes into account this ontology, one is able to instantiate the diverse classes and relations in this ontology, which results in an RDF graph for the building at hand. This was tested using the Topbraid Composer application [TopQuadrant, 2012] as a modeling application. The Topbraid Composer application allows modeling OWL ontologies and corresponding RDF graphs. It was tested to what extent the Book Tower could be described with terms and concepts from the Italian UNI standard [Italian Normative UNI, 1981, 1999] and to what extent the resulting semantic domain could be linked to the information model as it was available from Revit.

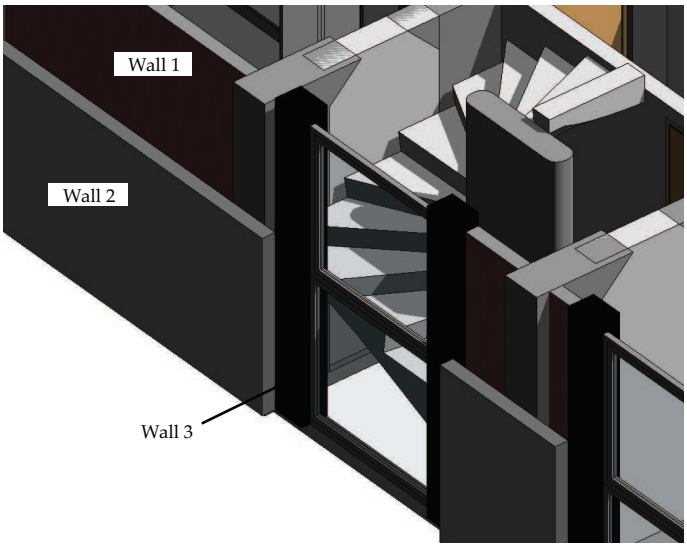


Figure 3.26: Part of the 3D model showing how the ontology of Revit in some sense requires modeling in a specific way.

The test started similarly to the test discussed in section 3.3.2: the building model in Revit was exported to IFC and this IFC model was converted into an IFC/RDF graph using the IFC-to-RDF web service [UGent Multimedia Lab, 2012a]. This IFC/RDF graph is available online via a SPARQL endpoint [UGent Multimedia Lab, 2012b]. Then, a distinct OWL ontology was constructed in Topbraid Composer representing the vocabulary and the structure that a specific user wants to use for modeling the building. In this case, the ontology represents the terms and structure provided by the Italian UNI standard for describing buildings [Italian Normative UNI, 1981, 1999]. Five additional ontologies were constructed in this test as well, each representing an alternative perspective on the building, including a `BuildingDesign` ontology, a `Documentation` ontology, a `BuildingDegradation` ontology, a `Material` ontology, and a `Location` ontology. An indication of some of the terms used in these ontologies is given in Fig. 3.27 .

Using these OWL ontologies, an RDF graph is modeled for a part of the Book Tower building using the GUI of TopBraid Composer. A small part of this graph is presented in Fig. 3.28. This simple graph indicates how one is able to describe a building using custom concepts and terms (`Story_14`, `VerticalElevation-Structure`, `Concrete_1`) and link this to a representation of relevant documentation (`Book_1`, `Drawing_2`).

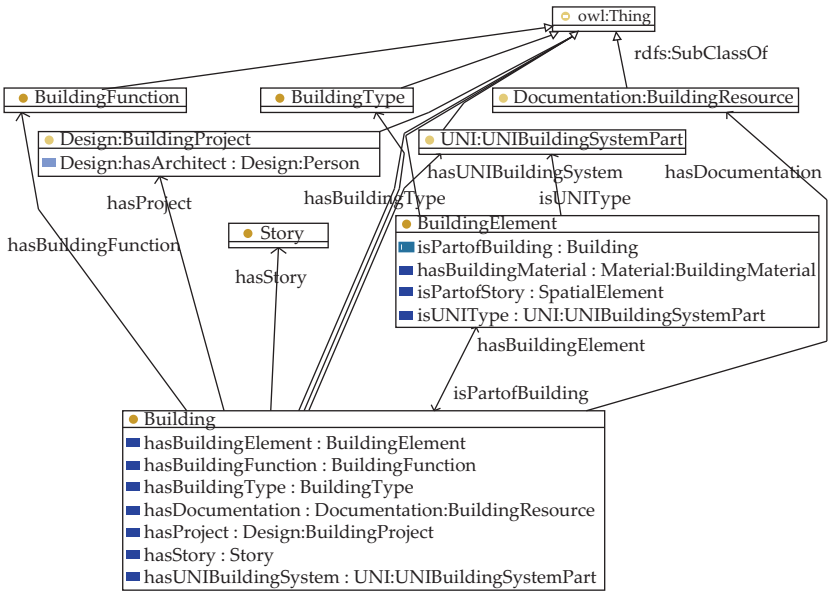


Figure 3.27: Some of the classes and properties used in the ontologies for the Book Tower project, as it is represented in Topbraid Composer.

Another RDF graph represents a description of the building in the terms of the UNI standard, and yet another RDF graph represents the building as it was modeled in Revit and exported in IFC. These and other RDF graphs are all statically linked together, following the approach documented in section 3.3.1. This eventually results in a network of information, of which an impression is given in Fig. 3.29.

This test case shows that the Topbraid Composer application allows to construct custom information structures as OWL ontologies. Once these ontologies are available, the interface of Topbraid Composer allows a user to model a design in the terms of that ontology. If this ontology reflects concepts and terms used by the designer, the information structure used by the modeling application thus conforms to the information structure of that designer. Note that the interface of Topbraid Composer is not an application that is typically used by architectural designers. An alternative GUI will thus have to be implemented on top of the functionality provided by an application as Topbraid Composer. As indicated in section 2.4.1, however, this topic is not considered in this thesis. Nevertheless, a first step is made in addressing the functionality mismatch issue by showing that the required information can be provided to an end user.

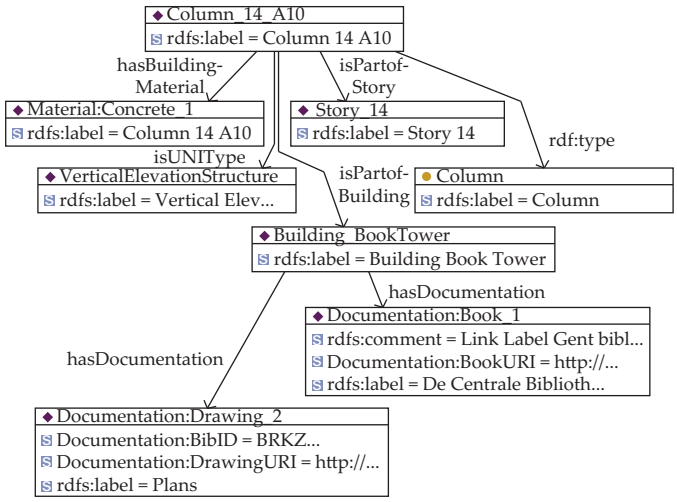


Figure 3.28: An RDF graph for the Book Tower project, as it is represented in Topbraid Composer, and following the ontologies of which a part is shown in Fig. 3.27.

A more important element that was encountered in the test case relates to the usability of the custom ontologies and the corresponding RDF graphs. Although users can model their custom information structures and corresponding information, other applications and other users are only able to directly use this information through the links to other ontologies and RDF graphs. This depends on the practical feasibility of making links, either directly or in a rule-based approach (section 3.3). This is possible, as can also be found in Fig. 3.16, for the more abstract or conceptual concepts. But it remains extremely difficult to directly combine more concrete and low level concepts, such as the diverse representations of a simple sphere. In terms of the Book Tower example, it is possible to state that a specific wall or column in one representation (Fig. 3.28) is identical to a specific wall or a group of walls in an other representation (IFC/RDF graph), but it is very hard to combine or to convert between the diverse characteristics of these walls.

3.4.2 Archive applications

The main aim of archive applications is to enable designers to easily find information that could in some way be useful for their design situation. Information that is typically stored in books, in the minds of people, or in

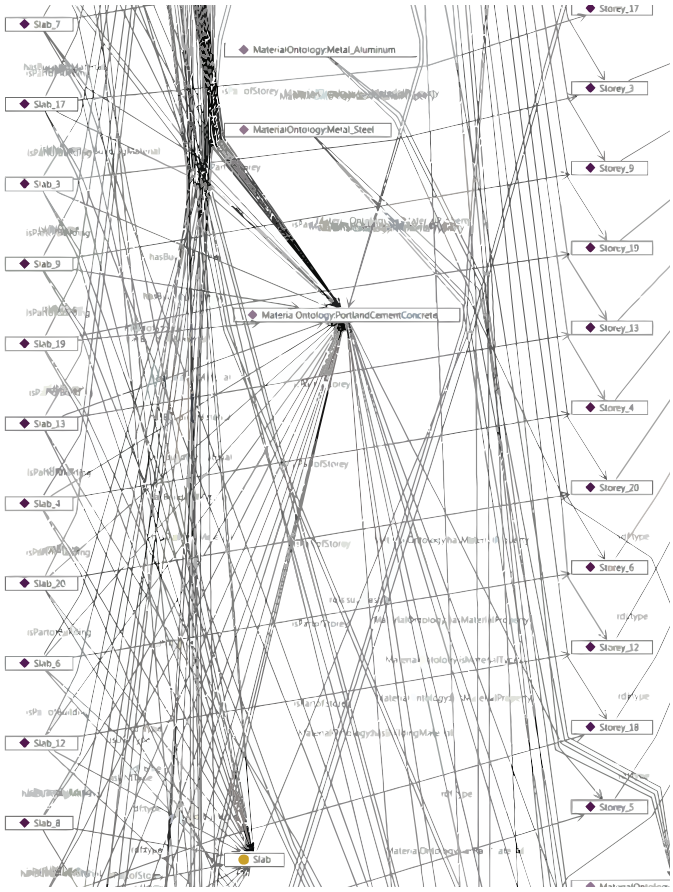


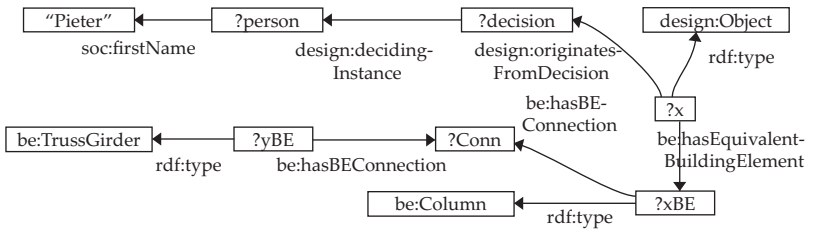
Figure 3.29: A small RDF graph was modeled with about 3000 RDF triple statements, after which relevant links were made with the IFC/RDF graph.

hard to find pictures and sketches, is brought together, digitized and made accessible via digital archive applications. The main targeted result of solving the functionality mismatch issue for archive applications, is that these applications are able to find and present information that is specifically useful for the design context and the architectural designer at hand.

In the AM component presented in section 3.2.1, we saw how a linked data approach enables to combine one's interpretation of a design in the AIM model directly to interpretations of diverse other architectural designs. As such, external information is directly available for SPARQL query access in the AM component (Fig. 3.11). This is nearly identical to the way in which information models are combined in a modeling application (sec-

tion 3.4.1). Only, in this case, information models are introduced for other architectural design situations as well.

The Finnish CultureSampo platform [Hyvönen et al., 2009a], which was discussed in section 2.4.2, is a recent example of such an archive application for information concerning Finnish cultural heritage. In this portal application, access is provided to diverse perspectives of Finnish cultural heritage, including information about specific artifacts, information about the people behind the cultural heritage, geographical information and timeline information. We present a similar example here, based on the AIM/RDF graph that was briefly discussed before for the design in Antwerp (Fig. 3.14 and 3.15). The information in this RDF graph is accessible through a SPARQL endpoint, such as the SPARQL endpoint on top of the IFC/RDF graphs [UGent Multimedia Lab, 2012b]. With the SPARQL query language [Prud'hommeaux and Seaborne, 2008], one is able to search via this SPARQL endpoint for very specific information in the RDF graph. Various queries can be processed, limited only by the diversity in the semantic structure of the queried RDF graph. The more detailed content is represented in the queried RDF graph, the more detailed one can search through this content. For instance, one may search for column designs that originated from a decision made by people with 'Pieter' as their first name, and that have a direct connection with a truss girder element (Fig. 3.30).




```
SELECT ?x WHERE {
      ?x rdf:type design:Object .
      ?x be:hasEquivalentBuildingElement ?xBE .
      ?xBE be:hasBEConnection ?Conn .
      ?xBE rdf:type be:Column .
      ?yBE be:hasBEConnection ?Conn .
      ?yBE rdf:type be:TrussGirder .
      ?x design:originatesFromDecision ?decision .
      ?decision design:decidingInstance ?person .
      ?person soc:firstName "Pieter" .
}
```

Figure 3.30: A SPARQL query for columns that are connected to a `be:TrussGirder` and that originated from a decision made by people with 'Pieter' as their first name.

Diverse user-friendly interfaces can be used in combination with this SPARQL endpoint, making the available information accessible in various ways. One could choose to use one of the existing semantic web browsers to browse and query the information, such as the DISCO browser [Bizer and Gauss, 2007]. One could choose to develop a custom web portal on top of the information, similar to how it is done in the MACE project [MACE Consortium, 2012, Stefaner et al., 2007], the OIKODOMOS project [ARC school of architecture La Salle, 2012], and the CultureSampo project [Hyvönen et al., 2009a].

The functionality that can be provided with this approach depends on the amount and the level of detail of the information in the RDF graph of the AIM model and the AM component. The archive application can only process SPARQL queries that contain concepts that are also included in this AIM model and the AM component. Consequently, the archive application is only able to respond to these queries. Concepts or suggestions that could be useful for the design context at hand, but that are not included in query responses, are not returned to the designer by the archive application. It is thus important to (1) include as much detailed information in the RDF graph as possible, and (2) to also use this information when making queries. Hence, solving the mismatch issue for archive applications is theoretically feasible, but only insofar as the AIM ontology and the SPARQL queries contain links to other ontologies and RDF graphs.

3.4.3 Calculation applications

Whereas modeling and archive applications typically aim at bringing information together, calculation applications typically aim at (re-)using this information. Numerous calculation applications exist in the AEC domain: thermal analysis applications, computational fluid dynamics (CFD) simulation environments, compliance-checking applications, structural analysis applications, and so forth. The main targeted result of solving the functionality mismatch issue for calculation applications, is that these applications are able to communicate their calculation results using the concepts deployed by the architectural designer and represented by the AIM model that is part of the RDF graph in the AIM framework. As an example, an application for acoustic simulations would supposedly not communicate a complete overview of all acoustic parameters and the corresponding analysis of a building to designers that are not familiar with this or simply do not need such a complete overview. Instead, this application might, for example, indicate optimal changes to the materials used in the structure, give only an `OKAY - NOT OKAY` indication, or might give any other spe-

cific advice. Before we look into the ways in which the AIM model can be taken into account to communicate results of a calculation procedure, we will first look into the ways in which this calculation procedure can be implemented using semantic web technologies.

Calculation applications typically provide a complex set of rules and algorithms on top of an information model (Fig. 3.9). In an acoustic simulation application, these rules represent the procedures that need to be followed for obtaining a complete acoustic simulation. This rule set needs to be closely related to the available information. Diverse strategies can be distinguished for realizing and optimizing this relation. Also when relying on semantic web technologies, diverse strategies can be considered. One strategy is to describe rules in the rule set using a query language (SPARQL), the other is to describe these rules using a rule language (N3Logic). Both strategies are examples of the 'language-based' approach in Eastman et al. [2009]. Each strategy has a specific way of describing the calculation rules and thus of accessing the corresponding information model.

Yurchyshyna et al. [2007] and Yurchyshyna and Zarli [2009] suggest an example implementation for the first strategy. In this strategy, building regulations are converted into `IF-THEN` rules. These rules are represented with the SPARQL query language, which has the ability to express singular `IF-THEN` functionality through its `CONSTRUCT` formalism [Prud'hommeaux and Seaborne, 2008]. Notwithstanding the significant value of this methodology and the significant improvements shown in the calculation process [Yurchyshyna et al., 2007], the implementation choices [Yurchyshyna and Zarli, 2009] appear to limit functionality in the eventual calculation application. Although SPARQL provides a very useful and intuitive `CONSTRUCT` formalism, it was still originally intended as a query language for RDF graphs [Prud'hommeaux and Seaborne, 2008] and not as a rule language. Apart from being restricted to singular `IF-THEN` functionality, SPARQL is mainly useful for a one-by-one querying or rule checking of a building model and does not allow an equally automated calculation process as it is provided by a combination of a dedicated rule language and a reasoning engine.

An example implementation for the second strategy can be found in the ConDes system, which was suggested by Kraft and Nagl [2007]. Although their ConDes system does not rely on semantic web technologies, it proposes a visual knowledge specification that can be considered equally expressive as an RDF graph specification. This visual knowledge specification is used for the specification of conceptual architectural design situations, using an add-on in the ArchiCAD 3D modeling environment of

Graphisoft. This specification can be combined with design rules that are to be formalized by one or more specialized knowledge engineers. By combining the specification and the design rules in ArchiCAD, similar to the methodology suggested in Yurchyshyna and Zarli [2009], an architect can model information from the very first conceptual design stage and check whether or not this design conforms to the applicable custom design rules. The ConDes system is a good example of what functionality a calculation application based on a rule-based language may provide to a specialist in construction industry.

An alternative implementation of the second strategy, based on semantic web technologies, was presented in Pauwels et al. [2011c]. More details about this implementation can be found in appendix C. The resulting system is similar to the ConDes system, but relying on semantic web technologies. By using such widely used technologies, a more solid basis is targeted for calculation applications. This strategy relies on an RDF graph and a set of rules in N3Logic. This strategy was tested in a case study for acoustic performance checking [Pauwels et al., 2011c]. One of the rules that was developed for the acoustic rule set was previously given in Fig. 3.8. A second, more elaborate example is given in Fig. 3.31.

```
{
    ?BE   rdf:type   EN12354:BoundaryElement .
    ?BE   EN12354:elementSurfaceArea   ?a .
    ?BE   EN12354:partOfRoomBoundary   ?RB .
    ?RB   rdf:type   EN12354:RoomBoundary .
    ?RB   EN12354:facadeSurfaceArea   ?atot .
    ?BE   EN12354:soundReductionIndex_4000Hz   ?R_4000 .

    ?a   math:notLessThan   1 .

    (?a   ?atot)   math:quotient   ?value_i .
    (?R_4000   -10)   math:quotient   ?value_j .
    (10   ?value_j)   math:exponentiation   ?value_k .
    (?value_i   ?value_k)   math:product   ?value_l
}
log:implies
{
    ?BE   EN12354:directSoundPowerRatio_4000Hz   ?value_l
}
```

Figure 3.31: Example rule in N3Logic from the EN12354-3:2000 rule set [European Committee for Standardization (CEN), 2000]. This rule specifies how the `EN12354:directSoundPowerRatio_4000Hz` *property can be derived from a building model. The* `EN12354` *prefix refers to the base URI of the rule ontology for this standard.*

The rules in Fig. 3.8 and 3.31 were combined with other rules into two rule sets for acoustic performance checking: a rule set for the European EN12354-3:2000 [European Committee for Standardization (CEN), 2000] and a rule set for the Belgian NBN S 01-400-1 [Bureau voor normalisatie (Commissie: Geluidsleer - algemeen), 2008]. An overview of the `IF-THEN` rules is given in Fig. 3.32. The EN12354-3:2000 rule set allows calculating acoustic performance values for the required building elements (T_f; $T_{e,i}$; R'; $D_{2m,nT}$) and the NBN S 01-400-1 rule set uses these values for calculating the performance level of building elements (`compliant to NBN` or `not compliant to NBN`) and of the façades (`ComfortLevel`).

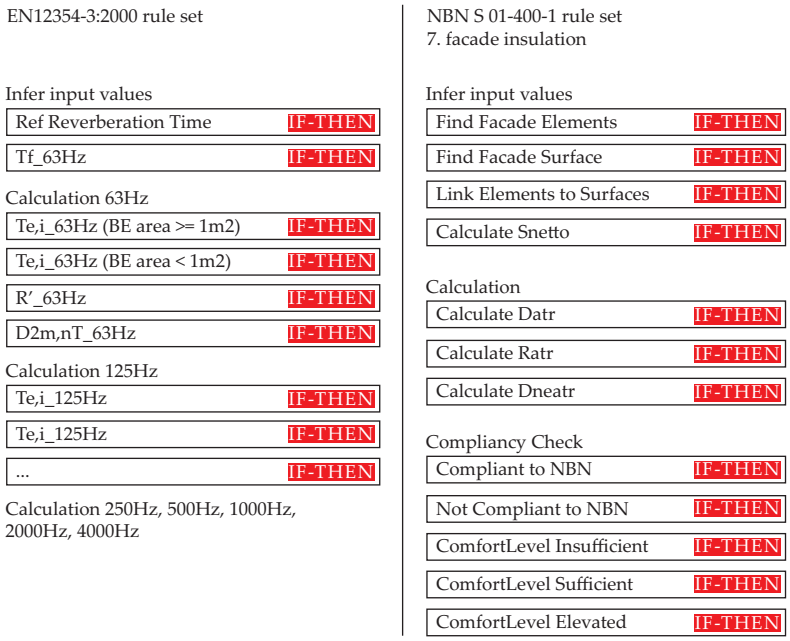


Figure 3.32: The N3Logic rules described in the two rule sets (EN12354-3:2000 and NBN S 01-400-1). The rule that was displayed in Fig. 3.31 is one of the calculation rules in the EN12354-3:2000 rule set, in which the value `Te,i_4000Hz` is calculated.

The two rule sets shown in Fig. 3.32 are available as separate rule sets, independent from each other. Similarly, other rule sets can be described as well using N3Logic, possibly in conjunction with one of the rule sets in Fig. 3.32. As such, one might add a rule set for energy performance checking, a rule set for structural analysis, and so forth.

What is important here, is that the two rule sets shown in Fig. 3.32 deploy their own vocabularies. These vocabularies consist of the concepts that are used in the corresponding documents [Bureau voor normalisatie (Commissie: Geluidsleer - algemeen), 2008, European Committee for Standardization (CEN), 2000]. These terms are typically not used by the designer and are thus often not included in the available information model. For instance, neither the IFC/RDF graph nor the Revit application includes concepts like `Boundary Element` or `Flanking Sound Power Ratio`. This can be considered as a more detailed illustration of what causes the functionality mismatch issue in calculation applications: a certain result is given by calculation applications, but this is not easily related to the concepts typically used by a designer.

To address this issue, 'rule ontologies' were created in OWL that describe the concepts deployed within each rule set separately. For the EN12354-3:2000 rule set, for instance, an OWL ontology was built which includes, among other concepts, the concepts shown in Fig. 3.33. This was similarly done for the NBN S 01-400-1 rule set.

The functionality mismatch issue can to some extent be addressed by linking the rule ontologies of the rule sets to the ontology or ontologies used for modeling the design. The calculation results can thus be directly related to the specific AIM ontology used by a designer in a design context. This can be realized with semantic web technologies by making the appropriate connections between both ontologies using the mechanisms outlined for addressing interoperability issues in section 3.3. If a rule-based conversion approach is used, this results in a calculation framework as schematically shown in Fig. 3.34. The nodes `Location`, `Design`, `IFC`, `Theory`, `Construction Elements` in this schema represent the diverse RDF graphs describing the same design in a different information structure. With rule ontologies, RDF graphs can be constructed in the terms of diverse rule sets (see also Fig. 3.33). The rule set uses the information in this RDF graph to make certain calculations (see also Fig. 3.31 and 3.32). A conversion rule set converts between one of these RDF graphs and the rule ontology used by a specific rule set, following the approach discussed in section 3.3.2.

The approach suggested in Pauwels et al. [2011c] relies on the rule-based conversion approach outlined in section 3.3.2, resulting in 'conversion rule sets' to convert between the rule ontologies and the other available ontologies (Fig. 3.34). This conversion rule set explicitly describes how the reasoning engine can interpret the information represented in the available AIM model [Pauwels et al., 2011c], and thus also how it can communicate calculation results in terms of concepts in this AIM model. As

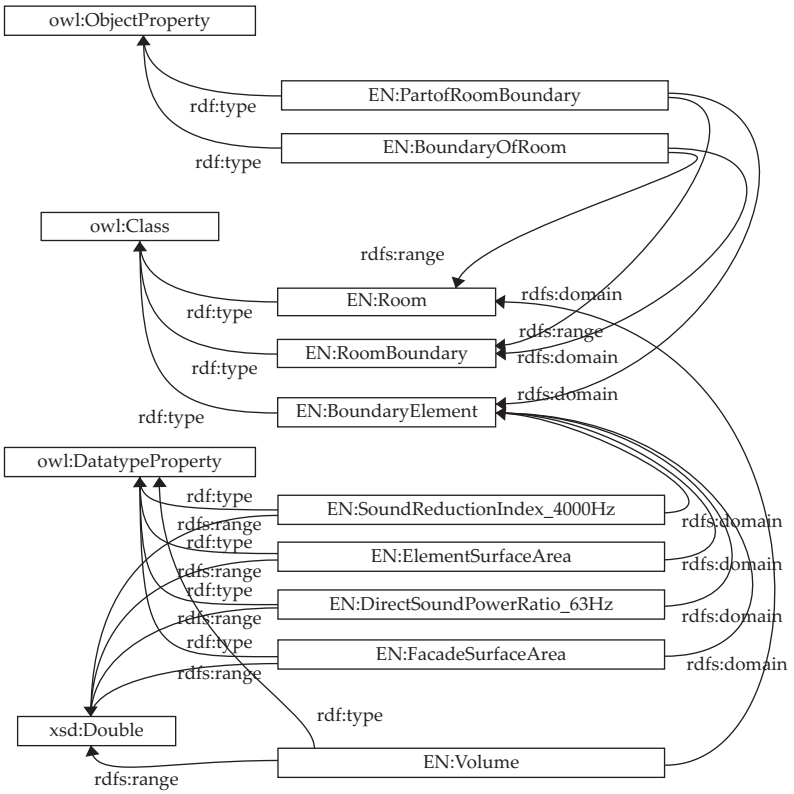


Figure 3.33: Part of the OWL ontology for the EN12354-3:2000 rule set.

a result, it is possible to communicate the calculation results in terms of geometric and material-based concepts [Pauwels et al., 2011c], instead of having to use the terms used in the procedure for acoustic performance checking [Bureau voor normalisatie (Commissie: Geluidsleer - algemeen), 2008, European Committee for Standardization (CEN), 2000]. It can be concluded that semantic web technologies can address the functionality mismatch issue to some extent for calculation applications, assuming that the designer's preferences are recorded in the AIM model that is part of the central RDF graph. However, crucial for this approach is the connection between the AIM model and the RDF graphs corresponding to the rule ontologies used by the calculation applications. This depends mainly on the possibility to combine different representations for the same information, as was also the case for modeling applications and archive applications.

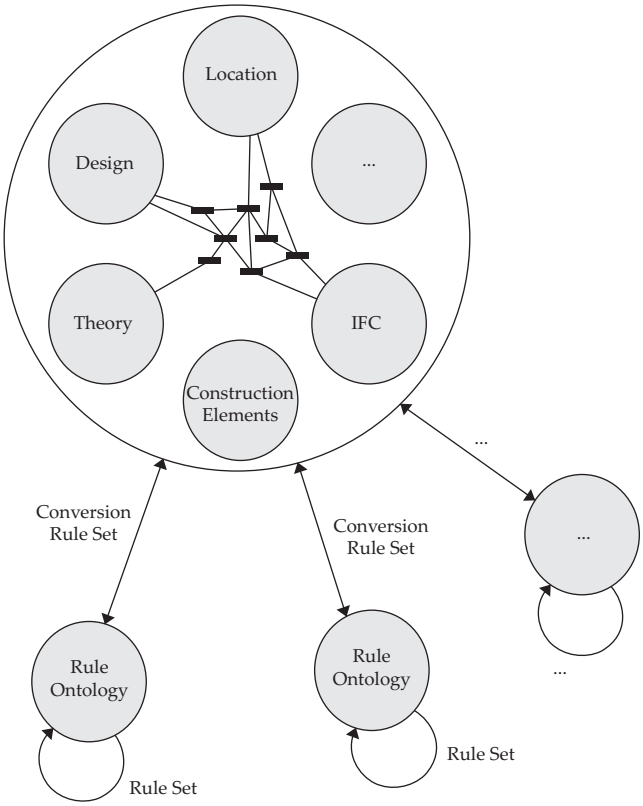


Figure 3.34: Rule sets linked to the core knowledge base deploy a separate ontology definition and a conversion rule set. Each rule ontology can thus be considered as a separate addition of implicit information, each time for a specific domain of knowledge.

3.4.4 Visualization applications

We already pointed at the distinction between two interpretations for 'visualization applications' (section 2.1.4). On the one hand, modeling, archive and calculation applications have an important visualization component, in the form of a Graphical User Interface (GUI) which gives users access to the functionality of the application. On the other hand, visualization applications can also be understood in a more narrow sense, namely, as separate applications that are solely meant for architectural visualization purposes, such as the rendering software 3DS Max, Maya or Blender, photo-editing software such as Photoshop or GIMP, the game engines Unity [Unity Technologies, 2012], Quest3D [Quest3D, 2012], or Unreal [Epic Games, Inc,

2012], and so forth. Section 2.1.4 indicated that, in both cases, the information that is visualized often does not conform to the requirements or desires of the designer.

To address this issue, we chose in section 2.4 to focus on ways for making the information required by the designer available to the application. In this section, we will do this for dedicated architectural visualization applications. This situation was investigated within this research project and documented in Pauwels et al. [2010b]. More details about this research can be found in the part of the article that is documented in appendix D. In this article, it is investigated how semantic building information, as it is contained in the AIM model and the web of information (lowest level in Fig. 3.9 and 3.24), can be visualized in a 3D virtual world.

Comparison of visualization applications

For this research, an analysis was made of existing visualization applications, including the following comparisons [Pauwels et al., 2010b]:

- a comparison of VR systems [Autodesk, 2012b, Graphisoft, 2012, Keough, 2009, Khemlani, 2007, Sketchworlds, 2012] and MR systems [El-Tawil and Kamat, 2006, Golparvar-Fard et al., 2009, Layar, 2012, Roberts et al., 2002, Shin and Dunston, 2009, Thomas et al., 1999, Webster et al., 1996, Wikitude, 2012],
- a comparison of VR environments produced with specialized VR toolkits and software libraries and VR environments produced with game engines that provide a more generic and out-of-the-box functionality, similar to the analyses documented in Al-Najdawi [2007], Marks et al. [2007], Moloney et al. [2003], Wünsche et al. [2005],
- a comparison of some of the most important game engines currently used, similar to the analysis documented in Koehler et al. [2008].

From the outcome of these three comparisons, the Unity game engine [Unity Technologies, 2012] was eventually chosen for the visualization effort in this research project [Pauwels et al., 2010b]. The engine focuses on a fast and intuitive game development for diverse hardware and software environments, including iPhone and iPad applications, immersive installations, and so forth [Unity Technologies, 2012]. After testing, it was considered as one of the best game engines in terms of usability, intuitiveness, and resulting quality in graphics and interactivity. The game engine relies mostly on import through the FBX file format. An FBX file can be obtained from diverse Autodesk products, among which Autodesk 3DS Max. After

importing the FBX file, the engine allows building a virtual world as displayed in Fig. 3.35, in which a building can be interactively explored by virtually walking around in the building.

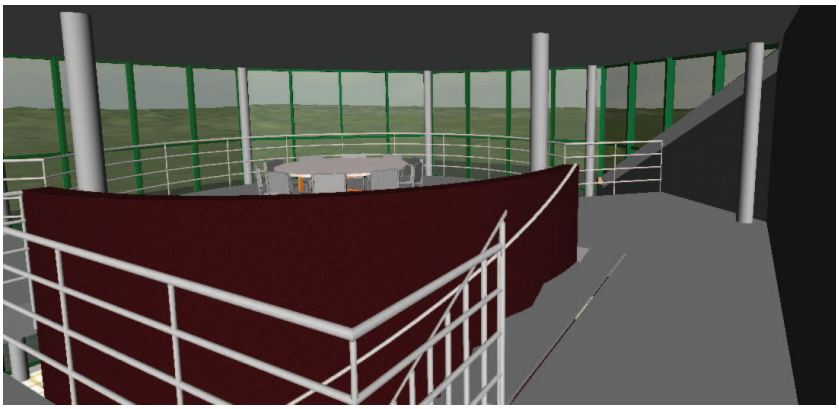


Figure 3.35: Screenshot of the virtual world created for the C3A building in Ghent (Belgium) using the Unity game engine.

Unity provides a useful API that is accessible through C# scripts and JavaScript for basic game engine functionality [Unity Technologies, 2012]. The more advanced API components, such as a VR library, a physics engine, a multiuser library, and so forth, are not available out-of-the-box, resulting in a compact, functional API for fast application development. In investigating Unity, a remarkable support was experienced, together with an elaborate documentation and an active user community. This indicates a sufficiently high level of user satisfaction. The fact that other initiatives for VR visualization, including goBIM [Keough, 2009], for instance, rely on the Unity engine, indicates that this is an appropriate game engine for visualization efforts. The availability of a free version adds to these advantages.

Part 1: the architectural visualization

In the Unity game engine, a three-dimensional building model can be used to produce an interactive 3D visualization of a building. In this research project, a visualization was made of the steel structure (Fig 3.36) for the building in Antwerp that was previously shown in Fig. 3.13. Not only is a 3D building model available for this building that can be imported in Unity, also other information about the building is available in the RDF graph (see also section 3.3.1). This RDF graph contains an IFC/RDF rep-

resentation of the steel structure (Fig. 3.15), an AIM model that describes topological and building element information (Fig. 3.14), and diverse links to information in the online LOD cloud [Bizer et al., 2009, Cyganiak and Jentzsch, 2011], including more detailed material information and geographical information [Pauwels et al., 2010a, 2011b].

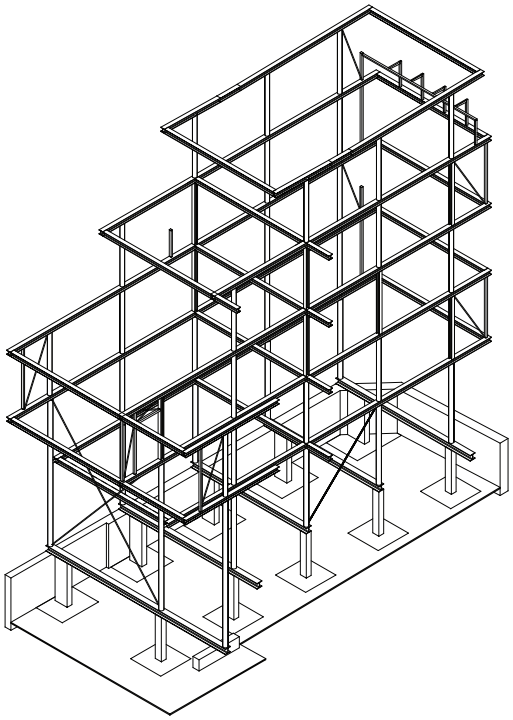


Figure 3.36: The steel structure of the building constructed in Antwerp (Fig. 3.13).

The Unity game engine uses its own information structure for describing the contents it is supposed to visualize. The engine obviously does not start from the information that is described in the online RDF graphs described in section 3.3.1. Alternatively, the RDF graphs do not include a graph that follows the information structure used by Unity. This is a clear example of the interoperability issue documented in section 2.3. This might be solved with semantic web technologies by relying on the mechanisms documented in section 3.3.1 or 3.3.2 to incorporate a Unity representation or information model in the overall RDF graph of the building.

For this case study, however, it was decided to use the BIM model that was modeled in Revit Architecture 2010 and export it to an FBX file. This

FBX file contains the information needed for a graphical representation of the building and excludes extra building information. This FBX file can be imported into Unity, and a few additional actions enable the creation of an interactive virtual world. The mere inclusion of a terrain, a global directional light and a First Person Controller enables a fast production of a basic virtual world (Fig. 3.37).

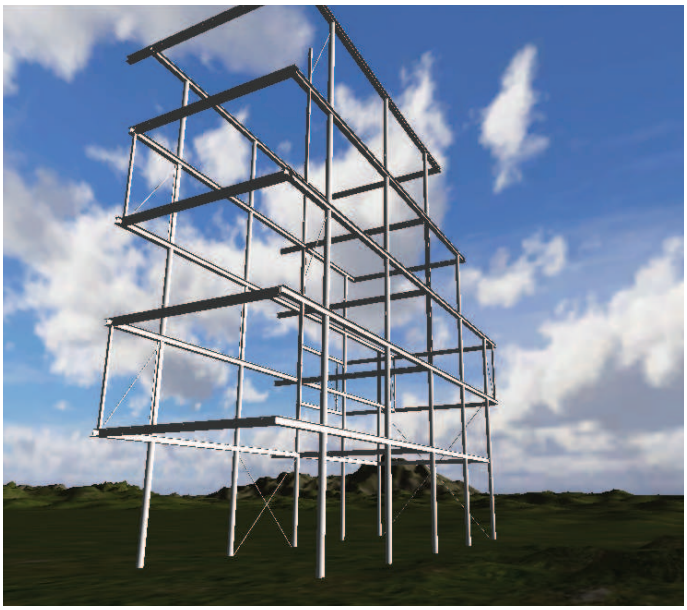


Figure 3.37: Screenshot from the virtual world with the visualized building.

Part 2: linking the architectural visualization to the RDF graphs

Via the Unity API, every building element that was modeled in Revit Architecture is available as a separate object, identified by the original CAD object ID. Because this CAD object ID is also present in the IFC file and thus also in the RDF graph (Fig. 3.38), a connection between the objects in the virtual world and the information in the RDF graph is available. Note, however, that this connection does not connect the objects directly to concepts in the AIM model. This link needs to be realized through the IFC/RDF representation.

The basic architectural visualization allows AEC specialists to walk around in the virtual 3D world (3.37). Through additional scripts, the link between Unity and the RDF graphs is realized. These scripts allow

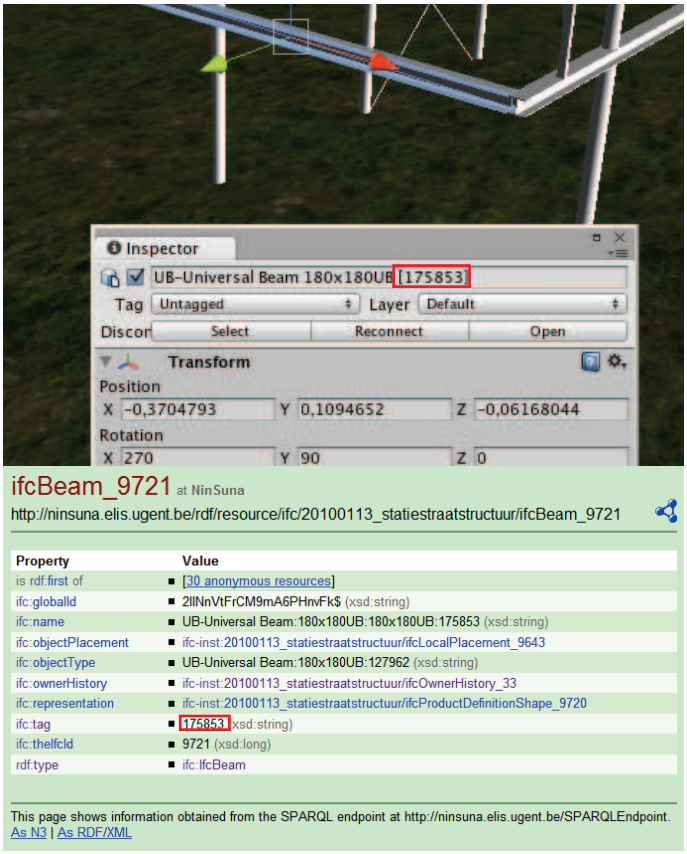


Figure 3.38: The unique CAD Object ID is available in Unity (top) and in the semantic IFC/RDF graph (bottom).

a connection from within the Unity game environment with the available SPARQL endpoint [UGent Multimedia Lab, 2012b]. When connected with this endpoint, any possible query can be sent through over HTTP, allowing the retrieval of information in the RDF graphs. Currently, scripts allow sending a query upon selection of an object. Thus, when a user walks around in the virtual world and selects an object, the CAD object ID of the object is retrieved and a SPARQL query is constructed that allows finding the corresponding object in the IFC/RDF graph and the AIM information linked to this object.

The query result received by the script can be processed or visualized as desired or required in the virtual world. In this test case, it was chosen to present a separate interactive 3D view to the user after selecting an

object. The user view changes into a separate focus view that isolates the selected element from the rest of the building and displays the information available for this element (Fig. 3.39).

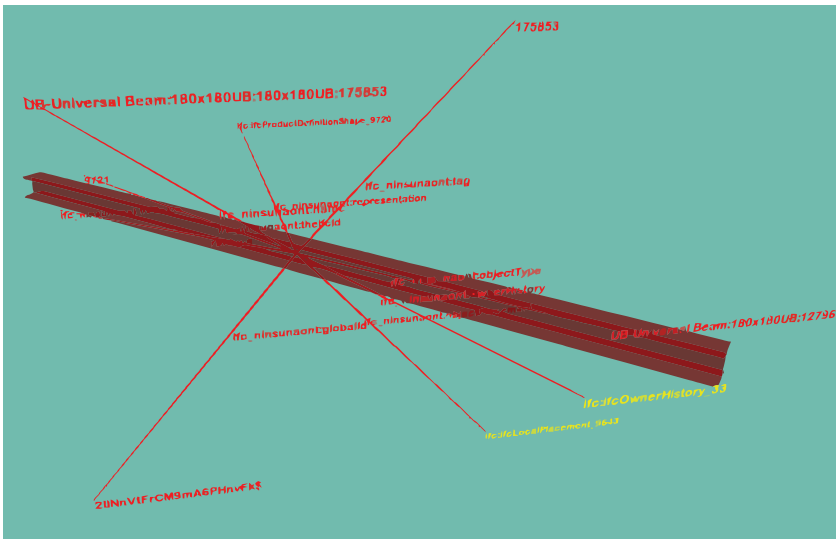


Figure 3.39: Virtual world with an isolated view for the building element highlighted earlier in Fig. 3.38 (right).

Central in the view is the selected object for which zooming, panning and orbiting is available. The information for the object is displayed as a three-dimensional graph that starts from the central object. This graph represents the information as it is available in the RDF graph (Fig. 3.40).

Initially, only directly related objects starting from the object in the IFC/RDF graph are visualized. The user is able to select all nodes in the graph of type `Owl:ObjectProperty`, for instance `ifc:ifcLocal-Placement_9643`, `ifc:ifcOwnerHistory_33`, `ifcont:IfcBeam`, and `ifc:ifcProductDefinitionShape_9720` in Fig. 3.40. Upon selection of one of these nodes, secondary queries are constructed and sent over HTTP to the appropriate servers. Via these queries, extra information is retrieved and made available for visualization in the game engine environment. In this way, a user can access the information in the AIM model or in alternative RDF graphs that are in some way linked to the IFC/RDF graph. The RDF graph displayed earlier in Fig. 3.15 indicates the kind of information that can thus be retrieved.

From this test case, it was finally concluded that it is possible to retrieve in a game engine environment the information described by an AEC

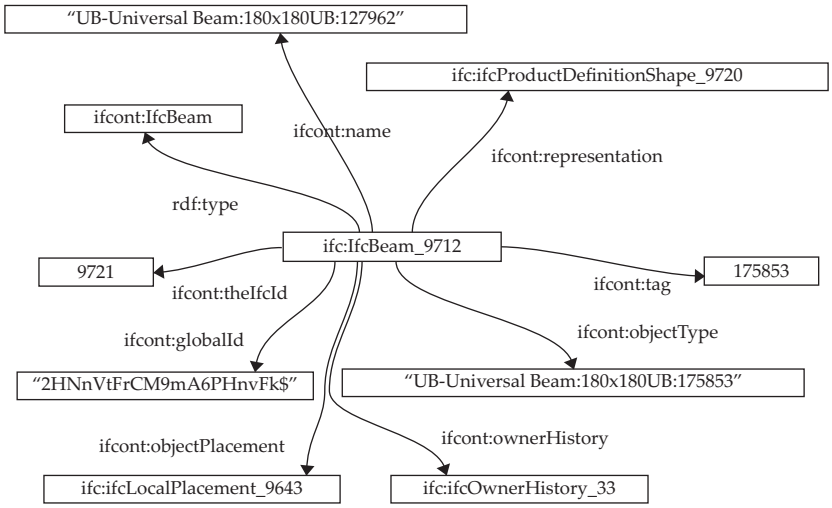


Figure 3.40: The information that is retrieved via the online SPARQL endpoint and visualized in the game engine environment.

specialist in an RDF graph. Assuming that it is possible to customize architectural visualizations with this information, the functionality mismatch in visualization applications might be addressed.

3.5 Conclusion

In this chapter, we have investigated to what extent a linked data approach based on semantic web technologies might allow addressing the interoperability issue in the AEC domain. Because of the relation between this issue and the functionality mismatch issue, which was documented in section 2.4.1, we additionally investigated to what extent also this functionality mismatch issue might be addressed with the linked data approach. Diverse case studies were briefly documented in the context of the AIM framework, illustrating results of this investigation for modeling, archive, calculation and visualization applications.

It is concluded that it is possible to address the interoperability issue with this linked data approach, in the sense that distinct information structures used by corresponding information systems can be combined into one RDF graph or one web of linked data. Two combination methods were outlined, namely a method relying on explicit links between concepts residing in different application domains, and a rule-based method which

enables an active and on-demand conversion between diverse information structures that represent nearly the same information. An information structure as it was shown in Fig. 2.21 is thus technically feasible.

Regarding the functionality mismatch issue, conclusions are less clear. The linked data approach might allow improvements regarding this issue as well, in the sense that the web of data used by applications can similarly be combined with an RDF graph that captures specific concepts and objects in the semantic domain or partial understanding of a design situation (the AIM model). We have seen how the information in this AIM model can be accessed by modeling, archive, calculation and visualization applications. Relying on the available information, applications might thus be customized so that they provide the functionality requested and needed by the designer.

Note that the presented case studies only indicate the feasibility of the approach. In order for the resulting applications to be usable, interfaces are needed: the modeling application requires an interface outside and different from Topbraid Composer, the query interface of the archive application requires an interface as typically provided for archive applications, the presented calculation framework requires a portal for managing the diverse rule sets and exploring the calculation results, the visualization application requires further optimization in the visualization of information, and so forth. One might also target an integration of the presented functionalities, eventually resulting in an AIM framework as presented in conceptual form in the beginning of this chapter.

In the following chapter, we will not look into the ways in which such interfaces or such an integrated AIM framework can be implemented. Instead, we will look into an important issue that requires further attention before such implementation work is started. Namely, following the presented linked data approach for the functionality mismatch issue assumes that designers and AEC specialists in general are able to make explicit representations of their requirements and designers in OWL ontologies and RDF graphs. The AIM model used in the reported case studies gives an idea of what this might look like (Fig. 3.9), but this is presumably not representative for all the knowledge used by designers in their design process.

Additionally, the case studies reported in this chapter show that it is not so easy to realize a mapping that provides a sufficient translation between the available web of data and the AIM model, although this is perfectly feasible from a technical point of view. More specifically, it is possible to state that two concepts in two different representations refer to the same object or concept, but it is very hard to correctly convert or connect all the attributes and properties assigned to both concepts (Fig. 3.9 and 3.24). It

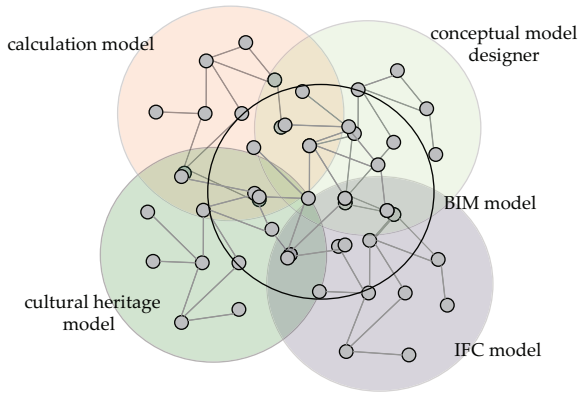


Figure 3.41: The mismatch between diverse information models in the RDF graph: information models in the web of linked data are often connected through 'equivalent' concepts that are in reality not so equivalent, leading to mistranslations and misunderstandings.

appears that the information structure as it was shown in Fig. 2.21, which gives a schematic overview of how diverse information models are combined into one RDF graph, might have to be replaced by the information structure shown in Fig. 3.41. This schema shows that distinct information models are available in the RDF graph for similar objects and concepts, but that there is no exact overlap or mapping available between these models. Consequently, it has to be questioned what the role and effect is of this representation in the design process of a designer.

Similar indications of this 'identity crisis' in the domain of linked data were given before by Halpin et al. [2010]. Using simple `owl:sameAs` links to connect 'identical' concepts in distinct RDF graphs can result at a later stage in wrong inferences. It might thus not be the best idea to connect distinct RDF graphs from different sources via the suggested mappings (e.g. `ifcinst:IfcBeam_1505` and `inst:BEBeam_1` in Fig. 3.15). This does not pose a problem in the small case studies presented in this chapter, but this should be carefully considered in larger projects.

4

Design information in design thinking

As concluded in the previous chapter, it is not entirely clear to what extent the linked data approach can really address the functionality mismatch issue outlined previously in section 2.4. It was concluded that improvements are possible regarding the functionality mismatch issue in the sense that the web of information used by applications can include an unambiguous RDF graph that to some extent represents aspects of design information of a designer. This naturally leads to the two following questions, namely, (1) to what extent can aspects of design information be represented in an unambiguous model or web of information, and (2) what is the role and effect of this representation in the design process of a designer. We will look into these two connected research questions in this chapter by having a glimpse at the most significant theories of design knowledge and design thinking of the last 40 to 50 years. These theories describe how designers deal with design situations and construct knowledge about design. Conclusions formulated in these theories might thus provide some indicative answers to the considered research questions.

4.1 Overviews of theories of design thinking

The theories of design thinking that are discussed in this chapter are chosen from recent overviews of such theories by Bayazit [2004], Cross [2007a], Eastman [2001]. These overviews typically span the last 40 to 50 years of research in design thinking. This chapter relies on the cited overviews, hereby focusing on the theories typically considered as reference works. These are the theories outlined by Simon [1969, 1973, 1996], Schön [1979, 1983], Cross [1982, 2006, 2007b, 2011], and Lawson [1979, 1980, 1999, 2002, 2005a].

It is certainly not our purpose to discuss all theories of design thinking in any comprehensive way, neither is it our purpose to discuss the considered theories of design thinking in complete detail. Rather, our main purpose is to provide a broad view on the diverse theories that have been presented over the last 40 to 50 years in design thinking and to relate these theories to our earlier discussion about information system support in architectural design thinking. More specifically, we want to address the two elements outlined at the end of the previous chapter (section 3.5):

- to what extent can aspects of design information be represented in an unambiguous model or web of information, and
- what is the role and effect of this representation in the design process of a designer

Our overview starts with theories published after World War II (WWII). Obviously, these theories are related to theories dating from the period before WWII. As an example, Cross [2007b] (p. 41) refers to the early 20th century Modern Movement. He points out how the Modern Movement is characterized by a desire to 'scientize design'. In this Modern Movement, arguments are made for reconsidering design as a (partially) rational process, a process that can be based on explicit methods and objective reasoning, 'just like a scientific process'.

Cross [2007b] (p. 41) cites van Doesburg to indicate how methods should be deployed for design according to this Modern Movement perspective. *"Our epoch is hostile to every subjective speculation in art, science, technology etc. The new spirit, which already governs almost all modern life, is opposed to animal spontaneity, to nature's domination, to artistic flummery. In order to construct a new object we need a method, that is to say, an objective system."* [van Doesburg and Van Eesteren, 1924] (p. 91). Furthermore, Cross [2007b] (p. 41) cites Le Corbusier to illustrate how he saw a house as an objectively designed 'machine for living'. *"The use of the house consists of a regular sequence of definite functions. The regular sequence of these functions is*

a traffic phenomenon. To render that traffic exact, economical and rapid is the key effort of modern architectural science." [Le Corbusier, 1929].

The period of the Modern Movement is a period with very strong ideas and aspirations, especially in design thinking. As Cross [2007b] (p. 41) indicates, the whole idea of this Modern Movement is to produce works of art and design by following the cornerstones of 'science': rationality and objectivity. These aspirations of the Modern Movement lessened during World War II, only to resurface again in the 1960s within the new 'design methods movement' [Bayazit, 2004, Cross, 2007a,b].

In the design methods movement of the 1960s, the idea emerges that a design situation might be disambiguated into a well-structured problem (see definition in [Simon, 1973]) by a designer (Fig. 4.1). Using scientific problem solving techniques and rational decision-making, a designer is supposedly able to find the optimal solution to the posed problem. This optimal solution is then put into practice, supposedly addressing the initial design situation and improving part of the world. Note that this approach to design has been adopted in diverse fields of engineering, among which the field of electrical engineering and of very-large-scale integration (VLSI) circuit design (see Gajski and Kuhn [1983] for an early reference).

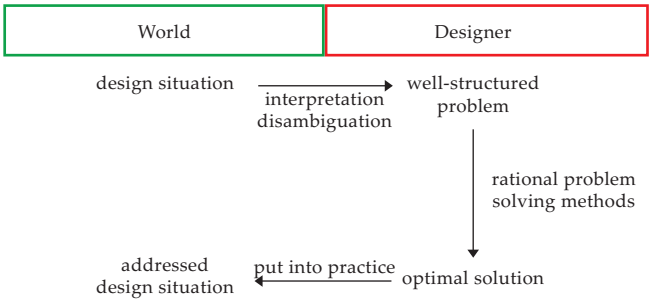


Figure 4.1: A schematic view of how design thinking research in the Modern Movement supposedly considers the usage of rational problem solving methods in design thinking: a design situation is disambiguated into a well-structured problem and then solved through rational methods.

In the following overview, we will first outline the main elements of design thinking in the period of the design methods movement, with the design process as a rational problem solving process (section 4.2 and 4.3). Second, we will also consider how this understanding changed and a period started in which design is considered as an ill-structured problem solving

process[1] that is hard to understand, let alone supported with information systems (section 4.4).

4.2 Design methods and design science

The global situation in the period after WWII stimulated the conception of objective research methods and well-functioning decision-making techniques. *"After World War II, the new techniques that had been used in the design and development of arms and wartime equipment, and the methods and techniques used in developing many new inventions, attracted many designers. Creativity methods were developed mainly in the U.S. in response to the launching of the first satellite, the Soviet Union's Sputnik."* [Bayazit, 2004] (p. 17-18). Another good indication of the post-war situation is given by the reference of Bayazit [2004] (p. 17) to a statement of Horst Rittel: *"The reason for the emergence of design methods in the late '50s and early '60s was the idea that the ways in which the large-scale NASA and military-type technological problems had been approached might profitably be transferred into civilian or other design areas."*. Whereas Bayazit [2004] mainly refers to military developments, other post-war developments can obviously be named. Cross [2007b], for instance, refers to the social problems that needed to be addressed after the war: *"The origins of this emergence of new design methods in the 1960s lay in the application of novel, scientific and computational methods to the novel and pressing problems of the Second World War - from which came civilian developments such as operations research and management decision-making techniques."* [Cross, 2007b] (p. 42).

Because novel scientific and computational methods apparently resulted in important social developments after WWII, this period became particularly known for the emergence of information systems *and* for the explicit belief in the capabilities of these systems in improving one's way of life. Thus, the idea emerges that computational technologies and rational, objective design methods may be able to help in dealing with design

[1]Note that not all researchers in design thinking consider the design process as either a well-structured or ill-structured *problem solving process*. This is caused by the fact that in real design situations, the 'problem' cannot be formulated in a sufficiently and complete and concrete way, let alone be 'solved' in a rational and methodological manner. Instead, it is stated by several authors that a design process typically requires the consecutive formulation of several 'design subproblems', each addressing only part of the situation but gradually defining it [Dorst, 2006] (p. 11). The majority of researchers referred to in the overviews considered in this chapter of the thesis, however, start from the interpretation that design situations are ill-structured and are addressed in a process that aims at shaping the situation into a new and alternative situation that shows improvements at least in some features of the design situation. This is often considered as a kind of problem solving, only not in its rational and methodological sense, but in the sense documented in Dorst [2006] (p. 15): *"the resolution of paradoxes between discourses in a design situation"*.

situations as well, assuming that this includes a kind of (rational) problem solving. This idea is generally considered an important factor for the emergence of the design research society. This is not a society of designers that 'investigate' what they do while designing for their own knowledge, but a society of researchers that investigate *"how an artist [or a designer] is working on his or her work of art to make a contribution to the common knowledge"* [Bayazit, 2004] (p. 16). With this belief, researchers in design thinking more or less proceeded where the Modern Movement stopped, only with the addition of information systems (Fig 4.2).

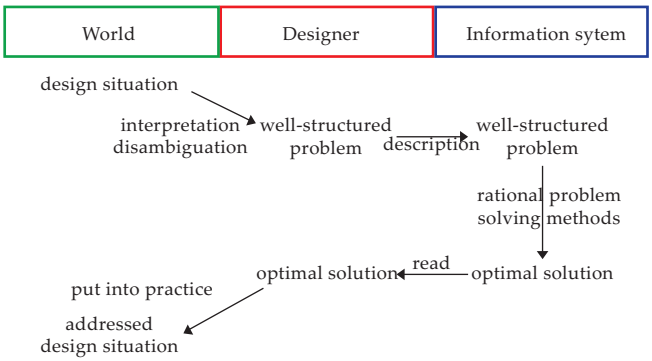


Figure 4.2: A schematic view of how information systems were supposed to impact on the design process and the problem solving process in the period after WWII.

The schemas shown in Fig. 4.1 and Fig. 4.2 can be related to the schemas used in chapter 2. Namely, the process of interpreting a certain design situation and describing it as a well-structured problem, first in the mind of the designer and then in an information system, is similar to how a designer is supposedly able to describe a design situation in his mind and then in an AIM model (Fig. 4.3).

However, in Fig. 4.1 and 4.2, it appears almost taken for granted that the human designer is perfectly capable of assessing a situation in the world and of restructuring it into a static well-structured problem, something that was questioned at the end of chapter 3. The focus then completely goes to solving this problem by 'objective' and 'rational' problem solving methods, either in the human mind or in an information system. This is of course a very brief and sketchy overview of how these movements considered problem solving in their time, but in many cases, the suggestions made by the design methodologists of that time can be understood along these lines. According to the overviews of design thinking that include the 1960s [Bayazit, 2004, Cross, 2007b], there was a lot of optimism

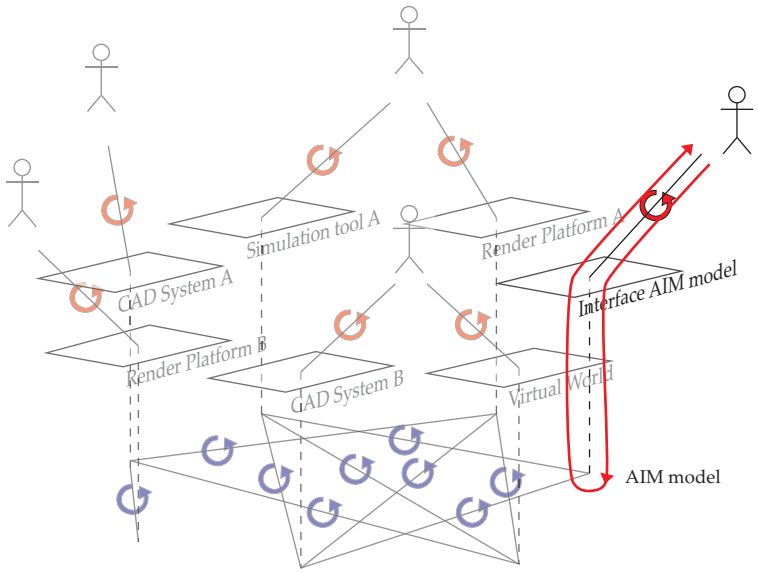


Figure 4.3: In the AIM framework, designers interpret design situations and describe these in AIM models. According to the schema in Fig. 4.2, these AIM models can be interpreted as well-structured problems, and they can thus be used in rational problem solving processes by an information system. Results are communicated back to designers, who supposedly put conclusions into practice, similar to how it was suggested in Fig. 4.2.

in the 1960s about the usability of rational methods in problem solving processes, with information systems topping this optimism.

4.2.1 Design methods

Several examples can be cited to document the situation in the 1960s, many of which have been described elsewhere [Alexander, 1964, Archer, 1965, Bayazit, 2004, Cross, 2007a,b, Eastman, 2001, Gregory, 1966a, Jones, 1970, Newell, 1955, Schön, 1983, Simon, 1969]. One of these examples is the 'Conference on design methods' [Jones and Thornley, 1962]. Several authors, including Cross [2007b], outline this conference as the starting point of 'the design methods movement', a research community in which researchers focus on design methods and design methodology as a subject or a field of inquiry [Cross, 2007b] (p. 41). Cross [2007a] refers to the following statement of Archer for an appropriate description of the situation. *"The*

most fundamental challenge to conventional ideas on design has been the growing advocacy of systematic methods of problem solving, borrowed from computer techniques and management theory, for the assessment of design problems and the development of design solutions." [Archer, 1965]. Another good example that describes the feeling of optimism at that time about 'the design method', is the 'Design methods' book [Jones, 1970]. In this book, Jones refers to a more scientific approach towards design instead of an approach with the 'designer as a creative black box magician': *"Readers who wish to try out these methods for themselves will find the need for skills that are well-developed among scientists, mathematicians or writers but which are often under-developed among designers. This is to be expected when one remembers that many of the new methods have been borrowed from such disciplines as computer programming, psychotherapy, behavioural science, electrical circuit theory and communications theory."* [Jones, 1970] (p. xix).

4.2.2 Design science

Jones [1970] also initiated the evolution in the understanding of design by the design research community from 'design as a result of rational design *methods*' towards 'design as a result of *the scientific method*'. This evolution was in a large part initiated by the conference 'The design method' [Gregory, 1966b]. The term 'design science' gained further significance and importance in the design research society, in part because of its further elaboration in the same conference [Gregory, 1966a].

The search for a design science was picked up by Simon, who was at that time building the foundations for the domain of Artificial Intelligence (AI). Simon's book 'The sciences of the artificial' [Simon, 1969] is perhaps the best example to illustrate the ideas of that time towards design and science. The work documented in Simon [1969] was more or less started by Simon around 1955, together with people like Newell, Shaw and Langley. The starting point of this group was a belief in the possibility that systems may contain intelligence and have the ability to learn and adapt. In 1955, Newell [1955] showed how a machine may be able to play chess and thus to deal with complex tasks and learning/adaptation. Newell and Simon started an intensive collaboration and together with Shaw they developed the Logic Theory Machine (LTM) [Newell and Simon, 1956] and the General Problem Solver (GPS) [Newell et al., 1959a,b].

An important part of the work of Simon's group is the definition of what constitutes 'a problem' [Newell et al., 1963]. A problem is, namely, redefined as a maze accompanied by a decision tree (Fig. 4.4), which is *"a set of paths (possibly partly overlapping) some subset of which are distinguished from*

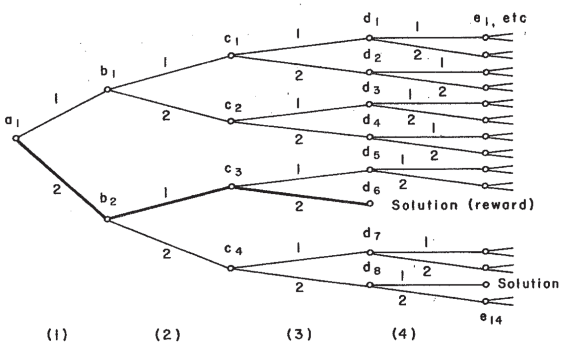


Fig. 1 — A problem maze, alternatives at choice points, m = 2; minimum length of path to solution, k = 3. The shortest path to a solution is given by the choices 2—1—2; it runs from choice point a_1, through b_2 and c_3 to d_6, the solution.

Figure 4.4: Finding the solution for a problem that is represented as a problem maze with a corresponding decision tree (original image from Newell et al. [1963]).

the others by having rewards at their termini. These latter are the 'correct' paths; to discover one of them is to solve the problem of running the maze." [Newell et al., 1963] (p. 11). When comparing this definition to the schema given in Fig. 4.2, the suggested 'heuristic problem solving' approach is completely situated in the right part of the graph (Fig. 4.5), or the lower layer of the schema in Fig. 4.3. The approach of Simon's group, namely, presupposes a well-structured problem (or a maze) to start from. With a clear description of the problem maze, one merely has to select the appropriate heuristic methods to produce the best possible solutions within the considered scope as fast as possible.

This is not only the case for the early work of Simon's group, but for many of the rational or methodological approaches towards design thinking in the 1950s and 1960s. In many cases, it is taken for granted that the human designer is capable of assessing a situation in the world and of restructuring it into a static well-structured problem (Fig. 4.3). However, most of the issues in information system support for architectural design thinking are not caused by limitations or misconceptions in the applications that do heuristic searches for optimal solutions to given problems (Fig. 4.5, right). This is, in relative terms, not that hard because one remains in one and the same environment. Instead, most of current issues

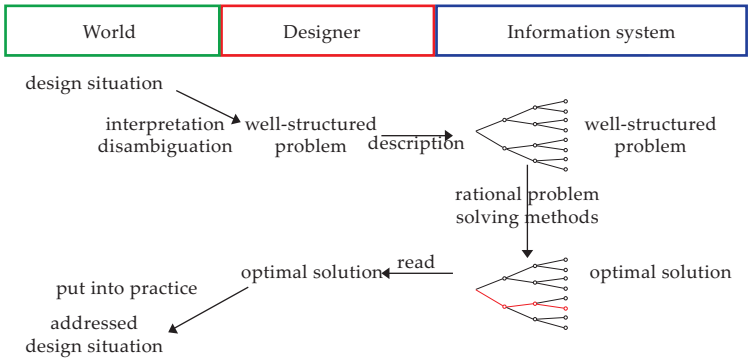


Figure 4.5: Heuristic problem solving in a design thinking context.

in information system support are situated in the process of information exchange from and to this information system (Fig. 4.3).

4.3 Doubts and skepticism in the 1970s

Similar conclusions have been made in the research domain of design thinking of the 1970s. This research concentrates primarily on a 'science of design', thereby following the suggestions given in the 1960s. However, the outcomes of this research lead to a *complete rejection* of the concept of design methodology and of design as a science. This rejection is to a large extent based on the conclusion that design thinking involves also kinds of thinking different from the kind of thinking involved in the heuristic optimization of well-structured problems. This conclusion brought researchers in design thinking back to considering 'design' and 'science' as different entities. Most of the following research initiatives thus concentrate on outlining the differences between both entities. Several examples can be named of such research initiatives. Among those examples is the research on convergent and divergent thinking skills [Schein, 1972], the research on major and minor professions [Glazer, 1974], and the research on wicked problems and tame problems [Rittel and Webber, 1973].

4.3.1 Convergent and divergent thinking skills

Schein [1972] makes the parallel between design and applied science on the one hand, and between science and fundamental science on the other. He concludes that there is a major difference between design/applied sci-

ence and science/fundamental science in that the former requires a 'divergent' thinking style and the latter a 'convergent' thinking style. The nature of these thinking styles is described by Cross [1985] as follows. *"Convergent thinking is primarily concerned with taking in information and producing, or 'converging' on, a single correct answer to the problem. In contrast, divergent thinking is not concerned with the one correct answer. Instead, the emphasis is on a person's ability to generate a wide range of answers; the response is a divergence or an expansion rather than one single answer."* [Cross, 1985] (p. 158).

Several parallels can be outlined between this definition of convergent thinking and the heuristic search process that determines the optimal solutions to a specific well-structured problem (Fig. 4.5, right). In both cases, a complete set of information is considered by the person who is addressing the given situation, who then produces optimal solutions for this situation. What Schein [1972] indicates, is that design thinking does not occur in this way because it does not converge to optimal solutions, but instead keeps widening the set of possible answers to the design situation. However, the exact nature of these divergent thinking skills remains fairly imprecise and unclear.

4.3.2 Major and minor professions

Glazer [1974] (p. 346), as a second example, distinguishes between major professions, such as medicine and law, and minor professions, such as divinity and social work. His viewpoint is briefly documented by Schön [1983] as follows: *"Glazer's distinction between major and minor professions rests on a particularly well articulated version of the model of Technical Rationality. The major professions are 'disciplined by an unambiguous end - health, success in litigation, profit - which settles men's minds' [Glazer, 1974] (p. 363), and they operate in stable institutional contexts. Hence they are grounded in systematic, fundamental knowledge, of which scientific knowledge is the prototype [Glazer, 1974] (p. 348) [...]. In contrast, the minor professions suffer from shifting, ambiguous ends and from unstable institutional contexts of practice, and are therefore unable to develop a base of systematic, scientific professional knowledge."* [Schön, 1983] (p. 23). In his conclusion, Glazer [1974] assigns the convergence of Schein [1972] to the major professions, which he applauds, and divergence to the minor professions, which he dismisses.

4.3.3 Wicked and tame problems

As a last similar example, Rittel and Webber [1973] (p. 159) points out how the original *"professional style of the systems analysts, who were commonly seen*

as forebearers of the universal problem-solvers," did not bring its expected improvements in design. *"With arrogant confidence, the early systems analysts pronounced themselves ready to take on anyone's perceived problem, diagnostically to discover its hidden character, and then, having exposed its true nature, skillfully to excise its root causes. [...] Two decades of experience have worn the self-assurances thin. These analysts are coming to realize how valid their model really is, for they themselves have been caught by the very same diagnostic difficulties that troubled their clients."* [Rittel and Webber, 1973] (p. 159).

Several reasons are named by Rittel and Webber [1973] why such an 'idealized planning system' is unattainable. In essence, it comes down to the kind of problems that planners deal with. These problems are, by their very nature, inherently different from the problems that scientists deal with and are, based on that description, classified as 'wicked problems' by Rittel and Webber [1973], in contrast to the 'tame problems' in science. These wicked problems cannot be dealt with by the system-approach of the first generation, according to Rittel and Webber [1973] (p. 162).

One of the most striking, and perhaps one of the most appropriate ways for describing the difference between tame and wicked problems is to state that tame problems can be solved, whereas wicked problems can only be 're-solved'. *"[...] Social problems are never solved. At best they are only re-solved - over and over again"* [Rittel and Webber, 1973] (p. 160). Approaches of the second generation should thus instead *"be based on a model of planning as an argumentative process in the course of which an image of the problem and of the solution emerges gradually among the participants, as a product of incessant judgment, subjected to critical argument."* [Rittel and Webber, 1984] (p. 138).

4.3.4 What is the problem?

What nearly all examples in the design thinking community of the 1970s have in common, is a focus on the notion that a scientific problem, as it was suggested in the 1960s (see Fig. 4.4 and 4.5), is different from a problem in design and planning. Reformulating a realistic wicked problem into a well-structured problem that can be solved by rational methods does not guarantee an appropriate solution for the initial wicked problem (Fig. 4.6). The design situation and addressed design situation in the world (process A in Fig. 4.6) are of a different nature than the well-structured problem and theoretically optimal solutions in an information system (process B in Fig. 4.6).

Following this conclusion, a turning point followed in which the very concept of design methodology was rejected, even by some of the early pioneers [Cross, 2007a]. *"I've disassociated myself from the field.. There is so*

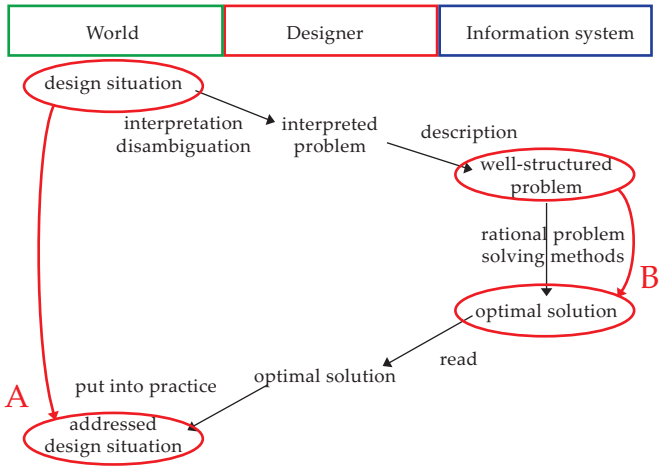


Figure 4.6: A schematic view of how the difference between wicked and tame problems [Rittel and Webber, 1973], or ill-structured and well-structured problems [Simon, 1973], impacts on the design thinking process.

little in what is called 'design methods' that has anything useful to say about how to design buildings that I never even read the literature anymore.. I would say forget it, forget the whole thing" [Alexander, 1971]. Even Jones, who was one of the organizers of the 'Conference on design methods' in 1962 and one of the leaders in the early pioneering research of the 1950s and 1960s, radically revised his point of view. *"In the 1970s I reacted against design methods. I dislike the machine language, the behaviourism, the continual attempt to fix the whole of life into a logical framework"* [Jones, 1977].

It thus became clear that the pioneering research initiatives for design methods did not bring improvements or success to design as a practice, and merely remained in some sort of academic, intellectual microculture. A considerable amount of skepticism and doubts followed towards the understanding of design as a science and towards design as a kind of rational problem solving process. Nevertheless, research continued on how designers address design situations. Many of these research initiatives start with an interpretation of the design process as a process in which an ill-structured or wicked problem is addressed. Although this context results in different discussions, research questions, and investigation in comparison to research embedded in a context of design as a well-structured problem solving process, a significant number of problem solving features remain of considerable importance. The question of how to support the designer with rational methods and corresponding information systems now

changes into the question of how to support designers with information systems in their ill-structured problem solving processes. From this moment onwards, design is typically considered as a problem solving process in the sense described later by Dorst [2006], which is repeated below.

"The 'rational problem solving paradigm' developed in the 1960s and the '70s largely was inspired by developments in AI and the cognitive sciences. The epic endeavor to build intelligent computer systems focused on the ability of such a system to solve ill-structured problems within an open context —somewhat comparable to designing. These systems, based on a rational problem solving approach, represented the 'relevant aspects' of the world, and set up formal procedures to manipulate these representations in order to solve a problem. This approach has failed [Dreyfus, 2002]. Alternative approaches were developed based on situating problem solving activity. [Suchman, 1987, Varela et al., 1991, Winograd and Flores, 1986]" [Dorst, 2006] (p. 11). Dorst [2006] furthermore describes design activity as a situated problem solving process, in which the design problem does not exist as an objective entity in the world, but consists *"of an amalgamation of different problems centered on the basic challenge described in a design brief"* [Dorst, 2006] (p. 11).

From the brief overview in the previous sections of this chapter, an initial answer can be formulated to the research questions given in the beginning of this chapter. It is clear that design situations can not be expressed in their entirety in explicit and unambiguous representations, such as the RDF graphs of chapter 3, because of their ill-structured or wicked nature. This confirms our initial considerations in chapter 2 about the difference between semantics as they are typically considered in a context of computer science and information systems, on the one hand, and semantics as they are typically considered in a context of cognitive science, philosophy and psychology. An information system can only consider a part of the information that is taken into account in the human mind. Similarly, semantic representations of information in information systems capture only to a limited extent the information considered by a human designer. However, this is only a partial answer to the considered research questions. This answer does not imply that information systems cannot be relied upon to support an ill-structured problem solving process such as design. Information systems are able to store and handle representations of design information, which can be used to provide supporting functionality to the designer.

In the following sections, we consider the second research question considered in this chapter, namely, what is the role and effect of the representations handled by information systems in the design process of the designer. We look into theories of design thinking as an ill-structured problem solv-

ing process, so that we can find an indication of how information systems *can* provide support to architectural design thinking.

4.4 A designerly way of thinking and knowing

From the 1980s onwards, the domain of design research starts to follow a course with a focus on design as a discipline in its own right. In this new research context, emphasis is put on how wicked problems are typically resolved; on how planners, practitioners and designers typically operate on their environment. In this research context, the previous research on heuristics and rational problem solving is incorporated, but, in reconsideration of Fig. 4.6, emphasis is now put on the interaction between world, designer, and heuristic methods and not on the heuristics only.

This evolution more or less starts with the publication of 'Design as a discipline' [Archer, 1979]. Archer [1979] suggests that, after investigating concepts like the design method, a science of design, a design science, wicked problems, and so forth, design might not at all be part of science nor of arts and humanities [Cross, 1982]. Design is considered a third discipline in its own right, next to the discipline of science and that of arts and humanities. This existence of a design discipline originates from Archer's conclusion that *"there exists a designerly way of thinking and communicating that is both different from scientific and scholarly ways of thinking and communicating, and as powerful as scientific and scholarly methods of inquiry when applied to its own kinds of problems"* [Archer, 1979] (p. 348). This idea is further developed by Cross [1982] into a 'designerly way of knowing'. Based on earlier research, including the theoretical investigations briefly outlined in section 4.1 to 4.3, Cross [1982] (p. 233) concludes that *"there is a distinct designerly form of activity that separates it from typical scientific and scholarly activities"*.

This distinction of design as a separate discipline in its own right relies on the discussion outlined earlier in section 4.3. Although Archer [1979] distinguishes three disciplines (science, arts and humanities, design), many still stick to a distinction between two disciplines (science/fundamental science and design/applied science) [Glazer, 1974, Schein, 1972]. Lawson [1979], for instance, considers the difference between 'reasoning' and 'imagining' in problem solving processes. But Lawson also concludes that the control of the delicate balance between rational and imaginative thought can be considered one of the most important skills of a designer [Lawson, 2005a] (p. 138). This makes both sides of the distinction not irreconcilable. Whether or not design is that different from science thus remains to be seen.

Nonetheless, because of the explicit focus on 'design as a discipline', research initiatives are started that explicitly focus on the process of design practice itself (process A on the left in Fig. 4.6), leading to significant research outcomes. The idea of design as a discipline in its own right still dominates current design thinking research: design is a reflective discipline with the design thinking process as the unique motor behind design activity [Alexander, 2004, Cross, 2006, 2007b, 2011, Do and Gross, 2001, Goldschmidt, 1991, 1994, Lawson, 2005a, Margolin and Buchanan, 1995, Martin, 2007, Pallasmaa, 2009, Rowe, 1987, Simon, 1996]. A brief review of the main outcomes of this research is given in the remainder of this section, which will result in four main characteristics for design thinking (section 4.4.1 to 4.4.4).

4.4.1 Analogical reasoning

With his theory of 'a designerly way of knowing' [Cross, 1982, 2006, 2011], Cross can be counted among those researchers that made a case for design as a separate discipline in its own right. His theory essentially distinguishes design knowledge from the kinds of knowledge typically at play in the disciplines of the arts and of the sciences [Cross, 1982]. Throughout his publications, Cross strongly associates this kind of knowledge with the specific nature of design thinking. In Cross [1982], he describes this kind of thinking as follows: *"Designers are immersed in this material culture, and draw upon it as the primary source of their thinking. Designers have the ability both to 'read' and 'write' in this culture."* [Cross, 1982] (p. 225). His additional reference to Douglas and Isherwood [1979] illustrates his understanding of design thinking. *"For too long a narrow idea of human reasoning has prevailed which only accepts simple induction and deduction as worthy of the name of thinking. But there is a prior and pervasive kind of reasoning that scans a scene and sizes it up, packing into one instant's survey a process of matching, classifying and comparing. [...] Metaphoric appreciation, as all the words we have used suggest, is a work of approximate measurement, scaling and comparison between like and unlike elements in a pattern."* [Douglas and Isherwood, 1979] (p. viii). Later on, Cross [1990] refers to several other research initiatives that distinguish a very similar kind of reasoning as fundamental for design thinking, thereby mentioning the terms abductive reasoning, productive reasoning and appositional reasoning as called by their respective inventors Peirce, March and Bogen [Bogen, 1969, March, 1976, Peirce, 1958].

This kind of reasoning is obviously relied upon in the interpretation step between design situation and interpreted problem in Fig. 4.6. This step is not even included in the schema that was part of our conclusion

in chapter 2. This is thus an interpretation step that currently only resides in the designer's mind (Fig. 4.7). Because of the crucial role of this interpretation step, diverse exploratory studies have been made of analogical reasoning and of the kinds of reasoning that were considered similar by Cross [1990], namely abductive reasoning, productive reasoning and appositional reasoning [Bogen, 1969, March, 1976, Peirce, 1958]. In appendix E, which is based on Pauwels et al. [2012a], an indication is given of how theories of design thinking relate to Peirce's process of inquiry, which includes abductive reasoning. Many other theories of human cognition, which are out of scope here, can provide additional insights to the ways in which designers think. Theories of direct relevance are the theory by Thomas Kuhn about paradigm shifts and scientific revolutions [Kuhn, 1962], Popper's falsificationism [Popper, 1958], the theory by Michael Polanyi about the tacit dimension of knowledge [Polanyi, 1958], and so forth.

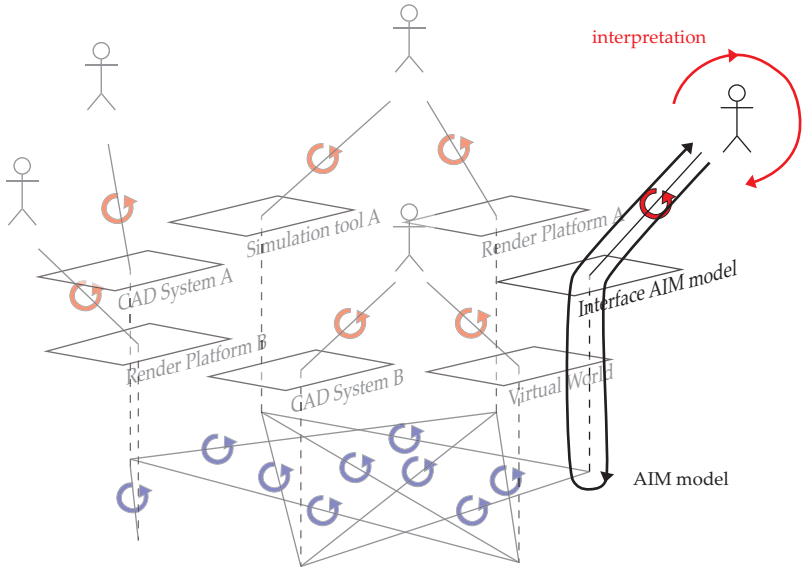


Figure 4.7: In interacting with an information system, the interpretation step, which relies on analogical reasoning, completely resides in the mind of the designer. This step is needed for translating between well-structured and ill-structured problems and vice versa (processes A and B in Fig. 4.6).

The 'work of approximate measurement' or 'metaphoric appreciation' of Douglas and Isherwood [1979] (p. viii) relates to analogical reasoning and the reasoning involved in pattern recognition. Analogical reasoning

is often explained as the cognitive ability to think about relational patterns [Grace et al., 2011, Holyoak et al., 2001, Kokinov, 1998, Ward, 1998]. It allows one to find a structural alignment or mapping between a base and a target pattern residing in (partially) different domains [Gentner et al., 2001, Grace et al., 2011, Hofstadter, 2001, Lakoff and Johnson, 1980, Ward, 1998]. By making such an analogical mapping, familiar knowledge about the base pattern can be related to the target pattern, thereby filling in the gaps of the target pattern and creating new knowledge. In a context of architectural design, analogical reasoning often occurs between a new design-related experience (building, sketch, 3D model, conversation, and so forth) and a previous design experience [Heylighen, 2007]. But also in the very act of sketching, analogical reasoning is crucial, because it allows reinterpreting or 'seeing as', as Goldschmidt [1991, 1994] puts it. In 'seeing as', the designer reinterprets the sketch and, as such, adds new and original meaning to it, thereby generating new ideas [Goldschmidt, 1991, 1994]. Because analogical reasoning is guided by encountered target patterns [Gentner et al., 2001, Grace et al., 2011, Hofstadter, 2001, Ward, 1998], the designer appears to proceed 'in an unstructured and perhaps aimless way'. Therefore, also the earlier mentioned definition of imagining [Lawson, 1979, 2005a] is closely related to analogical reasoning. A similar conclusion is given by Boden's research on 'the creative mind' [Boden, 2004]. She stresses the importance of the incubation phase in creative thinking. In this phase, the conscious mind focuses on other domains, problems, or projects, thus enabling the creative mind to make diverse analogies with the situation at hand [Boden, 2004] (p. 33-35) (Fig. 4.8).

4.4.2 The co-evolution of problem and solution

As Cross [1997] outlines, the kind of thinking involved in dealing with a design situation that is understood as a wicked problem, is furthermore characterized by 'an oscillation between subproblems and subsolutions' (Fig. 4.8) and not by making a single description of the design situation and applying a heuristic method in order to get to a solution (process B in Fig. 4.6). *"During the design process, partial models of the problem and of its solution are constructed side-by-side, as it were."* [Cross, 1997] (p. 439). Dorst and Cross [2001] link this to the notion of 'co-evolution' of problem and solution, which was earlier suggested by Maher and Poon [1996], Poon and Maher [1997]. According to the concept of co-evolution, both problem and solution evolve during a combined process of exploration (Fig. 4.9).

Other names for this phenomenon are 'sense-making' and 'problem framing/setting' [Cross, 1997, Dorst, 2011, Kolko, 2010, Schön, 1979, 1983].

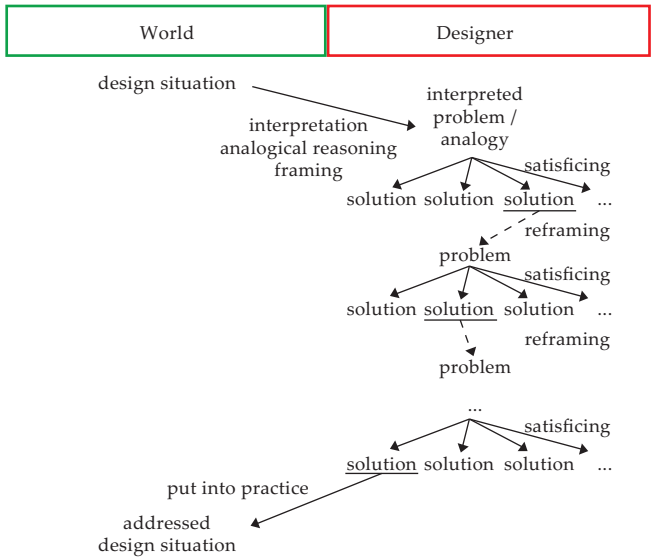


Figure 4.8: Problem and solution are continuously redefined, and the processes from problem to solution are similar to rational problem solving processes based on heuristic optimization techniques. These processes are only a small part of how one addresses the initial design situation and do not include the initial interpretation or framing of the design situation, nor the continuous reframing of problem and solution.

Schön [1979, 1983], for instance, characterizes design thinking as a specific kind of problem solving, in which the designer *"must make sense of an uncertain situation that initially makes no sense"*. Making sense of the situation then happens by switching back and forth between problem and solution, while continuously reframing both. What is probably the most interesting moment in this reframing process, is the point where a solution is considered satisfactory and ready to be put into practice. This moment, which is often referred to as the moment in which 'the flash of insight' occurs, is described by Cross [1997] as the moment in which the two oscillating points, 'problem' and 'solution', are still and close enough to be bridged by 'an apposite concept'. *"The crucial factor [...] is the bridging of these two partial models by the articulation of an apposite concept [...] which enables the models to be mapped onto each other. The 'creative leap' is not so much a leap across the chasm between analysis and synthesis, as the throwing of a bridge across the chasm between problem and solution. Such an apposite 'bridge' concept recognizably embodies satisfactory relationships between problem and solution. It is the recognition of a satisfactory bridging concept that provides the 'illumination' of*

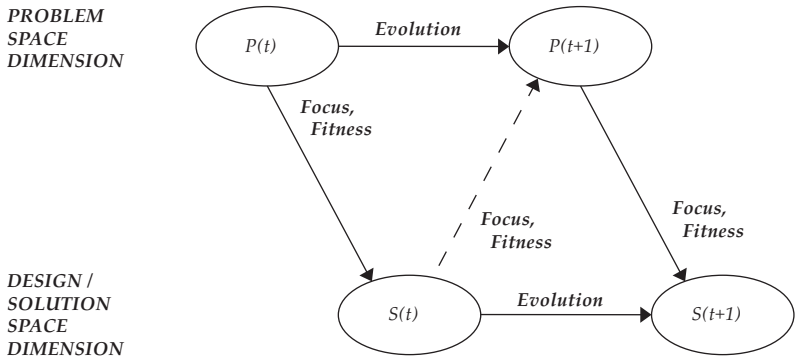


Figure 4.9: The 'problem-design exploration model' (original image from Maher and Poon [1996]).

the creative 'flash of insight'" [Cross, 1997] (p. 439-440).

The idea of co-evolution of problem and solution stresses that boundaries are continuously set and reset around the model within which a solution is sought: the design situation is continuously framed and reframed. Simon [1955, 1979, 1996] explicitly refers to a 'bounded rationality', whereas Schön [1983] discusses the topic in close relation to the perspective of 'Technical Rationality'. *"From the perspective of Technical Rationality, professional practice is a process of problem solving. Problems of choice or decision are solved through the selection, from available means, of the one best suited to established ends. But with this emphasis on problem solving, we ignore problem setting, the process by which we define the decision to be made, the ends to be achieved, the means which may be chosen. In real-world practice, problems do not present themselves to the practitioner as givens. They must be constructed from the materials of problematic situations which are puzzling, troubling, and uncertain."* [Schön, 1983] (p. 39-40).

Just how important the element of reframing the design situation as a problem-solution pair really is, is made clear by Schön [1983] in his documentation of the difference between problem solving in a rational world and problem solving in the real world (see also Fig. 4.6). *"When we set the problem, we select what we will treat as the 'things' of the situation, we set the boundaries of our attention to it, and we impose upon it a coherence which allows us to say what is wrong and in what directions the situation needs to be changed. Problem setting is a process in which, interactively, we* name *the things to which we will attend and* frame *the context in which we will attend to them."* [Schön, 1983] (p. 40).

The design thinking process shown in Fig. 4.8 also includes steps from (sub)problem to (sub)solution. In these steps of the design thinking process, rational problem solving techniques remain of considerable importance. Cross [1982] and Simon [1996] additionally stress the solution-focused character of this problem solving mode of a designer. One of the first features recognized by Cross [1982] in design activity, is *"its reliance on generating fairly quickly a satisfactory solution, rather than on any prolonged analysis of the problem."* [Cross, 1982] (p. 224). In this respect, one can refer to the process of 'satisficing' as it was recognized by Simon [1996] (p. 28-30), meaning that a designer rather produces *"any one of what might well be a large range of satisfactory solutions rather than attempting to generate the one hypothetically-optimum solution"* [Cross, 1982] (p. 224).

4.4.3 The back-talk of the situation: the experience

The experience is a third crucial element in the sense-making process. Sketches are among the most obvious examples of experiences that a designer goes through at design time in the context of architectural design thinking [Purcell and Gero, 1998]. As Goldschmidt [1991, 1994] indicates, sketches are not only visual expressions of what one wants to express, they are also elements for reinterpretation and thus for generating all kinds of new knowledge. Cross similarly refers to the importance of sketching as it enables a designer to explore several solutions and problems to a certain design situation at once, thereby considering several levels of detail at once [Cross, 1999, 2006]. Schön [1983], in turn, refers to the habit of many a designer to continuously make representations of the design situation at hand in documents, plans, sketches, and so forth, thereby allowing a designer both to answer a previous design situation, and frame the design situation anew into an alternative perspective. We will refer to these representations in an architectural design context as 'design tryouts' (Fig. 4.10).

The notions of design representations, design tryouts and experiences align to our discussion of co-evolution and problem-framing in section 4.4.2 (Fig. 4.8). Every experience (of a design representation) generates a new understanding or mental model of the situation, which the designer may use to reframe the design situation and which eventually results in new design tryouts and experiences in an altered design approach. This continuously starts anew, in accordance to the concept of co-evolution (Fig. 4.10). To say it in Schön's words, the designer *"shapes the situation in accordance with his initial appreciation of it, the situation 'talks back', and he responds to the situation's back-talk."* [Schön, 1983] (p. 79).

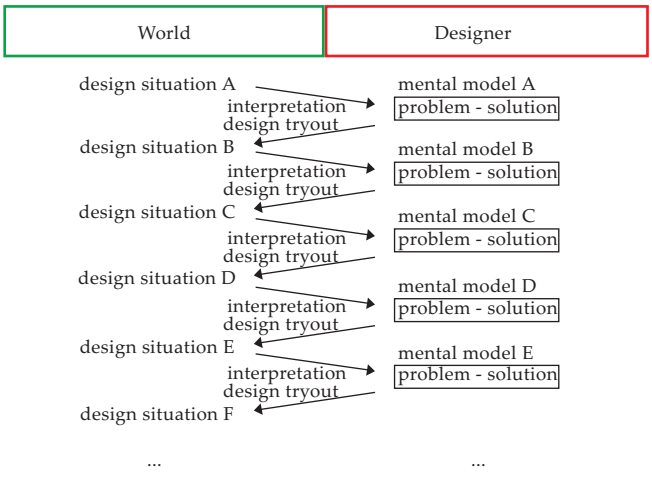


Figure 4.10: In addressing a design situation, a designer typically switches back and forth between the world and a mental model of this world.

This process is often referred to by Schön [1983] (p. 54-55) as 'reflecting-in-action': a process in which one thinks about actions while doing them. While practitioners[2] are doing something, they may find some feel about what they are doing, enabling them to do it over and over again. As such, they are thus reflecting in the action about what they are doing, thereby enabling themselves to reproduce and/or customize their action. *"Improvisation consists in varying, combining, and recombining a set of figures within the schema which bounds and gives coherence to the performance. As the musicians feel the direction of the music that is developing out of their interwoven contributions, they make new sense of it and adjust their performance to the new sense they have made."* [Schön, 1983] (p. 55). Part two of Schön [1983] tests this theory of reflection-in-action in the context of several professional practices, including design. It is described how designers continuously make representations of the things they are designing, such as sketches, plans, physical models, and so forth. These design tryouts are devised according to the designer's framed problem and solution [Maher and Poon, 1996, Poon and Maher, 1997, Schön, 1979, 1983], bounded rationality [Simon, 1955, 1979, 1996], or mental model. As indicated in Fig. 4.11, this mental model is the result of the interpretation step or analogical reasoning step for a certain design situation (section 4.4.1).

[2]Schön [1983] (p. 54-55) refers to architectural designers, baseball pitchers and musicians as example practitioners.

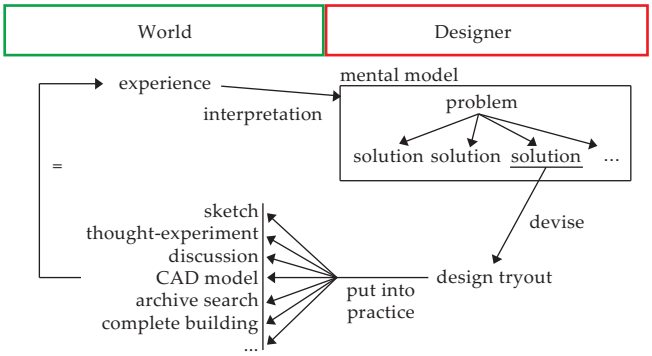


Figure 4.11: The role of design tryouts and experiences in the design thinking process. Design tryouts can be considered as products of the mental models residing in the mind of the designer. They constitute the experiences that are interpreted by designers in the design thinking process, and thus shape the further design process.

By making (a representation of) a design tryout, a designer aims at confirming the initial mental model of the design situation (which includes a problem-solution pair). This design tryout can be represented in many forms, including a sketch [Goldschmidt, 1991, Pallasmaa, 2009, Purcell and Gero, 1998], a thought-experiment, a simple discussion [Conklin, 2005], a CAD model, and so forth. Because of the high level of complexity of a design situation, a design tryout tends to *"produce consequences other than those intended. When this happens, the designer may take account of the unintended changes he has made in the situation by forming new appreciations and understandings and by making new moves."* [Schön, 1983] (p. 79). By experiencing the result of a design tryout, a new understanding or mental model of the situation thus emerges, which *reframes* the previous design situation and thus alters the design process (Fig. 4.11).

We already saw how the interpretation step in Fig. 4.11 has no parallel in current information system support for architectural design (Fig. 4.7). This step results in a mental model of the design situation, which resides only in the designer's mind and of which only reflections can be made available by making design tryouts (Fig. 4.12). According to the theories discussed in the previous sections, this mental model should not be understood as a static description of a (well-structured) problem, with a corresponding solution, but as a continuously changing model of an ill-structured or wicked problem (co-evolution), in which a designer each time attends to different parameters of the wicked problem in specific design tryouts. Each design tryout hereby represents only a limited semantic

domain, which is but a partial reflection of the 'complete' design situation. At the beginning of this chapter, we questioned to what extent an explicit representation of this model could be made. In a sense, design tryouts, represented in sketches, physical models and so forth, are typically considered as representations of this mental model by the theories considered in this section (for instance, [Goldschmidt, 1991]). However, they are seldom considered as exact 'copies' of this mental model. Instead, they are most often considered as representations that *result* from this mental model and that form, as such, initiators for further reflection about this mental model. In a sense, they are considered new experiences that confirm or refute the mental model and as such help designers in their design process (Fig. 4.11 and 4.12).

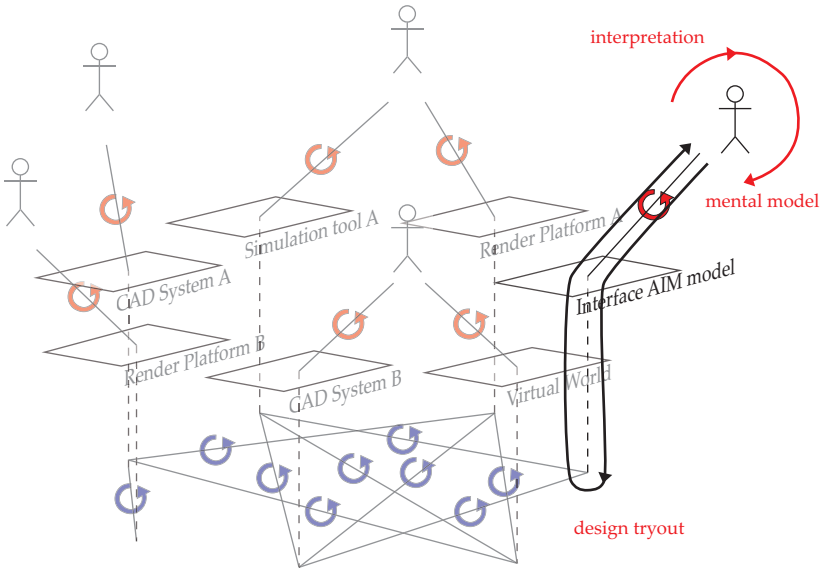


Figure 4.12: The position of the 'mental model' and the 'design tryout' in our schema concluded in chapter 2 (Fig. 2.19). Information coming from any information system is interpreted, subsequently resulting in an altered mental model in the designer's mind. This altered model then results in new, alternative design tryouts, which can happen in sketches, CAD models, complete building projects, discussions, and so forth.

By analogy, the information models that were outlined in chapter 2 and the RDF graphs that were documented in chapter 3 can be understood as alternative representations for the same design tryouts that are typically represented with traditional media (sketches, physical models, and

so forth). Consequently, an RDF graph (and other models) should be understood as a medium that can be used by the designer to represent and produce design tryouts (Fig. 4.12). As a result, the main merit of such information systems is that they form additional environments for making design tryouts, which result in their turn in alternative new experiences.

4.4.4 Guiding principles

In the previous sections, a crucial shift has already been made regarding the interpretation of central issues in information system support for architectural design (Fig. 4.12). However, an important last element is still missing from our discussion of theories of design thinking as a discipline in its own right, namely the 'guiding principles' element. This element has various names and forms, but we will use the term coined by Lawson [2005a] (p. 159) here, namely, guiding principles. These can be understood as the personal background knowledge or the knowledge by experience of a designer. They consist of familiar design patterns that a designer relies upon throughout the design thinking process. A designer thus never starts a design from an empty page, never from scratch or a blank mind. Instead, a designer always relies on a lifetime of knowledge built up by experiences. These guiding principles are not only crucial in the analogical reasoning step, they are equally crucial in the other steps or phases in the design thinking process. Lawson [2005a] (p. 179) documents how these guiding principles, in combination with a mental model of the situation at hand, essentially guide practitioners, among which designers, through their thinking process. They play an important role not only in framing the design situation ('interpretation' in Fig. 4.13), but also in generating solutions for a problem and devising experiments ('devise' in Fig. 4.13), and in learning from experiences ('learn' in Fig. 4.13).

According to Lawson [2005a] (p. 159), these guiding principles include not just objective, factual information, but include much more information, involving, for instance, motivations, beliefs, values and attitudes. Guiding principles may be very intense and clearly structured or, on the other hand, vague and unclear, but they always influence design decisions one way or another. These characteristics of guiding principles can be related to the tacit dimension suggested by Polanyi [1958, 1967], which states that some knowledge cannot be formalized and is essentially experience-based, vague and thus tacit. In some research initiatives, guiding principles are almost considered part of a 'personal religion', thereby implicitly redefining design as 'a very complicated act of faith' [Jones, 1970] (p. 3). This refers to the sometimes profound intensity of the designer's belief in personal guid-

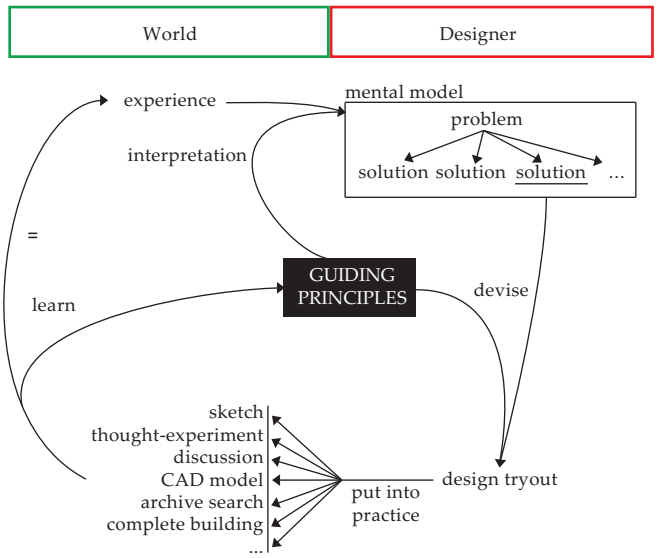


Figure 4.13: The set of guiding principles as the central element needed for (1) framing the design situation, (2) generating solutions for a problem and devising experiments, and (3) learning from design tryouts and experiences.

ing principles, making it 'morally right' to follow them. Ward [1994] similarly indicates how imagination relies almost entirely on known concepts, and, although modifications are made, they are typically only constituted by different combinations of known elements. It is hard to entirely step outside one's own categories and beliefs, also in imagining [Ward, 1994].

In what form guiding principles are stored in the mind of a designer, is very unclear. From our discussion of the nature of guiding principles and of how they are used in diverse reasoning processes, one could conclude that they are formed by experience - mental model pairs (Fig. 4.13), but this is but one of the many possibilities. What is clear from the previous sections, however, is that this background information of the designer is continuously and actively reorganized and restructured in memory rather than passively recorded and recalled. In reconsideration of our initial research questions of this chapter, even if designers would be able to make their background knowledge explicit in an unambiguous representation, they would still have to update this representation every single second according to the changes made by addressing new situations and experiences.

4.5 Implications to information system support for architectural design thinking

In terms of the kind of information system support that can be provided to an architectural designer, the practical linked data approach based on semantic web technologies is not that different from the traditional way in which information systems provide support to the architectural designer (chapter 2). Both the traditional information systems and the systems that are based on a linked data approach are parts of the world with which designers interact. In terms of the schema that we used throughout this chapter for documenting theories of design thinking (Fig. 4.14), information systems are thus situated on the left side.

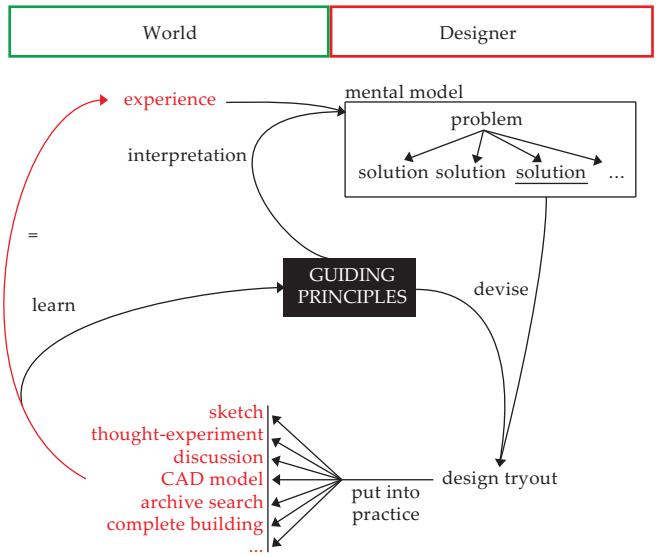


Figure 4.14: Information systems provide diverse environments for making design tryouts and, as such, influence design processes.

Information systems provide very diverse environments for making design tryouts, including sketches, CAD models, computer simulations, archive searches, and so forth, which are, in turn, 'experienced' by designers and, as such, influence their design thinking processes. The analogical reasoning, the interpreting, the mental model, the devising of experiments, and the learning, however, are all intrinsic parts of the (individual) design thinking process and cannot be directly influenced by information systems. They can only be influenced indirectly by making the designer

go through specific alternative experiences. For instance, using one particular modeling application (for instance, AutoCAD) instead of another (for instance, Revit) generates different experiences and will result in a different design process and final design. Similarly, using a particular archive application that contains historical buildings (for instance, Art Nouveau buildings) will result in different design decisions than using a particular archive application that contains buildings built over the last ten years.

Situated on the left of the schema in Fig. 4.14, information systems provide extra environments for producing design tryouts. In this application development approach, the design thinking takes place entirely in the human mind and the application is 'just' another environment to conduct design tryouts in. These design tryouts are devised according to the guiding principles or background knowledge of the designer, which is built up through a lifetime of experiences. A design tryout thus inhibits an expectation that is to be confirmed or contradicted by the outcome of this design tryout (cfr. 'act of faith' [Jones, 1970] (p. 3)). The design tryout eventually confirms the expectation, leading to a strengthened belief in a set of guiding principles, or the design tryout contradicts the expectation, leading to surprise and a need to modify guiding principles.

4.5.1 Example tools

Standard tools for design tryouts

Modeling environments are good examples of how applications can function as tools for design tryouts. But also archive applications, calculation applications and visualization applications can be considered as tools for design tryouts. In an archive application, for instance, designers already have the kind of results in mind that they expect to see when performing a search or browse action. The result produced by the system confirms or refutes this expectation. Similarly, when preparing a model in an application for energy performance calculation, for instance, designers already have in mind what kind of energy performance they expect the system to produce. According to our schema in Fig. 4.14, this expectation follows, on the one hand, from their previous experiences with energy performance calculations and with applications for energy performance calculations, and, on the other hand, from the current interpreted mental model of the considered design situation. Again, the result produced by the application confirms or refutes this expectation, thereby influencing the future design process.

In using such tools for design tryouts, architectural designers not only learn about architecture, they also learn to handle the tools they are using.

This happens similarly to how architectural designers learn both about architecture and about sketching by making those sketches. This learning-by-doing takes place in the mind of the designer and not in the information system (see Fig. 4.14).

When developing applications in accordance with this tools for design tryouts approach, one should bear in mind that the information structure in the application has a limited impact on the design thinking of the designer. As was also indicated by Lawson [2005b], applications for design tryouts should not be considered as 'oracles' that calculate 'the truth' and are to be obeyed by the architectural designer. Instead, these applications should be considered as 'mini-oracles' that can be ignored if necessary, but that might provide some interesting new insights in some cases, similar to how other tools (paper and pencil, physical models, etc.) might only in some cases provide new insights. As indicated in Fig. 4.15, information systems thus provide additional tools for design tryouts. The actual reasoning takes place in the human mind, which is perfectly fit to handle ill-structured problems and is thus far more powerful than any of the possible information structures of the application. The main merit of producing applications in this approach is that the number of tools for design tryouts in the world notably increases.

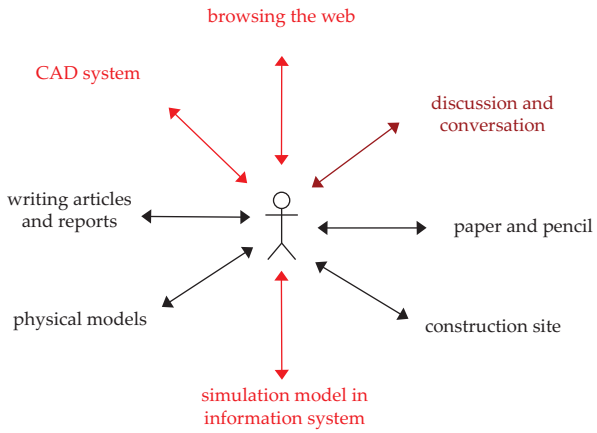


Figure 4.15: Information systems as standard environments for making design tryouts (CAD system, browsing the web, simulation model in information system, discussion and conversation), in addition to the traditional environments used by designers for design tryouts (paper and pencil, construction site, physical models, writing articles and reports, discussion and conversation.

Many, if not all, of the traditional tools for design tryouts (indicated in black in Fig. 4.15) have digital equivalents. Paper and pencil have an equivalent in sketch applications (e.g. SketchUp, AutoCAD), libraries have an equivalent in archive applications (e.g. MACE), physical models have an equivalent in digital (3D) models (e.g. Revit, ArchiCAD, SketchUp, AutoCAD), mathematical calculation procedures have an equivalent in simulation applications (e.g. Robot structure, EPB software), and so forth. Also discussion and conversation, which are the central features of collaborative design, can be accommodated in digital environments, using simple video conferencing tools (e.g. Skype) or more complex tools for annotation (e.g. Google Docs, ShareLaTeX). The actual reasoning of the designer is only supported indirectly by these tools, and a designer should choose wisely which environment to use when devising design tryouts (Fig. 4.14 and 4.15).

Specialized assistance in problem solving

Many of the information systems for design tryouts can be interpreted as applications in which a well-defined problem is formalized so that, within the scope of the formalized situation, one or more optimal solutions of the well-defined problem can be sought by the system. These information systems appear to follow the traditional problem solving approach that was discussed in section 4.2, and indeed, much of the same functionality is included. However, in several cases, the ill-structured nature of design situations is taken into account and the tools have a specialized but limited purpose. The purpose of the application is to allow architectural designers to model a snap-shot of the design situation at hand, including all the constraints which the designer wants to attend to in this design tryout, and generate several possible 'optimal'[3] suggestions using heuristic methods, given the bounded rationality of the information structure in the system.

Numerous example information systems with the given aim can be named in the domain of architectural design (see Bittermann [2011], Erbas et al. [2011], Wong and Chan [2009] for a selection of recent initiatives). An interesting development in this approach is to use 'metaheuristics' [Strobbe et al., 2011]. This approach allows an information system to choose (through metaheuristics) which near-optimal heuristic method it should use for addressing a specific design situation. When combining this approach with generative approaches, in which a system generates design alternatives based on the available information [Caldas, 2007, Krish, 2011,

[3]The given suggestion is optimal within the scope of the well-structured problem, but at best near-optimal in the scope of the ill-structured problem.

Shea et al., 2005], such information systems are able to suggest various design alternatives to the designer which might include some alternatives that a designer has not considered before and which might thus alter the design process accordingly (Fig. 4.16). These advising information systems can be devised as limited but specialized 'agents'. Such agents can provide very specific functionality for very specific situations or they can be customized to the design situation at hand with its required functionality. Such agile agents are often better tailored to the end user than standard and large software packages with a complexity that goes past the desires and needs of specific designers.

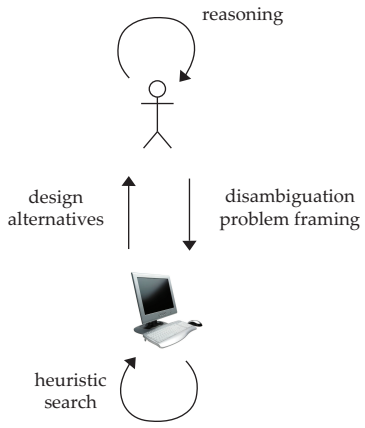


Figure 4.16: Information systems as specialized assistants in problem solving act upon a disambiguated design situation to generate diverse design alternatives. These design alternatives are communicated to the designer, who may or may not use these suggested design alternatives.

Note, however, that also these information systems or agents are essentially tools for design tryouts and that also in this case the actual reasoning and decision making takes place in the designer's mind. This can be recognized in the situation where designers use applications that are meant as 'assistants in problem solving' to generate design alternatives, but eventually discard these alternatives for something entirely else that they already had in mind. As is indicated in Fig. 4.14, a designer starts the design tryout with a hypothetically 'optimal' solution already in mind. The application will presumably be able to alter the designer's mind, only if the application suggests something completely different and thus causes doubt in the designer's mind. The design alternatives that are generated by the suggested information systems [Caldas, 2007, Krish, 2011, Shea et al., 2005, Strobbe

et al., 2011], might thus very well not influence the design process at all (Fig. 4.16). This typically occurs when designers have a very strong belief in their guiding principles and are less open to alternative design solutions.

Similar conclusions are made by the analysis of the impact of CAD tools on creative 'problem solving' in Roberson and Radcliffe [2009]. It is concluded that designers should not attempt to put the *whole* design situation or problem into the application in order to 'solve it all'. *"Because creativity is associated with novelty, comprehensive tools for creative work will be neither possible nor necessary to develop, any more than it is necessary for a pencil to include all functions for drawing"* [Mitchell et al., 2003]. Instead, bounded well-structured problems should be given to an information system so that it can rapidly generate some design alternatives that a designer might not have thought of earlier. Information systems in such a strategy should in this case focus on making the diverse parameters in the design situation and their effect on this design situation 'highly visible and easily modifiable' [Candy and Edmonds, 1997].

4.5.2 The role of semantic web technologies

We indicated in the beginning of this thesis (section 2.1) that large architectural firms or construction firms have the ability and the resources to accommodate an in-house group of expert programmers. This group typically develops custom in-house information systems, directly tailored to the needs of the design team. Examples that were similarly pointed out in section 2.1 are the office of architect F. Gehry [Gehry, 2012] with their 'Digital Project' modeling application, and the office of Foster and Partners with their 'Specialist Modeling Group' (SMG) [Peters and De Kestelier, 2006]. The availability of expert programming skills allows to construct and use custom tools for specific architectural design situations. Although Digital Project relies on CATIA modeling software, important features were added that were of particular use to the architectural design style or guiding principles of architect Gehry. The SMG in Foster and Partners similarly provides custom optimal design tools compliant with specific needs in specific design projects, leading to what could be considered specialized assistant tools. Such firms are capable of customizing standard tools for design tryouts (section 4.5.1) and developing specialized assistant tools (section 4.5.1).

However, because regular architectural design firms or construction firms do not have the ability to rely on an in-house group of expert programmers with a similar task, the given examples are to be considered exceptions to the rule. Smaller architectural design firms or construction

firms typically rely on a far more pragmatic approach [Pauwels et al., 2012b], in which information systems provide custom assistance only to a limited extent. To enable a more customized assistance also to such architectural firms and construction firms, information systems *should focus on making the diverse parameters in the design situation and their effect on this design situation 'highly visible and easily modifiable' [Candy and Edmonds, 1997]*, which is exactly what might be within reach by relying on a linked data approach with semantic web technologies.

The main improvement generated by using a linked data approach based on semantic web technologies in this context consists in the improved information management facilities, as was concluded in chapter 3. With this improvement, the linked data approach mainly addresses the interoperability issue that was outlined in the beginning of this thesis. The linked data approach, namely, provides the possibility to formalize distinct information structures used by corresponding information systems and explicitly and unambiguously combine these information structures into one RDF graph or one web of linked data (section 3.3.1 and 3.3.2).

Note that, because of the reasons outlined in section 4.3.4, these information structures will not contain *all* information that is considered by the designer, but only a subset that forms a distinct semantic domain. Semantic web technologies, however, allow this semantic domain to be represented as precise and usable as possible for a particular design situation, on the one hand, and allow the information represented in the semantic domain to be more easily linked to other semantic domains with related representations of information. This situation is to some extent anticipated by Lawson [1999, 2002], which refers to the possible future usage of advanced information technologies to build systems that can handle the information that is used in one semantic domain (see 'verge', 'hip', 'eaves', 'ridge' for a roof form [Lawson, 1999, 2002]) and relate this information to the descriptions used in another semantic domain (see 'box', 'pyramid', 'extrusion' [Lawson, 1999, 2002]).

The functionality mismatch issue will only be addressed to a limited extent, because of the reasons outlined throughout chapter 4. It is not possible to document all the guiding principles which an architectural designer relies upon in the design process. These guiding principles are different for every person and change continuously under the effect of new incoming experiences. Even if these guiding principles *could* be formalized (in an information system), it is not feasible to continuously manage this dynamic web of linked data. Therefore, applications cannot be customized more than they already are. As was pointed out in chapter 3, semantic web technologies allow designers to describe both the information (see modeling

AIM models in section 3.4.1) and the functionality (see calculation applications in section 3.4.3) that they desire or require in their environment for design tryouts. Because of the improved information management, a linked data approach based on semantic web technologies allows to realize this customization of applications more easily, as was indicated in section 3.4.

4.6 Conclusion

This chapter was started with the objective of finding out to what extent the linked data approach with semantic web technologies can address the functionality mismatch issue outlined before in information system support for architectural design thinking. For this purpose, the two following connected research questions were outlined in chapter 3.

- To what extent can aspects of design information be represented in an unambiguous model or web of information?
- What is the role and effect of this representation in the design process of a designer?

We have addressed these two connected research questions in this chapter by having a glimpse at the most significant theories of design knowledge and design thinking of the last 40 to 50 years. These theories describe how designers deal with design situations. The overview of these theories, more particularly the theories about the difference between well-structured and ill-structured problems, very soon (section 4.3.4) confirmed our initial considerations in chapter 2 about the difference between semantics as they are typically considered in a context of computer science and information systems, on the one hand, and semantics as they are typically considered in a context of cognitive science, philosophy and psychology. An information system can only consider a part of the information that is taken into account in the human mind. Consequently, it is clear that design situations cannot be expressed in their entirety in explicit and unambiguous representations, because of their ill-structured or wicked nature.

This answer does not imply that information systems cannot be relied upon to support an ill-structured problem solving process such as design. Because information systems are able to capture and handle some of the concepts used by designers, one might still aspire to provide support for the design process as an ill-structured or wicked problem solving process. The overview of theories of design thinking in section 4.4 therefore concentrated on the second research question above: what is the role and effect of

this representation in the design process of a designer. From this overview, we have eventually outlined four main points that characterize the way in which architectural designers handle information and use information for decision-making in the design process.

- the importance of *analogical reasoning* in producing creative ideas,
- the concept of *co-evolution* of (sub)problems and (sub)solutions,
- the world with which the designer interacts (*experiences*),
- *guiding principles* or background knowledge built up by experiences.

A concluding schema that displays these four main characteristic elements of design thinking is given in Fig. 4.14. Central in this schema are Lawson's guiding principles, which constitute the knowledge by experience of a designer. This background knowledge continuously changes under the effect of the designer's interaction of the world. It is not clear how the background knowledge or guiding principles are stored in a designer's mind, and even if this was known, it would most probably be impossible to update its corresponding representation in an information system. These guiding principles help architectural designers to recognize and categorize a new experience through an analogical reasoning process, whether this be the interface of an information system, the feedback given by a colleague designer, a sketch, a thought experiment, or a CAD model. Furthermore, they help designers to devise design tryouts that can be used to confirm or refute their mental model of a design situation. And finally, these guiding principles also allow learning from experiences resulting from design tryouts. By continuously iterating through such a process, problem and solution are gradually defined and refined. This iterative problem solving process does not result in a perfectly fit solution to a certain situation. Instead, this process takes only certain parameters into account, thereby creating a context of bounded rationality, in which any human being is able to find a local optimum by satisficing [Simon, 1955, 1979, 1996].

The schema in Fig. 4.14 forms a sufficiently good framework to explain what role and effect the information represented in an information system constitutes in the design process of a designer. It is concluded in section 4.4.3 that a representation in an information system should not be considered as an exact parallel for the mental model of the designer for a specific design situation. Instead, it should be considered as an *indirect* parallel representation for the mental model of the designer for a specific design situation, similar to how a sketch, a physical model, a discussion, and so forth, can be considered as indirect parallel representations for design situations.

Information systems are in this understanding nothing more than parts of the world with which designers interact. They are thus part of a process of prediction and testing, in which the designer only attends to very specific elements in the design situation, and does not attend to the wicked problem as a whole. Interaction with applications then occurs similar to how it was discussed by Purcell and Gero [1998] for sketches, diagrams and drawings.

Considering these indications, the following conclusion can be made regarding our question of the extent to which information system support can be provided for architectural design thinking. As is indicated in Fig. 4.14, information systems primarily serve as environments for making design tryouts, including sketches, CAD models, computer simulations, archive searches, and so forth, which are, in turn, 'experienced' by designers and, as such, influence their design thinking processes. The analogical reasoning, the interpreting, the mental model, the devising of experiments, and the learning, however, are all intrinsic parts of the (individual) design thinking process and cannot be directly influenced by information systems.

This clearly indicates the bounded impact that current information systems have on the complete design process. Nevertheless, even if information systems are mainly of significance in the design tryout phase, they can still provide important support for the architectural designer. Namely, they provide alternative environments that are different in functionality than the traditional environments designers can rely upon for making design tryouts. These alternative environments typically provide elaborate and complex functionalities to designers, allowing them to make in-depth design explorations that would not be feasible with pencil and paper, for instance. Additionally, these information systems also provide an improved information management to the designer, especially when relying on the linked data approach documented in chapter 3, even if the information that is being managed can never be an exact 'copy' of the designer's mental model.

5
Conclusion

THIS thesis started from the observation that numerous information systems are available which support various aspects in architectural design, but that support by these information systems suffers from substantial shortcomings. In chapter 2, four categories of information systems were discussed, namely modeling applications, archive applications, calculation applications, and visualization applications. Modeling applications are hereby considered as applications that allow designers to model a design into a certain information structure, which typically stands for a 2D or 3D representation. Archive applications are understood as applications that enable designers to more easily find information that could in some way be useful for their design situation. Calculation applications typically reuse the information that is available via modeling applications. They are considered as applications that provide a complex set of rules and algorithms for inferring information from existing information. Visualization applications, finally, are understood as applications that allow producing architectural visualizations, typically using the information from modeling applications.

We have looked into the issues that are typically encountered when using the considered types of information systems. As a result, it was indicated how the considered information systems have difficulties of coping with the amount and complexity of information that is typically considered in the AEC domain. Many domain experts with different backgrounds typically meet within the context of a building project, each of them compos-

ing a personal understanding of the building design and providing with this personal understanding a specific contribution to the project. Additionally, each of these experts relies on diverse software tools, which causes a multiplication of the number of information structures at play in a project. Because all these information structures, human interpretations as well as representations in information systems, are all part of one and the same project, a lot of information flows emerge between these information structures. The architectural design needs to be communicated to the structural engineer, the structural engineer needs to take into account the design of the electricity engineer, compliance is needed with all kinds of regulations and standards, and so forth. Crucial in this context of continuous information flows are the points where two information structures meet and interpretation is needed, because these are the points where misunderstandings and corresponding mistakes emerge. Such points of information exchange can be subdivided in points of information exchange (1) among information systems, (2) between information system and designer, and (3) among designers.

We have considered only the first two points of information exchange in this thesis, namely, the points of information exchange (1) among information systems, and (2) between information system and designer. From our overview of information systems, two central issues are outlined, one for each kind of information exchange. The first issue in information system support is a *lack of interoperability* among information systems. The second issue is a *functionality mismatch* or a *mismatch between the functionality provided by information systems and the functionality expected by end users*. By addressing these two central issues, information system support in architectural design thinking might be improved.

In the remainder of chapter 2, we have first documented diverse strategies for addressing the interoperability issue: sharing information in the wild, the remodeling effort, kernel-level interoperability, the centralized information structure, the software suite strategy, and the linked data approach. When relying on semantic web technologies, the last approach appeared to be the most promising, because these technologies allow to explicitly and unambiguously connect information deployed in diverse application domains and applications, using one common language with a common logical basis. Two important reasons why this approach might be better compared to the other approaches, is (1) that semantic web technologies rely on a common language for describing information, namely the Resource Description Framework (RDF), and (2) that semantic web technologies appear to be deployed on a global scale. Consequently, information that would typically be unavailable in the other strategies, has a

notably higher chance of being incorporated in the linked data approach, making this currently one of the most promising approaches.

Additionally, we have made a parallel in chapter 2 between the interoperability issue and the functionality mismatch issue. With the functionality mismatch issue, we indicated that the information presented by a system does not conform to the needs and/or desires of the end user. It is as if two different information models are maintained, by the human user and by the information system, and both models do not match. The resulting functionality, either expected or provided, is based on these information models and does not match either. This situation is parallel with the interoperability issue, because two information models are available that to not map onto each other. Considering this parallel, it was suggested that the linked data approach might also be used for addressing the functionality mismatch issue. With semantic web technologies, the linked data approach provides designers the possibility to represent those aspects of design information considered by them and explicitly and unambiguously connect this representation to representations deployed in diverse information systems. Relying on the information that is available in this global web of data, applications might be customized so that they provide the functionality requested and needed by specific designers in specific design contexts.

In chapter 3, we have investigated in closer detail to what extent this linked data approach can really address both issues by going through diverse small case studies. Besides giving an idea of how information is represented in RDF graphs, two initial case studies in section 3.3 indicated how different RDF graphs can be linked together into one (global) web of information that includes diverse representations of information with respect to their proper syntaxes and semantics. One might choose to make explicit links between different RDF graphs, which each represent a part of the considered information, using simple and relatively static links between concepts of these graphs. Alternatively, one might choose to make implicit links between different RDF graphs by relying on rules and a reasoning engine. In this approach, explicit rules provide an unambiguous representation of how a representation of information (an RDF graph) can be inferred from another RDF graph. When using this approach, the connection between two representations is obtained on-demand. As such, these two mechanisms to link diverse RDF graphs can be used to address the considered interoperability issue.

Relying on the parallel between the interoperability issue and the functionality mismatch issue, it was investigated to what extent a linked data approach with semantic web technologies might also be used to address

the functionality mismatch issue. This was tested for the four application types outlined earlier. With a case study about the Book Tower in Ghent, an indication was first given of how concepts from the designer's perspective on a design situation can be modeled into a separate RDF graph using semantic web technologies, and how this RDF graph can be linked to other RDF graphs. Second, we gave an indication of how diverse linked RDF graphs can be searched through using the SPARQL query language provided for RDF graphs. Using this query language and an appropriate user interface, an archive application can be built that takes better into account the concepts typically used by specific designers. Third, we showed how the set of rules and algorithms typically used by calculation applications can be represented using a dedicated rule language. Because rule languages in the semantic web domain are based on the same logical principles as the other languages used by semantic web technologies, the rules described with these languages form an explicit and unambiguous representation of rule functionality in a specific context, similar to how RDF graphs form explicit representations of information in specific contexts. Finally, we indicate how the resulting web of information, which includes the diverse interlinked RDF graphs, can be accessed by visualization applications. Produced architectural visualizations can take into account this source of information, eventually enabling architectural visualizations that are tailored to the designer and the design situation.

However, our investigation of the linked data approach in chapter 3 also resulted in the two following connected research questions. These essentially question the extent to which the considered approach can really address the interoperability issue and the functionality mismatch issue.

1. To what extent can aspects of design information be represented in an unambiguous model or web of information?

2. What is the role and effect of this representation in the design process of a designer?

These two research questions are looked at in chapter 4. In this chapter, a brief overview is given of theories of design thinking of the last 40 to 50 years, leading to significant insights regarding the usage of information by designers. First, these theories confirm that external representations of designerly knowledge, such as the kind of representations that would result from using semantic web technologies, differ substantially from the designerly knowledge used by designers in their design thinking process. Furthermore, it was concluded that architectural designers rely on this designerly knowledge in the designer's mind and use external representations, such as a graph in the semantic web, a sketch or a physical model, as

external mediators or design tryouts. This is a small but important difference. This clearly indicates the bounded impact that current information systems have on the complete design process: information systems that provide support to designers should primarily be considered as additional parts of the world, with which architectural designers can engage for making design tryouts. Similar to a paper and a pencil, a CAD system or a simulation environment allows a designer to make design tryouts. This is how information systems are generally used nowadays, knowingly or not. Also the linked data approach that was documented in chapter 3 results in such an information system support.

Even if information systems are mainly of significance in the design tryout phase, they can still provide important support for the architectural designer. They provide, namely, alternative environments that are different in functionality than the traditional environments designers can rely upon for making design tryouts. These alternative environments typically provide elaborate and complex functionalities to designers, allowing them to make in-depth design explorations that would not be feasible with pencil and paper, for instance. Additionally, these information systems also provide an improved information management to the designer. Semantic web technologies to some extent allow customizing the exchange of information among diverse information systems to the requirements and desires of the designer.

In the last chapter, two perspectives are offered towards future information system support, and an indication is given of the role that semantic web technologies can play in the resulting information systems. One perspective indicates how standard tools for design tryouts, such as the modeling, archive, calculation, and visualization applications of section 2.1, can be used for design tryouts. Semantic web technologies first and foremost provide the possibilities to improve the management of information in such information systems, in the sense that they provide better tools to address the interoperability issue.

The second perspective outlined at the end of chapter 4 presents an approach in which specialized information systems can be built which are able to provide functionality tailored to the end user. Such an approach is in some cases already used in large architectural firms or construction firms, which have the ability to rely on expert in-house programming teams. Semantic web technologies allow to bring this approach to some extent to smaller architectural firms or construction firms. They allow representing a certain situation in custom terms, which can subsequently be used by various custom rule sets, custom algorithms and custom queries. As such, an information system can emerge that provides custom function-

ality to specific design situations.

By taking into account the ill-structured nature of design situations, the information systems resulting in this perspective have a specialized but limited purpose. A possible purpose of the application could be to generate several possible suggestions for addressing the design situation at hand, thereby relying on a partial representation of the design situation. Alternatively, one could develop a specialized agent which uses a specific set of (meta)heuristic methods to find out how the design situation at hand should optimally be addressed. These advising information systems can be devised as limited but specialized 'agents'. With the help of semantic web technologies, our abilities and facilities can be improved to customize those agile agents for a functionality that is better tailored to specific design situations.

References

A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.

S. Abdul-Ghafour, P. Ghodous, B. Shariat, and E. Perna. A common design-features ontology for product data semantics interoperability. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 443–446, 2007.

S. Abdul-Ghafour, P. Ghodous, B. Shariat, and E. Perna. Towards an intelligent CAD models sharing based on semantic web technologies. In *Proceedings of the 15th ISPE International Conference on Concurrent Engineering*, pages 195–203, 2008.

N. Al-Najdawi. Introduction to visualization using game engines. AHRC Methods Network Workshop "From Abstract Data Mapping to 3D Photorealism: understanding emerging intersections in visualisation practices and techniques", 2007. Available online: http://www.arts-humanities.net/system/files/gameenginedevelopments-1.pdf. Last accessed on 22 February 2012.

C. Alexander. *Notes on the synthesis of form*. PhD thesis, Harvard University Press, Cambridge, MA, USA, 1964.

C. Alexander. The state of the art in design methods. *Design Methods Group Newsletter*, 5(3):3–7, 1971.

C. Alexander. *The nature of order: an essay on the art of building and the nature of the universe*. CES Publishing, Berkeley, CA, USA, 2004.

A. Aliseda. Logics in scientific discovery. *Foundations of Science*, 9(3):339–363, 2004.

A. Aliseda. *Abductive reasoning: logical investigations into discovery and explanation*. Springer-Verlag, Dordrecht, NL, 2006.

ARC school of architecture La Salle. OIKODOMOS case repository: a digital library of housing cases, 2012. Available online: http://www.-oikodomos.org/caserepository/. Last accessed on 22 February 2012.

L.B. Archer. Systematic methods for designers. *Developments in design methodology*, pages 57–82, 1965.

L.B. Archer. Design as a discipline. *Design Studies*, 1(1):17–20, 1979.

E. Arlati, E. Bogani, and A. Cammarata. MACE - metadata for architectural contents in Europe. In *Proceedings of the International Conference of Education, Research and Innovation*, 2008.

Autodesk. AutoCAD - 2D and 3D CAD design and documentation software - Autodesk, 2012a. Available online: http://usa.autodesk.com/-autocad/. Last accessed on 22 February 2012.

Autodesk. Navisworks project review software - Autodesk, 2012b. Available online: http://usa.autodesk.com/navisworks/. Last accessed on 22 February 2012.

Autodesk. Revit Architecture - building design software - Autodesk, 2012c. Available online: http://usa.autodesk.com/revit-architecture/. Last accessed on 22 February 2012.

F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *Description Logic Handbook: Theory, Implementation, and Applications*, pages 47–100. Cambridge University Press, Cambridge, MA, USA, 2003.

N. Bayazit. Investigating design: a review of forty years of design research. *Design Issues*, 20(1):16–29, 2004.

D. Beckett and T. Berners-Lee. Turtle - terse RDF triple language. W3C team submission, 2011. Available online: http://www.w3.org/Team-Submission/turtle/. Last accessed on 22 February 2012.

J. Beetz, J. Van Leeuwen, and B. de Vries. IfcOWL: a case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):89–101, 2009.

T. Berners-Lee. Notation 3 (N3): A readable language for data on the web, 2005. Available online: http://www.w3.org/DesignIssues/Notation3.-html. Last accessed on 12 July 2010.

T. Berners-Lee. CWM - a general purpose data processor for the semantic web, 2009. Available online: http://www.w3.org/2000/10/swap/-doc/cwm. Last accessed on 22 February 2012.

T. Berners-Lee and D. Connolly. Notation 3 (N3): a readable RDF syntax. W3C team submission, 2011. Available online: http://www.w3.org/-TeamSubmission/n3/. Last accessed on 22 February 2012.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):35–43, 2001.

T. Berners-Lee, D Connolly, and S. Hawke. Semantic web tutorial using N3, 2003. Available online: http://www.w3.org/2000/10/swap/doc/-tutorial.pdf. Last accessed on 22 February 2012.

T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: a logical framework for the world wide web. *Theory and Practice of Logic Programming*, 8(3):249–269, 2008.

M.S. Bittermann. Sustainable conceptual building design using a cognitive system. In *Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAADFutures)*, pages 297–313, 2011.

C. Bizer and R. Cyganiak. D2R server - publishing relational databases on the semantic web, 2010. Available online: http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/. Last accessed on 22 February 2012.

C. Bizer and T. Gauss. Disco - hyperdata browser: a simple browser for navigating the semantic web, 2007. Available online: http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/. Last accessed on 22 February 2012.

C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

C. Bizer, A. Jentzsch, and R. Cyganiak. State of the LOD cloud, 2011. Available online: http://www4.wiwiss.fu-berlin.de/lodcloud/state/. Last accessed on 22 February 2012.

M.A. Boden. *The creative mind: myths and mechanisms*. Routledge, Taylor & Francis Group, London, UK, second edition, 2004.

J.E. Bogen. The other side of the brain II: an appositional mind. *Bull Los Angeles Neurological Societies*, 34(3):135–162, 1969.

M. Böhms, P. Bonsma, M. Bourdeau, and F. Josefiak. Semantic product modelling with SWOP's PMO. In *Proceedings of the 7th European Conference on Product and Process Modelling: eWork and eBusiness in architecture, engineering and construction*, pages 95–104, 2009a.

M. Böhms, P. Bonsma, M. Bourdeau, and A.S. Kazi. Semantic product modelling and configuration: challenges and opportunities. *Journal of Information Technology in Construction*, 14:507–525, 2009b.

G. Booch, R.A. Maksimchuk, M.W. Engle, B.J. Young, J. Conallen, and K.A. Houston. *Object-oriented analysis and design with applications*. Addison-Wesley Professional, Boston, MA, USA, third edition, 2007.

D. Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, 2004. Available online: http://www.-w3.org/TR/rdf-schema/. Last accessed on 22 February 2012.

D. Brickley and L. Miller. FOAF vocabulary specification 0.98, Namespace document (Marco Polo Edition) 2010. Available online: http://xmlns.-com/foaf/spec/. Last accessed on 22 February 2012.

BuildingSMART International. BuildingSMART - international home of openBIM, 2012. Available online: http://www.buildingsmart.com/. Last accessed on 22 February 2012.

Bureau voor normalisatie (Commissie: Geluidsleer - algemeen). NBN S01-400-1: Akoestische criteria voor woongebouwen - critères acoustiques pour les immeubles d'habitation, 2008.

L. Caldas. Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system. *Advanced Engineering Informatics*, 22:59–70, 2007.

L. Candy and E.A. Edmonds. Supporting the creative user: a criteria-based approach to interaction design. *Design Studies*, 18(2):185–194, 1997.

S.K. Card, J.D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman Publishers, San Francisco, CA, USA, 1999.

C. Chen. *Information visualization: Beyond the horizon*. Springer-Verlag, London, UK, 2004.

A. Colmerauer. The birth of Prolog. In *The Second ACM-SIGPLAN History of Programming Language Conference*, pages 37–52, 1993.

J. Conklin. *Dialogue mapping: building shared understanding of wicked problems*. John Wiley & Sons, Chichester, UK, 2005.

V.S. Costa, R. Rocha, and L. Damas. The YAP prolog system. *Theory and Practice of Logic Programming*, 12:5–34, 2011.

N. Cross. Designerly ways of knowing. *Design Studies*, 3(4):221–227, 1982.

N. Cross. Styles of learning, designing and computing. *Design Studies*, 6 (3):157–162, 1985.

N. Cross. The nature and nurture of design ability. *Design Studies*, 11(3): 127–140, 1990.

N. Cross. Descriptive models of creative design: application to an example. *Design Studies*, 18(4):427–455, 1997.

N. Cross. Natural intelligence in design. *Design Studies*, 20(1):25–39, 1999.

N. Cross. *Designerly ways of knowing*. Springer-Verlag, London, UK, 2006.

N. Cross. Forty years of design research. *Design Studies*, 28:1–4, 2007a.

N. Cross. From a design science to a design discipline: understanding designerly ways of knowing and thinking. In *Design Research Now*, pages 41–54. Birkhäuser, Basel, CH, 2007b.

N. Cross. *Design thinking: understanding how designers think and work*. Berg, New York, NY, USA, 2011.

CultureSampo. CultureSampo - Historical areas, 2012. Available online: http://www.kulttuurisampo.fi/historiallisetAlueet.shtml. Last accessed on 22 February 2012.

R. Cyganiak and C. Bizer. Pubby: a linked data frontend for SPARQL endpoints, 2012. Available online: http://www4.wiwiss.fu-berlin.de/pubby/. Last accessed on 22 February 2012.

R. Cyganiak and A. Jentzsch. The linking open data cloud diagram, 2011. Available online: http://lod-cloud.net/. Last accessed on 22 February 2012.

J. De Roo. Eye, 2012a. Available online: http://eulersharp.sourceforge.net/2003/03swap/eye-2009.txt. Last accessed on 22 February 2012.

J. De Roo. Euler proof mechanism, 2012b. Available online: http://www.agfa.com/w3c/euler/. Last accessed on 22 February 2012.

D. Di Mascio. Digital reconstruction and analysis of Turchinio's Trabocco: A method of digital reconstruction of a complex structure as a way to improve our knowledge of a cultural heritage artifact. In *Proceedings of the 4th International Conference of the Arab Society for Computer Aided Architectural Design*, pages 177–189, 2009.

D. Di Mascio. Preserving memories with digital media: a methodology for the reconstruction of Castelnuovo Village. In *Proceedings of the 15th International Conference on Computer Aided Architectural Design Research in Asia*, pages 83–92, 2010.

E.Y. Do and M.D. Gross. Thinking with diagrams in architectural design. *Artificial Intelligence Review*, 15:135–149, 2001.

K. Dorst. Design problems and design paradoxes. *Design Issues*, 22(3):4–17, 2006.

K. Dorst. The core of 'design thinking' and its application. *Design Studies*, 32:521–532, 2011.

K. Dorst and N. Cross. Creativity in the design process: co-evolution of problem-solution. *Design Studies*, 22:425–437, 2001.

M. Douglas and B. Isherwood. *The world of goods*. Allen Lane, London, UK, 1979.

H.L. Dreyfus. Intelligence without representation: Merleau-Ponty's critique of mental representation. *Phenomenology and the Cognitive Sciences*, 1:367383, 2002.

C. Eastman. New directions in design cognition: studies of representation and recall. In C. Eastman, W. Newstetter, and M. McCracken, editors, *Design Knowing and Learning: Cognition in Design Education*. Elsevier Science Press, Oxford, UK, first edition, 2001.

C.M. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM handbook: a guide to building information modeling for owners, managers, architects, engineers, contractors, and fabricators*. John Wiley & Sons, Hoboken, NJ, USA, 2008.

C.M. Eastman, J. Lee, Y. Jeong, and J. Lee. Automatic rule-based checking of building designs. *Automation in Construction*, 18:1011–1033, 2009.

S. El-Tawil and V.R. Kamat. Rapid reconnaissance of post-disaster building damage using augmented situational visualization. In *Proceedings of the 17th Analysis and Computation Specialty Conference*, pages 1–10, 2006.

Epic Games, Inc. Game engine technology by Unreal, 2012. Available online: http://www.unrealengine.com/. Last accessed on 22 February 2012.

I. Erbas, M.S. Bittermann, and R. Stouffs. Use of a knowledge model for integrated performance evaluation for housing (re)design towards environmental sustainability: a case study. In *Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAAD-Futures)*, pages 281–295, 2011.

Esperient. Esperient creator - 3ds max to creator workflow. Esperient Whitepaper, 2009. Available online: http://www.esperient.com/-images/Whitepapers/MaxtoCreatorWorkflow.pdf. Last accessed on 20 July 2010.

Esperient. Interactive architectural visualisation with Esperient Creator. Esperient Whitepaper, 2012. Available online: http://www.esperient.-com/images/Whitepapers/Architectural%20Visualization.pdf. Last accessed on 20 July 2010.

European Committee for Standardization (CEN). EN 12354-3: Building acoustics - Estimation of acoustic performance of buildings from the performance of elements - Part 3: Airborne sound insulation against outdoor sound, 2000.

H.R. Fischer. Abductive reasoning as a way of worldmaking. *Foundations of Science*, 6:361–383, 2001.

P.A. Flach and A.C. Kakas. Abductive and inductive reasoning: background and issues. In P.A. Flach and A.C. Kakas, editors, *Abduction and induction: essays on their relation and integration*, pages 1–27. Kluwer Academic Press, Dordrecht, NL, 2000.

H.G. Frankfurt. Peirce's notion of abduction. *The Journal of Philosophy*, 55: 593–597, 1958.

D.D. Gajski and R.H. Kuhn. New VLSI tools. *Computer*, 16(12):11–14, 1983.

M.P. Gallagher, A.C. O'Connor, J.L. Dettbar, and L.T. Gilday. Cost analysis of inadequate interoperability in the U.S. capital facilities industry. Technical Report NIST Report GCR 04-867, National Institute of Standards and Technology (NIST), 2004.

E. Gardner. Reasoning in architecture - about the diagrammatic nature of thinking with real and imagined objects. Master's thesis, Delft University of Technology, Delft, NL, 2009.

F. Gehry. Overview - Gehry Technologies, 2012. Available online: http://-www.gehrytechnologies.com. Last accessed on 22 February 2012.

D. Gentner, B.F. Bowdle, P. Wolff, and C. Boronat. Metaphor is like analogy. In D. Gentner, K.J. Holyoak, and B.N. Kokinov, editors, *The analogical mind: perspectives from cognitive science*. The MIT Press, Cambridge, MA, USA, 2001.

GeoNames. GeoNames, 2012. Available online: http://www.geonames.-org/. Last accessed on 22 February 2012.

S. Gerbino. Tools for the interoperability among CAD systems. In *Proceedings of the XIII ADM - XV INGEGRAF International Conference on Tools and Methods Evolution in Engineering Design*, 2003.

Getty Research Institute. Art & Architecture Thesaurus Online, 2012. Available online: http://www.getty.edu/research/tools/vocabularies/aat/index.html/. Last accessed on 10 May 2012.

N. Glazer. Schools of the minor professions. *Minerva*, 12(3):346–364, 1974.

V. Goel and P. Pirolli. The structure of design problem spaces. *Cognitive Science*, 16:395–429, 1992.

G. Goldschmidt. The dialectics of sketching. *Design Studies*, 4:123–143, 1991.

G. Goldschmidt. On visual design thinking: the vis kids of architecture. *Design Studies*, 15(2):158–174, 1994.

M. Golparvar-Fard, P.M. Feniosky, and S. Savarese. Application of D4AR - A 4-Dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *Journal of Information Technology in Construction*, 14:129–153, 2009.

K. Grace, R. Saunders, and J.S. Gero. Interpretation-driven visual association. In *Proceedings of the Second International Conference on Computational Creativity*, pages 132–134, 2011.

J. Grant and D. Beckett. RDF test cases. W3C recommendation, 2004. Available online: http://www.w3.org/TR/rdf-testcases/. Last accessed on 22 February 2012.

Graphisoft. Communicate your design effectively, 2012. Available online: http://www.graphisoft.com/products/archicad/interoperability/. Last accessed on 22 February 2012.

Graphisoft. GRAPHISOFT virtual building explorer for ArchiCAD, 2012. Available online: http://www.graphisoft.com/products/virtual-building-explorer/. Last accessed on 22 February 2012.

S. Gregory. Design science. In S. Gregory, editor, *The Design Method*, pages 323–330. Butterworth, London, UK, 1966a.

S. Gregory. Conference 'The Design Method'. 1966b.

H. Halpin, P.J. Hayes, J.P. McCusker, D.L. McGuinness, and H.S. Thompson. When owl:sameAs isn't the same: an analysis of identity in linked data. *Lecture Notes in Computer Science: ESWC Part I*, 6496:305–320, 2010.

M. Harney and J. Tredinnick. An interactive tool for the exploration of contextual architecture: case study: 18th century Prior Park, Bath. In *Proceedings of the 27th Conference on education and research in Computer Aided Architectural Design in Europe (eCAADe)*, pages 623–630, 2009.

M. Hennessy. *The semantics of programming languages*. Chichester, UK, 1990.

A. Heylighen. Building memories. *Building Research & Information*, 35:90–100, 2007.

A. Heylighen and H. Neuckermans. DYNAMO: a dynamic architectural memory on-line. *Educational Technology and Society*, 3(2):86–95, 2000.

D.R. Hofstadter. Analogy as the core of cognition. In D. Gentner, K.J. Holyoak, and B.N. Kokinov, editors, *The analogical mind: perspectives from cognitive science*. The MIT Press, Cambridge, MA, USA, 2001.

K.J. Holyoak, D. Gentner, and B.N. Kokinov. Introduction: the place of analogy in cognition. In D. Gentner, K.J. Holyoak, and B.N. Kokinov, editors, *The analogical mind: perspectives from cognitive science*. The MIT Press, Cambridge, MA, USA, 2001.

I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: a semantic web rule language combining OWL and RuleML. W3C member submission, 2004. Available online: http://www.w3.org/Submission/SWRL/. Last accessed on 22 February 2012.

E. Hyvönen, E. Mäkelä, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MUSEUMFINLAND – Finnish museums on the semantic web. *Journal of Web Semantics*, 3:224–241, 2005.

E. Hyvönen, E. Mäkelä, T. Kauppinen, O. Alm, J. Kurki, T. Ruotsalo, K. Seppälä, J. Takala, K. Puputti, H. Kuittinen, K. Viljanen, J. Tuominen, T. Palonen, M. Frosterus, R. Sinkkilä, P. Paakkarinen, J. Laitio, and

K. Nyberg. CultureSampo - Finnish cultural heritage collections on the semantic web 2.0. In *Proceedings of the 1st International Symposium on Digital Humanities for Japanese Arts and Cultures*, 2009a.

E. Hyvönen, E. Mäkelä, T. Kauppinen, O. Alm, J. Kurki, T. Ruotsalo, K. Seppälä, J. Takala, K. Puputti, H. Kuittinen, K. Viljanen, J. Tuominen, T. Palonen, M. Frosterus, R. Sinkkilä, P. Paakkarinen, J. Laitio, and K. Nyberg. CultureSampo: a national publication system of cultural heritage on the semantic web 2.0. In *The semantic web: research and applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 851–856. Springer, 2009b.

International Council of Museums (ICOM). The CIDOC Conceptual Reference Model, 2006. Available online: http://www.cidoc-crm.org/. Last accessed on 22 February 2012.

International Organization for Standardization. ISO 10303-11: Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual, 2004. Available online: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047. Last accessed on 22 February 2012.

Italian Normative UNI. Residential building. Building elements. Classification and terminology, 1981.

Italian Normative UNI. Building. Terminology for users, performances, quality and building process, 1999.

Y.-S. Jeong, C.M. Eastman, R. Sacks, and I. Kaner. Benchmark tests for BIM data exchanges of precast concrete. *Automation in Construction*, 18(4): 469–484, 2009.

J.C. Jones. *Design methods: seeds of human futures*. John Wiley & Sons, New York, NY, USA, first edition, 1970.

J.C. Jones. How my thoughts about design methods have changed during the years. *Design methods and Theories*, 11(1):48–62, 1977.

J.C. Jones and D. Thornley. Conference on design methods: papers presented at the conference on systematic and intuitive methods in engineering, industrial design, architecture and communications. Pergamon Press, 1962.

I. Keough. goBIM: BIM review for the iPhone. In *Proceedings of the 29th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, pages 273–277, 2009.

L. Khemlani. Exploring Second Life and its potential in real life AEC. AECBytes 'Building the Future', 2007. Available online: http://www.-aecbytes.com/buildingthefuture/2007/SecondLife.html. Last accessed on 22 February 2012.

R.D. King, J. Rowland, S.G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Markham, P. Pir, L.N. Soldatova, A. Sparkes, K.E. Whelan, and M. Clare. The automation of science. *Science*, 324:85–89, 2009.

T. Koehler, A. Dieckmann, and P. Russell. An evaluation of contemporary game engines. In *Proceedings of the 26th eCAADe Conference*, pages 743–750, 2008.

B. Kokinov. Analogy is like cognition: dynamic, emergent and context sensitive. In K. Holyoak, D. Gentner, and B. Kokinov, editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*, pages 96–105. NBU Press, Sofia, BG, 1998.

J. Kolko. Abductive thinking and sensemaking: the drivers of design synthesis. *Design Issues*, 26:15–28, 2010.

B. Kraft and M. Nagl. Visual knowledge specification for conceptual design: definition and tool support. *Advanced Engineering Informatics*, 21: 67–83, 2007.

S. Krish. A practical generative design method. *Computer-Aided Design*, 43 (1):88–100, 2011.

T. Kuhn. *The structure of scientific revolutions*. University of Chicago Press, 1962.

G. Lakoff and M. Johnson. The metaphorical structure of the human conceptual system. *Cognitive Science*, 4(2):195–208, 1980.

A. Lau and A. Vande Moere. Towards a model of information aesthetics in information visualization. In *Proceedings of the 11th International Conference on Information Visualization*, 2007.

B. Lawson. Cognitive strategies in architectural design. *Ergonomics*, 22(1): 59–68, 1979.

B. Lawson. *How designers think - the design process demystified*. Architectural Press, Elsevier, Oxford, UK, first edition, 1980.

B. Lawson. 'Fake' and 'real' creativity using computer aided design: some lessons from Herman Hertzberger. In *Proceedings of the 3rd ACM Conference on Creativity & cognition*, pages 174–180, 1999.

B. Lawson. CAD and creativity: does the computer really help? *Leonardo*, 35(3):327–331, 2002.

B. Lawson. *How designers think - the design process demystified*. Architectural Press, Elsevier, Oxford, UK, fourth edition, 2005a.

B. Lawson. Oracles, draughtsmen, and agents: the nature of knowledge and creativity in design and the role of it. *Automation in Construction*, 14: 383–391, 2005b.

Layar. Augmented reality browser: Layar, 2012. Available online: http://-www.layar.com/. Last accessed on 22 February 2012.

Le Corbusier. CIAM 2nd Congress. Frankfurt, DE, 1929.

T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, K. Karstila, K. Reed, S. Richter, and J. Wix. Industry Foundation Classes IFC2x edition 3 technical corrigendum 1, 2012. Available online: http://www.building-smart-tech.org/ifc/IFC2x3/TC1/html/index.htm. Last accessed on 22 February 2012.

MACE Consortium. MACE: metadata for architectural contents in Europe, 2012. Available online: http://portal.mace-project.eu/. Last accessed on 22 February 2012.

M.L. Maher and J. Poon. Modelling design exploration as co-evolution. *Microcomputers in civil engineering*, 11:195–209, 1996.

E. Mäkelä, K. Hypen, and E. Hyvönen. In *The semantic web - International Semantic Web Conference 2011*, volume 7032 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 2011.

E. Mäkelä, E. Hyvönen, and T. Ruotsalo. How to deal with massively heterogeneous cultural heritage data - lessons learned in CultureSampo. *Semantic Web Interoperability, Usability, Applicability*, 3(1), 2012.

F. Manola and E. Miller. RDF Primer. W3C Recommendation, 2004. Available online: http://www.w3.org/TR/rdf-primer/. Last accessed on 22 February 2012.

L. Manovich. *Info-aesthetics*. Bloomsbury Academic, London, UK, 2012.

L.J. March. The logic of design and the question of value. In *The architecture of form*, pages 1–40, Cambridge, MA, 1976. Cambridge University Press.

V. Margolin and R. Buchanan. *The idea of design: a design issues reader*. The MIT Press, Cambridge, MA, USA, first edition, 1995.

S. Marks, J. Windsor, and B. Wünsche. Evaluation of game engines for simulated surgical training. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007.

R.L. Martin. *The opposable mind: how successful leaders win through integrative thinking*. Harvard Business School Press, Boston, MA, USA, 2007.

D.L. McGuinness and F. van Harmelen. OWL 2 Web Ontology Language document overview. W3C recommendation, 2009. Available online: http://www.w3.org/TR/owl2-overview/. Last accessed on 22 February 2012.

W.J. Mitchell, A.S. Inouye, and M.S. Blumenthal, editors. *Beyond productivity: information technology, innovation, and creativity*. National Academic Press, Washington, DC, USA, 2003.

J. Moloney, R. Amor, J. Furness, and B. Moores. Design critique inside a multi-player game engine. In *Proceedings of the CIB W78 Conference on Construction IT Bridging the Distance*, pages 255–262, 2003.

A. Newell. The chess machine: an example of dealing with a complex task by adaptation. In *Proceedings Western Joint Computer Conference*, pages 101–108, 1955.

A. Newell and H.A. Simon. The logic theory machine – a complex information processing system. *IRE Transactions on Information Theory*, 2(3): 61–79, 1956.

A. Newell and H.A. Simon. *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1972.

A. Newell, J.C. Shaw, and H.A. Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, 1959a.

A. Newell, J.C. Shaw, and H.A. Simon. A general problem solving program for a computer. *Computers and Automation*, 8:10–16, 1959b.

A. Newell, J.C. Shaw, and H.A. Simon. The processes of creative thinking. In *Contemporary approaches to creative thinking*, pages 63–119. Atherton Press, New York, NY, USA, 1963.

J. Pallasmaa. *The thinking hand: existential and embodied wisdom in architecture*. John Wiley & Sons, Chichester, UK, 2009.

P. Pauwels. Architectural information modelling : a semantic description framework for historical and theoretical knowledge in architecture. In *Proceedings of the 15th Joint doctoral seminar in Theory and History of Architecture*, 2008.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. Architectural information modelling for virtual heritage application. In *Proceedings of the 14th International Conference on Virtual Systems and Multimedia*, pages 18–23, 2008.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. Semantics-based design: can ontologies help in a preliminary design phase? *Design Principles and Practices: An International Journal*, 3(5):263–276, 2009a.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. Architectural information modelling in construction history. In *Proceedings of the Third International Congress on Construction History*, pages 1139–1146, 2009b.

P. Pauwels, R. Verstraeten, R. De Meyer, and J. Van Campenhout. Architectural information modelling to address limitations of BIM in the design practice. In *Proceedings of the 5th Conference on Information and Knowledge Management in Building (CIB W102 2009) - Deconstructing Babel: sharing global construction knowledge*, pages 15–17, 2009c.

P. Pauwels, R. De Meyer, and J. Van Campenhout. Interoperability for the design and construction industry through semantic web technology. In T. Declerck, M. Granitzer, M. Grzegorzek, M. Romanelli, S.M. Rüger, and M. Sintek, editors, *5th International Conference on Semantic and Digital Media Technologies*, volume 6725 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2010a.

P. Pauwels, R. De Meyer, and J. Van Campenhout. Visualisation of semantic architectural information within a game engine environment. In K. Makanae, N. Yabuki, and K. Kashiyama, editors, *Proceedings of the 10th International conference on Construction Applications of Virtual Reality*, pages 219–228, 2010b.

P. Pauwels, R. De Meyer, and J. Van Campenhout. Extending the design process into the knowledge of the world. In P. Leclercq, A. Heylighen, and G. Martin, editors, *Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures (CAADFutures)*, pages 203–216, 2011a.

P. Pauwels, D. Van Deursen, J. De Roo, T. Van Ackere, R. De Meyer, R. Van de Walle, and J. Van Campenhout. Threedimensional information exchange over the semantic web for the domain of architecture, engineering and construction. *Artificial Intelligence for Engineering, Design and Manufacturing*, 25:317–332, 2011b.

P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout. A semantic rule checking environment for building performance checking. *Automation in Construction*, 20 (5):506–518, 2011c.

P. Pauwels, R. De Meyer, and J. Van Campenhout. A critical evaluation of information system support for design thinking. *Design Issues*, 2012a. in press.

P. Pauwels, P. Present, and T. Strobbe. A pragmatic approach towards software usage in construction projects: the port house in antwerp, belgium. In *Proceedings of the 9th European Conference on Product and Process Modelling*, 2012b. in press.

T. Pazlar and Z. Turk. Interoperability in practice: geometric data exchange using the IFC standard. *Journal of Information Technology in Construction*, 13:362–380, 2008.

C.S. Peirce. *Collected papers of Charles Sanders Peirce. vols. 1-6 (Eds. C. Hartshorne & P. Weiss) (1931-1935), vols. 7-8 (Ed. A.W. Burks) (1958).* Harvard University Press, Cambridge, MA, USA, 1958.

B. Peters and X. De Kestelier. The work of Foster and Partners Specialist Modelling Group. In *The Bridges Conference: Mathematical Connections in Art, Music, and Science*, 2006.

J. Plume and J. Mitchell. Collaborative design using a shared IFC building model - learning from experience. *Automation in Construction*, 16:28–36, 2007.

M. Polanyi. *Personal knowledge: towards a post-critical philosophy*. Routledge and Kegan Paul, London, UK, 1958.

M. Polanyi. *The tacit dimension*. Doubleday, Garden City, NY, USA, first edition, 1967.

J. Poon and M.L. Maher. Co-evolution in design: a case study of the Sydney Opera House. In Y.-T. Liu, J.-Y. Tsou, and J.-H. Hou, editors, *Proceedings of the Second Conference on Computer Aided Architectural Design Research in Asia*, pages 439–448, 1997.

K.R. Popper. *The logic of scientific discovery*. Harper & Row, New York, NY, USA, 1958.

E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C recommendation, 2008. Available online: http://www.w3.org/-TR/rdf-sparql-query/. Last accessed on 22 February 2012.

A.T. Purcell and J.S. Gero. Drawings and the design process. *Design Studies*, 19:389–429, 1998.

H.C. Purchase, N. Andrienko, T.J. Jankun-Kelly, and M. Ward. Theoretical foundations of information visualization. In A. Kerren, J.T. Stasko, J. Fekete, and C. North, editors, *Information visualization: human-centered issues and perspectives*, volume 4950 of *Lecture Notes In Computer Science*, pages 46–64. Springer-Verlag, Heidelberg, DE, 2008.

Quest3D. Quest3D - The 3D engine for VR and simulation, 2012. Available online: http://quest3d.com/. Last accessed on 22 February 2012.

O. Ray. Automated abduction in scientific discovery. In L. Magnani, editor, *Model-Based Reasoning in Science, Technology and Medicine*, pages 103–116. Springer-Verlag, Berlin Heidelberg, DE, 2007.

O. Ray, A. Clare, M. Liakata, L.N. Soldatova, K.E. Whelan, and R.D. King. Towards the automation of scientific method. In *Proceedings of the International Joint Conference on Artificial Intelligence: Workshop on Abductive and Inductive Knowledge Development*, pages 27–33, 2009.

H. Reichenbach. *Experience and prediction: an analysis of the foundations and the structure of knowledge*. The University of Chicago Press, Chicago, IL, USA, 1938.

H. Rittel and M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4:155–169, 1973.

H. Rittel and M. Webber. Planning problems are wicked problems. In N. Cross, editor, *Developments in design methodology*, pages 135–144. John Wiley & Sons, Chichester, UK, 1984.

B.F. Roberson and D.F. Radcliffe. Impact of CAD tools on creative problem solving in engineering design. *Computer-Aided Design*, 41:136–146, 2009.

G. Roberts, A. Evans, A. Dodson, B. Denby, S. Cooper, and R. Hollands. The use of augmented reality, GPS and INS for subsurface data visualisation. In *Proceedings of the XXII International Congress of the FIT*, 2002.

P. Rowe. *Design thinking*. The MIT Press, Cambridge, MA, USA, 1987.

Rule Interchange Format (RIF) Working Group. Implementations - RIF, 2012a. Available online: http://www.w3.org/2005/rules/wiki/-Implementations. Last accessed on 22 February 2012.

Rule Interchange Format (RIF) Working Group. RIF working group, 2012b. Available online: http://www.w3.org/2005/rules/wiki/RIF_Working_-Group. Last accessed on 22 February 2012.

J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

D. Scheck and P. Wilson. *Information modelling: the EXPRESS way*. Oxford University Press, New York, NY, USA, 1994.

E.H. Schein. *Professional education: some new directions*. McGraw-Hill, New York, NY, USA, 1972.

D. Schön. Generative metaphor: A perspective on problem-setting in social policy. In A. Ortony, editor, *Metaphor and Thought*. Cambridge University Press, Cambridge, UK, 1979.

D. Schön. *The reflective practitioner: how professionals think in action*. Temple Smith, London, UK, 1983.

J.J. Shah and M.T. Rogers. Functional requirements and conceptual design of the feature-based modelling system. *Computer-Aided Engineering Journal*, 5:9–15, 1988.

E. Shapiro. Inductive inference of theories from facts. In J.L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. The MIT Press, Cambridge, MA, USA, 1991.

S.C. Shapiro, editor. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, Chichester, UK, fourth edition, 2003.

K. Shea, R. Aish, and M. Gourtovaia. Towards integrated performance-driven generative design tools. *Automation in Construction*, 14(2):253–264, 2005.

D.H. Shin and P.S. Dunston. Evaluation of augmented reality in steel column inspection. *Automation in Construction*, 18(2):118–129, 2009.

ShiVa. ShiVa 3D game engine with development tools, 2012. Available online: http://www.stonetrip.com/. Last accessed on 22 February 2012.

H.A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69(1):99–118, 1955.

H.A. Simon. *The sciences of the artificial*. The MIT Press, Cambridge, MA, USA, first edition, 1969.

H.A. Simon. The structure of ill-structured problems. *Artificial Intelligence*, 4:181–201, 1973.

H.A. Simon. Information processing models of cognition. *Annual Review of Psychology*, 30:363–396, 1979.

H.A. Simon. *The sciences of the artificial*. The MIT Press, Cambridge, MA, USA, second edition, 1996.

J. Simpson and E. Weiner, editors. *Oxford English Dictionary*. Oxford University Press, Oxford, UK, second edition, 1989. Available online: http://www.oed.com/.

Sketchworlds. SketchWorlds: Your real-time 3D interactive virtual world, 2012. Available online: http://www.sketchworlds.com/. Last accessed on 22 February 2012.

R. Spence. *Information Visualization. Design for interaction*. Pearson Education Limited, Essex, UK, second edition, 2006.

M. Stefaner, E. Dalla Vecchia, M. Condotta, M. Wolpers, M. Specht, S. Apelt, and E. Duval. MACE - enriching architectural learning objects for experience multiplication. In E. Duval, R. Klamma, and M. Wolpers, editors, *Second European Conference on Technology Enhanced Learning*, volume 4753 of *Lecture Notes in Computer Science*, pages 322–336. Springer-Verlag, Berlin Heidelberg, DE, 2007.

T. Strobbe, P. Pauwels, R. Verstraeten, and R. De Meyer. Metaheuristics in architecture. In *Sustainable Construction and Design*, pages 190–196, 2011.

L.A. Suchman. *Understanding Computers and Cognition*. Cambridge University Press, Cambridge, MA, USA, 1987.

The Rule Markup Initiative. RuleML, 2012. Available online: http://ruleml.org/. Last accessed on 22 February 2012.

B. Thomas, W. Piekarski, and B. Gunther. Using augmented reality to visualise architecture design in an outdoor environment. Design Computing on the Net, 1999. Available online: http://www.tinmith.net/papers/-thomas-dcnet-1999.pdf. Last accessed on 22 February 2012.

TopQuadrant. Topbraid - Products - Topbraid Composer, 2012. Available online: http://www.topquadrant.com/products/TB_Composer.html. Last accessed on 22 February 2012.

UGent Multimedia Lab. IFC-to-RDF service, 2012a. Available online: http://ninsuna.elis.ugent.be/IfcRDFService/. Last accessed on 22 February 2012.

UGent Multimedia Lab. IFC/RDF SPARQL endpoint, 2012b. Available online: http://ninsuna.elis.ugent.be/SPARQLEndpoint/. Last accessed on 22 February 2012.

Unity Technologies. Unity: Game development tools, 2012. Available online: http://unity3d.com/. Last accessed on 22 January 2012.

T. Van Ackere and J. De Roo. Rules for the conversion of 3d geometry information, 2010. Available online: http://eulersharp.sourceforge.net/-2010/05smml/. Last accessed on 22 February 2012.

T. van Doesburg and C. Van Eesteren. Towards a collective construction ("vers une construction collective"). *De Stijl*, 6, 1924.

F.J. Varela, E. Thompson, and E. Rosch. *The Embodied Mind*. The MIT Press, Cambridge, MA, USA, 1991.

K.H. Veltman. Syntactic and semantic interoperability: new approaches to knowledge and the semantic web. *The New Review of Information Networking*, 7:159–184, 2001.

R. Verstraeten, P. Pauwels, W. Meeus, R. De Meyer, and J. Van Campenhout. Industry foundation classes: a space-based model scheme? In *Proceedings of the 26th eCAADe conference on education and research in computer aided architectural design in Europe*, pages 117–124, 2008.

R. Verstraeten, P. Pauwels, R. De Meyer, J. Van Campenhout, and G. Lateur. IFC-based calculation of the Flemish energy performance standard. In *Proceedings of the 7th European Conference on Product and Process Modelling: eWork and eBusiness in architecture, engineering and construction*, pages 437–443, 2009.

Virtools. 3DVIA Virtools - Dassault Systèmes, 2012. Available online: http://www.3ds.com/products/3dvia/3dvia-virtools/welcome/. Last accessed on 22 February 2012.

W3C. W3C semantic web activity, 2012. Available online: http://www.-w3.org/2001/sw/. Last accessed on 22 February 2012.

T.B. Ward. Structured imagination: the role of category structure in exemplar generation. *Cognitive Psychology*, 27:1–40, 1994.

T.B. Ward. Analogical distance and purpose in creative thought: Mental leaps versus mental hops. In K. Holyoak, D. Gentner, and B. Kokinov, editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*, NBU Series in Cognitive Science. NBU Press, Sofia, BG, 1998.

C. Ware. *Information visualization: Perception for design*. Morgan Kaufman Publishers, San Francisco, CA, USA, second edition, 2004.

A. Webster, S. Feiner, B. MacIntyre, W. Massie, and T. Krueger. Augmented reality in architectural construction, inspection, and renovation. In *Proceedings of the ASCE Third Congress on Computing in Civil Engineering*, pages 17–19, 1996.

Wikitude. Wikitude, 2012. Available online: http://www.wikitude.org/. Last accessed on 22 February 2012.

T. Winograd and F. Flores. *Understanding Computers and Cognition*. Ablex Publishing, Norwood, NJ, USA, 1986.

S. Wong and K. Chan. EvoArch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, 41:649–667, 2009.

B.C. Wünsche, B. Kot, A. Gits, R. Amor, J. Hosking, and J. Grundy. A framework for game engine based visualisations. In *Proceedings of Image and Vision Computing New Zealand*, 2005.

Q.Z. Yang and Y. Zhang. Semantic interoperability in building design: Methods and tools. *Computer-Aided Design*, 38:1099–1112, 2006.

A. Yurchyshyna and A. Zarli. An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction*, 18(8):1084–1098, 2009.

A. Yurchyshyna, C.F. Zucker, N. Le Thanh, C. Lima, and A. Zarli. Towards an ontology-based approach for conformance checking modelling in construction. In *Proceedings of the 24th CIB W78 Conference*, 2007.

X. Zhang, C. Hu, and H. Li. Semantic query on materials data based on mapping MATML to an OWL ontology. *Data Science Journal*, 8:1–17, 2009.

A

Exchange of 3D information in the semantic web

P. Pauwels, D. Van Deursen, J. De Roo, T. Van Ackere, R. De Meyer, R. Van de Walle, and J. Van Campenhout

This is a summary of the work that was published in Pauwels et al. [2011b]. ***Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction, in Artificial Intelligence for Engineering Design, Analysis and Manufacturing 25 (4) : 317 - 332, 2011. http://hdl.handle.net/1854/LU-1141922.*** **For more details, please consult the original article.**

Abstract In Pauwels et al. [2011b], an explanation is given of the interoperability issue that exists in information system support in the AEC domain. The article specifically focuses on the problems in exchanging simple 3D information. Diverse existing approaches for addressing this interoperability issue are presented in Pauwels et al. [2011b]. However, suggested approaches do not appear to provide for an adequate solution, because, in most cases, the translation issue between object syntax and semantics is merely shifted to another level and not really addressed. It is suggested in Pauwels et al. [2011b] that semantic web technologies may well provide for an alternative approach to this issue, because these technologies enable a logic-based description of information with respect for the inherent syn-

tax and semantics of this information. Instead of following an approach in which information stemming from *different* application domains is simply combined, Pauwels et al. [2011b] suggests to look at the capabilities of semantic web technologies to handle the combination of schemas for the description of the *same* 3D information. After a survey of existing information structures for 3D information (files and 3D kernels), and an overview of semantic web technologies, a double test case is documented in Pauwels et al. [2011b] that investigates the feasibility of a rule-based approach towards 3D information exchange. The documentation of this double test case is repeated in the sections below.

A.1 Test case 1: IFC to STL

A.1.1 IFC to IFC/RDF

A simple 3D model was created using a BIM environment, namely Autodesk Revit Architecture 2010, only containing four connected wall objects with a box-like shape. From the BIM environment, an IFC description was exported using the standard available export functionality provided by Autodesk. The resulting IFC description describes the model as a combination of four `IfcWallStandardCase` concepts with both purely geometric (such as position, representation, and so forth) and extra product information (such as material, cost, and so forth) attached. This IFC description is then converted into an IFC/RDF graph by our IFC-to-RDF web service [UGent Multimedia Lab, 2012a] and made accessible through a SPARQL endpoint [UGent Multimedia Lab, 2012b]. The IFC-to-RDF service relies on an OWL ontology for IFC that is designed as an exact reflection of the original EXPRESS schema [Liebich et al., 2012], similar to how it was suggested by Beetz et al. [2009]. This implies an almost identical syntax and semantics, which in turn enables the most optimal conversion process possible with a minimal amount of information loss. Further design and implementation details of the IFC-to-RDF service can be found in Pauwels et al. [2011c].

The geometry of the `IfcWallStandardCase` concepts is described in the resulting IFC/RDF graph through `IfcExtrudedAreaSolid` concepts that are each in turn described by an extruded direction, a depth and a rectangular profile defining the base profile or `swept area`. The positions of the wall objects are described by a series of rotations and translations, all relative to each other, following the overall structure displayed in Fig. A.1.

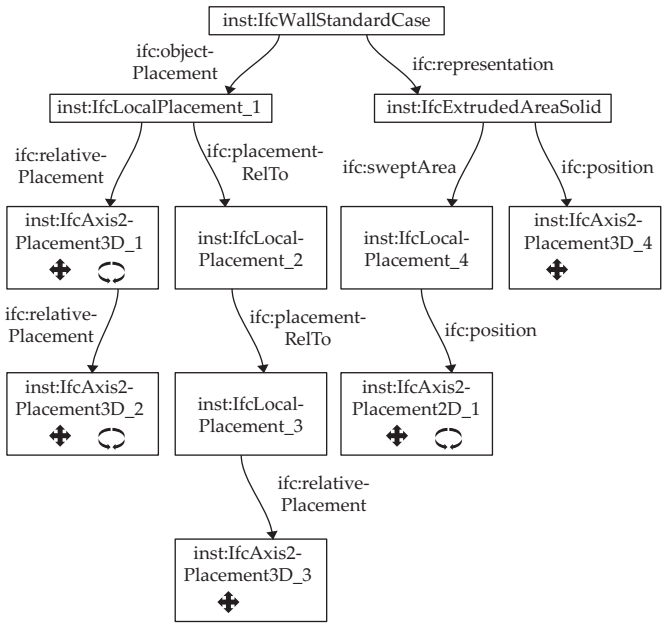


Figure A.1: Overview of how the location of an `IfcWallStandardCase` *is described in the IFC/RDF graph as a series of relative rotations and translations.*

A.1.2 IFC/RDF to X3D/RDF

The resulting IFC/RDF graph may then be converted into an X3D/RDF graph through a set of N3Logic rules. Starting from the IFC/RDF description, we looked for the most appropriate concepts in the X3D specifications to rephrase the IFC/RDF information with. The schema of X3D does not include any notion of a wall object, let alone its product data. This information therefore needs to be disregarded in the conversion process. As 3D primitives were found to be part of the X3D schema, the descriptions of the wall objects are converted into the corresponding descriptions of X3D boxes and linked to the original IFC/RDF descriptions. A rule describing part of this conversion process is displayed in Fig. A.2. Note that for other IFC concepts, such as doors, windows, floors, railings, and so forth, a similar process can be followed and rules can be obtained for the conversion of these IFC objects to the appropriately corresponding X3D objects. An analogous conversion process can presumably be followed for the further conversion into an STL description.

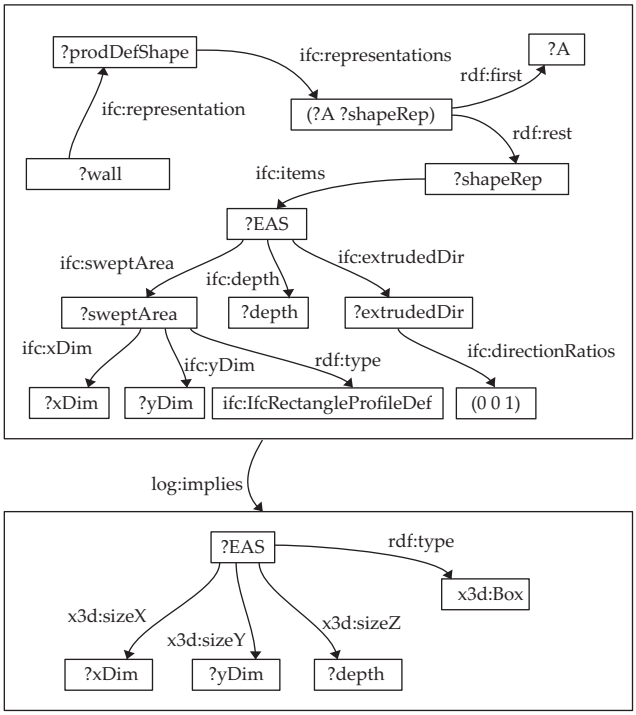


Figure A.2: N3Logic rules creating a box description according to the X3D specifications with its additional dimensions.

The X3D schema describes the position of objects differently from how it is described in IFC. Instead of defining the position of a point through a whole set of relative transformations of both the position and orientation of the object (Fig. A.1), X3D describes the location of an object through a combined translation matrix and rotation matrix relative to the world coordinate system used. These translation and rotation matrices can be inferred from the sequence of relative transformations of both the position and orientation as described in IFC. For each relative transformation, a rotation and translation matrix is calculated iteratively by the backward-chaining reasoning engine using specific N3Logic rules, after which another rule set in N3Logic performs the required matrix calculation on these matrices [Van Ackere and De Roo, 2010]. The eventually resulting translation and rotation matrices are added to the X3D/RDF graph.

A.1.3 X3D/RDF to STL/RDF

A third step comprises the conversion of the X3D/RDF graph into an STL/RDF graph. A similar methodology is followed as in the previous conversion step, starting with a search over the X3D and STL specifications for the appropriate STL concepts to be used for representing the available X3D concepts. Because the STL schema allows the description of geometry only through its triangles, the X3D/RDF box description needs to be converted into a 3D mesh description in STL/RDF.

This mesh description refers to a set of triangles, each in turn described by its normal and vertices. This is again a completely different description compared to the description of a primitive box in the X3D schema. The conversion rules go through a whole range of matrix calculations to obtain these coordinates and convert these in a correct description of all the triangles in the mesh. One of these rules is given for reference in Fig. A.3.

A.1.4 STL/RDF to STL

In a final phase, the STL/RDF graph is converted into an STL file, using a JAVA rewriting application. Because the STL/RDF ontology follows the syntax and semantics of the schema of STL, an easy one-to-one mapping can be followed. Importing the resulting STL file in a 3D modeling application, such as Rhinoceros 3D, indicated a correct conversion of the 3D information. Tests with several other simple models indicated an equally successful process, as long as the concepts with which they were described are incorporated in the rule set.

A.2 Test case 2: STL to IFC

The second test case concentrated on the backward conversion process. Note that in the above test case, the newly inferred information is added to the original information and stored in the same graph as the original IFC/RDF graph. This implies that one knows that the STL/RDF triangles actually represent IFC wall objects or an X3D box object. Also the original product data, such as cost and material information, is still available in an STL context. In this case, no backward conversion process is needed because all the original information describing the IFC wall object according to the schema of IFC, is still present and can be used. This is already an important improvement over the existing approaches toward 3D information exchange in the AEC domain, as these approaches almost always lose or distort this information during the conversion processes.

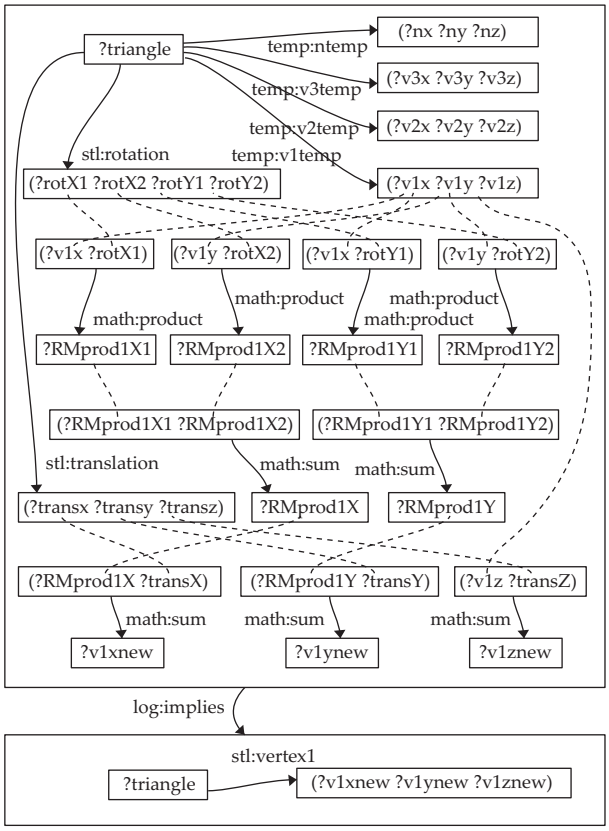


Figure A.3: N3Logic rules describing the recalculation of the normal vector coordinates and of the vertex coordinates of one triangle in the STL mesh.

However, if only the STL/RDF graph is available, no explicit statement exists saying that part of the STL/RDF graph, for instance, represents an IFC wall object or an X3D box object. This second test case investigates whether semantic web technologies could also provide help in this situation. The original test model was therefore remodeled inside a Rhinoceros 3D modeling environment, and exported into an STL format. This was then converted into a semantic STL/RDF graph using the same JAVA application previously used. From this STL/RDF graph we tried to describe rules that can infer that this set of STL triangles actually represents a set of four IFC wall objects or four X3D box objects. The original description is thus semantically enriched using these rules.

These rules explicitly describe interpretations, or presumptions based on specific conditions. The description of these rules is therefore crucial and should be handled with extreme caution and care. It might seem correct to infer that a geometrically closed object bounded by three pairs of parallel faces, can be considered a box object or an `IfcExtrudedAreaSolid` based on a rectangular profile. Inferring that a box with a certain height, width, and depth actually represents an `IfcWallStandardCase` concept is quite another matter. This requires contextual information if a sufficiently reliable result is needed, which is often not available in the original 3D model description and thus needs to be added by a human user.

We nevertheless proceeded with the test case to see how far it goes. Starting from the STL/RDF graph, we went through the following steps for adding extra semantics based on the geometric representation solely.

1. extending the geometric information (such as identification of edges, adjacent triangles, and so forth);

2. geometry recognition: infer the attributes required to describe the geometry according to a specific schema (such as depth and width attributes); and

3. feature recognition: infer extra information, such as identification of wall objects and door objects.

A.2.1 Extending geometric information in the STL graph

The STL/RDF graph describes 3D geometry through a set of unrelated triangles, described uniquely by the coordinates of the vertices of these triangles. The STL/RDF graph does not describe which triangles are neighboring, let alone over which edge they might be neighboring; neither is it known whether some of the edges are parallel to each other, and so forth. This information, however, may be inferred from the explicit STL/RDF graph. This information is thus implicitly present in the STL/RDF graph and may be retrieved using a set of N3Logic rules, similar to how it was done in the first test case.

An example of how one may infer if a triangle has a common edge with another triangle, including naming this particular common edge, is given in Fig. A.4. The triangles that were previously identified as possibly adjacent triangles are checked and identified as neighboring triangles if they have two vertices in common. In addition, the common edge is explicitly named through its two vertices and two adjacent triangles. In a similar

fashion, one can easily describe rules able to infer whether faces are parallel to each other, for instance [Van Ackere and De Roo, 2010].

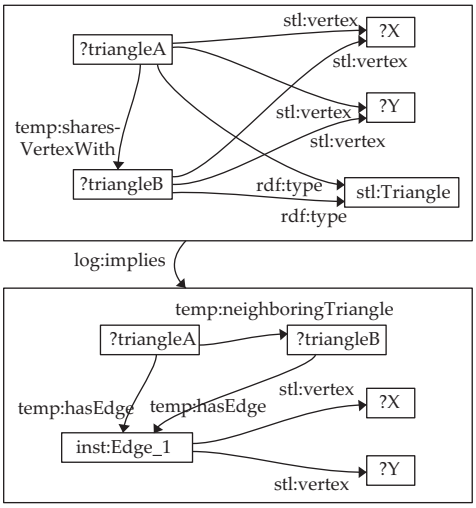


Figure A.4: N3Logic rules for inferring which edges are neighboring.

Once all adjacency information is added, this information can be used to derive separate objects described by the triangles. In our rules, we defined such an object as a set of triangles each having at least one neighboring triangle [Van Ackere and De Roo, 2010]. Using these rules, the EYE inference engine loops through all the triangles and finds such objects instantaneously. This process repeats until each object is found and described through its triangles. As one can conclude from the definition, these objects are both closed and open meshes of triangles. The explicit distinction between these objects is nevertheless possible using additional rules that identify objects of which the triangles all have exactly three adjacent edges or neighboring triangles [Van Ackere and De Roo, 2010].

A rule set similar to the one used to identify objects, allows the identification of separate faces in these objects. A face is in this regard considered as a set of triangles all having at least one neighboring triangle within the set of triangles identifying an object *and* having the same normal vector or orientation as this neighboring triangle. Based on the information available at this point in the RDF graph, also extra information may be inferred about the faces in the model. Each face has the same normal vector as the ones linked to the triangles that are part of these faces. An appropriate interpretation of these vectors, for instance, allows the identification of

parallel faces, similar to how parallel triangles were identified [Van Ackere and De Roo, 2010].

A.2.2 Geometry recognition

Using the extended geometric information derived in the previous steps, one may describe rules that enable the recognition of 3D primitives. In this test case, these geometry recognition rules were developed only for a box primitive. A 3D object in the STL/RDF graph is considered a box primitive if the object consists of exactly three pairs of parallel faces. The lengthy rule set used can be found in Van Ackere and De Roo [2010]).

Because the edges of the faces of this box differ from the edges of the STL triangles, the respective length of each of these edges cannot be inferred directly. It requires the explicit identification of these face edges, linked to the coordinates of the triangles they consist of. The length of each of the edges of this box object can be inferred as the sum of the lengths of all triangle edges that are part of the face edge. This inference requires a lengthy rule set that is not described here for reasons of brevity. One of the rules for inferring the eventual dimension along the x-axis is displayed in Fig. A.5. At this time, we were only able to infer this information for boxes oriented according to the world coordinate system. More complex rules are normally able to infer this information for other box objects as well.

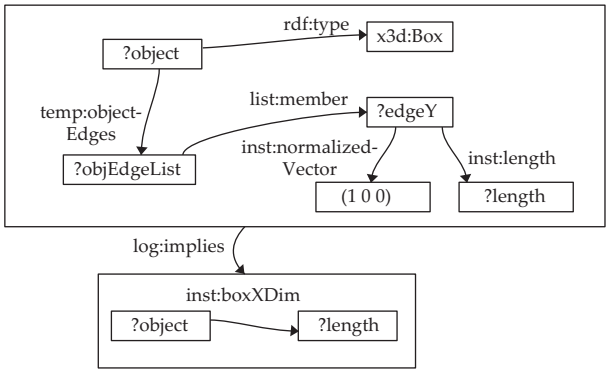


Figure A.5: N3Logic rules for inferring the dimension of a box along the x-axis.

From this information, one may easily infer an RDF graph following the X3D schema. However, the elaborate set of rules that resulted from this test case for the quite simple inference that this 3D model actually represents a box primitive should be taken into account when considering the

recognition or identification of more complex geometry through semantic web technologies. Note also, on the other hand, that once the above rules are available, a lot of implicit information is available automatically for any STL/RDF graph, which can be reused or extended as wished.

A.2.3 Feature recognition

The inference that the box described through the X3D/RDF graph inferred through the rules documented above actually represents a wall object seems a more challenging inference. Not a lot of information in the X3D/RDF and STL/RDF graphs at this stage of the inference process indicates that the model described is even part of a building, let alone that it represents a wall object. Only the information on the dimensions of the 3D objects might give an indication that the object described actually is a wall object. One might decide, for instance, that any box object that is approximately two meters high, 30cm wide, and more than one meter long is a wall object. In addition, one can define upper limits for these dimensions as well. This may enable the distinction between wall entities and other building entities, such as doors, columns, windows, floors, and so forth. Note, however, that any building entity that differs from its description in the rules, cannot be recognized. For instance, a wall object wider than the indicated 30cm or shorter than one meter,will not be recognized as a wall object. In addition, one has to be aware that blindly following this kind of assumptions can undoubtedly lead to very strange and incorrect results.

This inference may be enhanced significantly with the explicit confirmation of certain assumptions. An assumption that might easily be provided by the user, is that the object in question is part of an architectural model. Additional assumptions that may be confirmed explicitly by the user include the definition of a vertical orientation, a connection to a horizontal floor or ceiling element, explicit material information, and so forth. A user may confirm, for instance, that no wall object wider than 30cm or shorter than one meter is present.

Figure A.6 shows how a rule may rely on this kind of information to enable recognition of a wall feature in the available RDF graph. This rule may be extended with more criteria in order to detect more specific types of walls. Once walls are recognized in this test case, enough information should be available in the RDF graph to convert it into a representative IFC file.

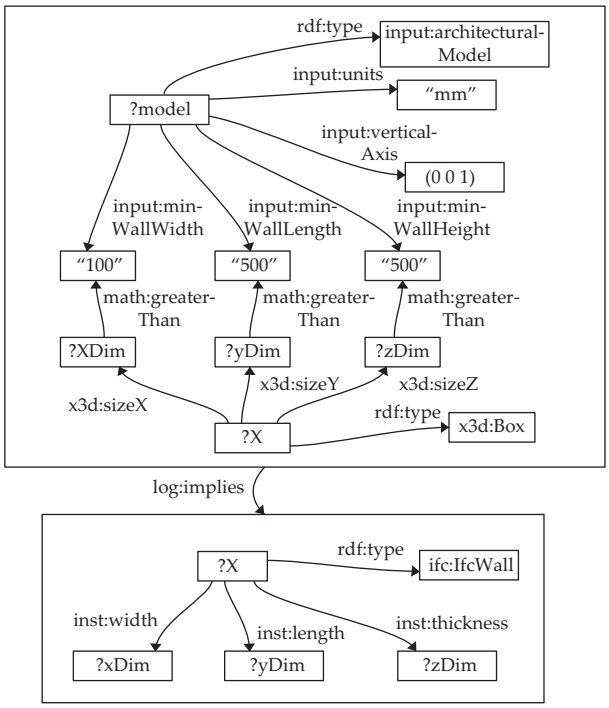


Figure A.6: N3Logic rules describing how one may infer whether or not an X3D box represents a wall object.

B

Modeling architectural information in the semantic web

P. Pauwels, D. Di Mascio, R. De Meyer, and J. Van Campenhout

This is a summary of a research project about the digitization of the Book Tower in Ghent (Belgium): ***About the interpretation of virtual heritage artifacts: case study of the Book Tower in Ghent, Belgium.*** **This work has not been published.**

Abstract The research project about the digitization of the Book Tower in Ghent (Belgium) targeted the investigation of the interoperability issue and the functionality mismatch issue in the context of cultural heritage. In this project, the Book Tower was chosen as our 'cultural heritage artifact'. We relied on a BIM environment, semantic web technologies, and game engines to model information about the building and make it accessible to end users. It was continuously evaluated to what extent the two outlined issues were at play in the documentation process and in the products that were realized. The sections below do not explicitly focus on this evaluation. Instead, only a brief overview is given of the documentation process.

B.1 Learning phase

In order to describe, manage and analyze a built heritage project appropriately, it is necessary to understand it, so that information on the project can eventually be organized 'correctly'. In this initial learning phase, one goes through all kinds of relevant information sources to construct a personal understanding or a mental model of the building. For instance, through a critical literature review one can investigate what data and information (historical, geometrical, structural, spatial, and so forth) should be used for documenting the particular aspects of the cultural heritage artifacts at hand, and how this information can be organized. In the case of the Book Tower, our survey consisted of the elements enumerated below.

1. an on-site survey which was documented with a series of sketches, photographs and specific measurements
2. a literature review of historical documents describing the construction of the tower
3. discussions with several of the people familiar with the tower
4. existing 2D CAD drawings made available by the responsible university service

Notwithstanding the amount of structure and objectivity one puts in working with the documents resulting from this survey, it has to be clear that the way in which these documents are interpreted by us shapes how we understand the building and subsequently document it. For instance, the images shown in Fig. B.1 gave us an insight in the construction of two specific parts of the building, thereby influencing our assumptions and understanding for several other features of the building. This does not necessarily imply that the structure in the rest of the building is the same, although we are documenting it as such anyway.

B.2 Digital reconstruction phase

Relying on the information obtained by actively going through the learning phase and making the necessary interpretations, we further analyzed and disassembled the historical building following a 'reverse engineering' approach. From this analysis, we digitally reconstructed the building in a building information modeling (BIM) environment, namely Revit Architecture (Fig. B.2). This is a 3D modeling environment which allows one to describe the building through semantically rich elements, such as wall

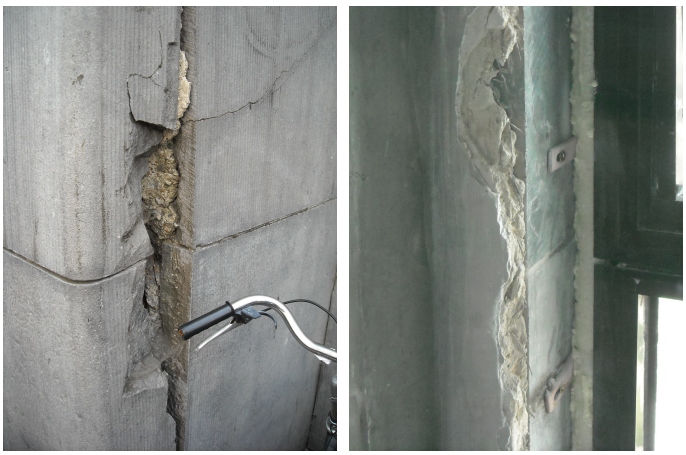


Figure B.1: (1) The inner structure of the outer walls of the Book Tower (left) and the connection of window elements to the outer walls of the building (right).

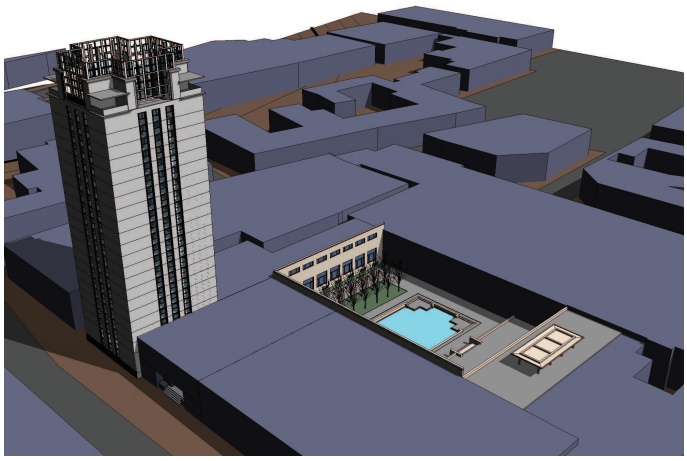


Figure B.2: The 3D BIM model as a bearer of all the building information of the cultural heritage artifact.

types, window types, floor types, and so forth [Eastman et al., 2008]. In this digital reconstruction process, we relied on methods already experienced on other occasions [Di Mascio, 2009, Pauwels, 2008]. It can be compared to redesigning a real building in order to analyze its structure and functionality, resulting in a 3D model capturing this understanding.

Important for this digital reconstruction phase is the ontology followed for describing the information. When using a specific BIM environment, such as Revit Architecture, one is confined to the ontology used in the application logic of this 3D modeling application. Since this digital reconstruction phase goes hand in hand with continuous evaluation, analysis and interpretation, this choice for a specific ontology has its consequences on how we interpret and describe the information. For instance, Fig. B.3 gives an indication of how we modeled the outer walls of the tower, which was not completely identical to how it was built in the 1930s. We decided to separate the inner side of the wall ('wall 1') and the outer side of the wall ('wall 2'), because the two walls were also constructed as separate walls in reality, with the outer wall cast in concrete together with the column structure, and the inner wall as a later addition. So, from the cultural heritage perspective, this appears more correct. However, since Revit Architecture allows windows to be only in one wall at the same time, we decided to make a separate, 'third' wall just for the window. This third 'wall' represents the stone blocks that are placed at the sides of the windows. This is not the best representation for how this is constructed, but with the information and tools available at this point of time, however, this appears as the most optimal choice.

B.3 Analysis phase

The original idea was to construct a schedule describing all types of elements, dimensions, material characteristics, and so forth, that are available as a result of the digital reconstruction phase, so that we could analyze this for completeness and correctness. This phase would be the preparation phase for the next step: describing all additional information that did not fit in the previously used schema or ontology, namely that of the BIM environment, in a separate semantic structure. Using the appropriate semantic web technologies [Berners-Lee et al., 2001, Manola and Miller, 2004, McGuinness and van Harmelen, 2009], it would then be possible to add this missing extra information and further enrich the information model of the cultural heritage artifact to an appropriate level (Fig. B.4, left).

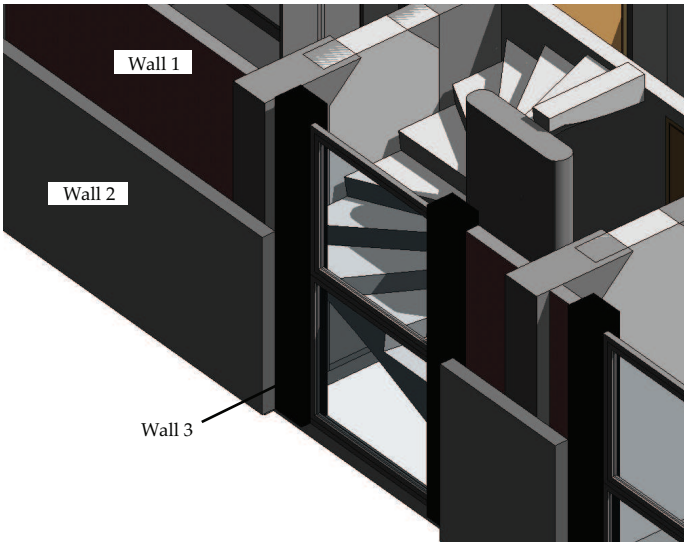


Figure B.3: Part of the 3D model showing how the ontology of the BIM modeling environment in some sense confines you to modeling into a certain way.

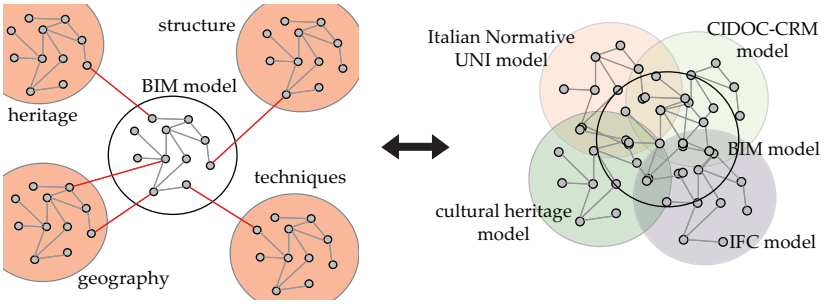


Figure B.4: Two perspectives on the web of information that describes a cultural heritage artifact: distinct domains of information (left) versus overlapping domains of information (right).

We noticed, however, that only one description of the building information is available as a result of the digital reconstruction phase, namely the description corresponding to this one specific ontology (Revit Architecture). There are of course diverse other schemas on which we could rely as well. For instance, we could use one of the several building standards, such as the Italian UNI standard [Italian Normative UNI, 1981, 1999], or the European metadata schema for architectural contents (MACE) [Stefaner et al., 2007], but there are also many other schemas or ontologies which one could use to describe information about the building, such as the CIDOC-CRM ontology specifically designed for cultural heritage information [International Council of Museums (ICOM), 2006]. The information described in these schemas does not really *extend* the original information model, but rather provides an alternative *parallel* description schema that is very hard to relate explicitly to the information model already available. Although this linking together of diverse parallel information models might appear feasible in a theoretical schema, setting up the connections between the diverse schemas proves to be very delicate in practice (Fig. B.4, right). This is mainly because elements and concepts are understood and described in a slightly different way, so that one can not just say '*X owl:sameAs Y*' [Halpin et al., 2010].

B.4 Semantic enrichment phase

Since our main objective was to test how far technology goes, we nevertheless tried to follow the schema in Fig. B.4 (right) and relate diverse parallel information models together using semantic web technologies [Berners-Lee et al., 2001, Manola and Miller, 2004, McGuinness and van Harmelen, 2009], with each of the models describing a different interpretation of the same cultural heritage artifact. Before we could deploy these technologies, we first had to 'translate' the information described in the BIM environment into the language used by semantic web technologies, which are essentially Resource Description Framework (RDF) graphs [Manola and Miller, 2004]. This translation was done in two steps, thereby relying on tools we developed earlier [Pauwels et al., 2011c]. First, the information model was exported in the Industry Foundation Classes (IFC) schema [Liebich et al., 2012], which is a standard recently developed in construction industry for describing building information. Second, this IFC model was converted into an IFC/RDF graph, thereby using our IFC-to-RDF converter service [UGent Multimedia Lab, 2012a]. At this point, an RDF graph is available, describing all the information previously added into the information model using the Revit Architecture application.

As we are now able to use semantic web technologies, we are now able to describe all other schemas, including the Italian UNI standard [Italian Normative UNI, 1981, 1999], the MACE metadata schema [Stefaner et al., 2007], or the CIDOC-CRM ontology [International Council of Museums (ICOM), 2006], in Web Ontology Language (OWL) ontologies [McGuinness and van Harmelen, 2009], enabling us to describe the cultural heritage artifact also according to these schemas and link these descriptions to the original (IFC) description. We thus described some of these 'additional interpretations' and added them to the semantic web structure, explicitly linked to the building information previously described and available through the IFC/RDF graph. A small snap-shot of this semantic structure is shown in Fig. B.5.

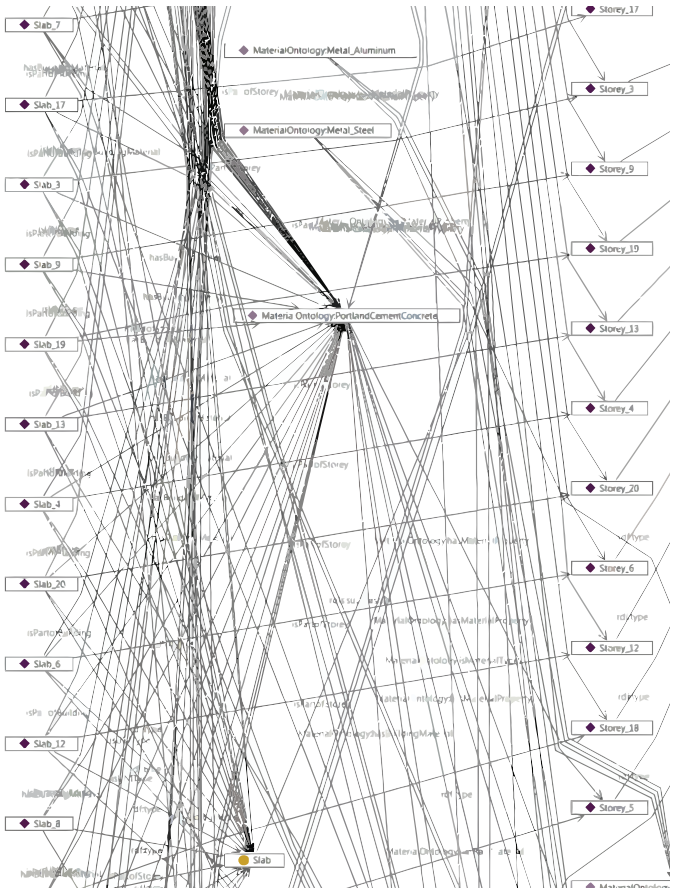


Figure B.5: Information about the Book Tower in Ghent described according to diverse ontologies and linked together as good as possible using semantic web technologies.

B.5 Integration phase

A 3D building model resulted from the four previous phases, giving access to one central model containing all building information according to the IFC ontology. This information is further connected to several parallel descriptions of the cultural heritage artifact in question. Since all this information is stored in a not so intuitive online data repository, a need arises to combine all this information and make it accessible through a user-friendly interface. In this regard, we decided to test the game engines Unity [Unity Technologies, 2012] and Unreal [Epic Games, Inc, 2012], since this enables one to present the cultural heritage artifact in an interactive and intuitive 3D walk-through environment [Di Mascio, 2010, Harney and Tredinnick, 2009]. Both game engines allow to a certain extent to import the 3D model elaborated in Revit Architecture and to add interactive elements, such as a real-time first-person camera, a solar system and a 3D environment (Fig. B.6, left).

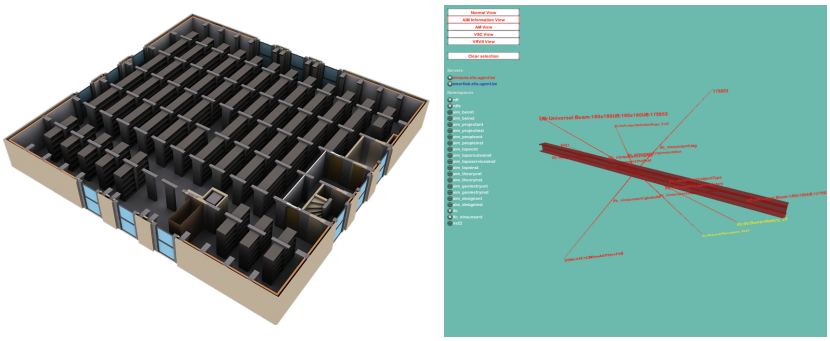


Figure B.6: Visualization of one of the floors of the building model in the Unreal engine (left) and the link between the selected item in the Unity game engine interface and the information described in the semantic web (right).

In Unity, we have implemented some additional scripts using the Unity API, thereby mainly adding the user interface itself and the functionality to select elements and access the information about these elements available on the online server(s) (Fig. B.6, right). The main conclusions made from how people use this interface to learn about the cultural heritage behind this building, is that they learn various aspects about the buildings, but most of the things learned are not exactly what is present in the semantic web graph underlying the 3D building model. Instead mainly *additional* interpretations are 'triggered' by the view and by the information that is displayed in the interface. Users tend to just walk around in the building, select what attracts their attention most, which corresponds often to their

interests or backgrounds, and learn *new* things almost randomly. It almost appears that the interface and the narrative through which the building model and the information is communicated to the user is more important than the information itself.

C

Semantic building performance checking

P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout

This is a summary of the work that was published in Pauwels et al. [2011c]. ***A semantic rule checking environment for building performance checking, in Automation in Construction 20 (5) : 506 518, 2011. http://-hdl.handle.net/1854/LU-1077604.*** **For more details, please consult the original article.**

Abstract In Pauwels et al. [2011c], an analysis is made of the possibilities that could be generated in building performance checking by relying on semantic web technologies. A comparison is made between the usage of RDF and IFC as a language for information exchange. The improvements generated by deploying semantic web languages are briefly discussed in Pauwels et al. [2011c], after which a concrete implementation approach is presented for a semantic rule checking environment for building design and construction. An implemented test case for acoustic performance checking illustrates the improvements of such an environment compared to traditionally deployed approaches in rule checking. In the sections below, both the suggested implementation approach and the implemented test case are presented.

C.1 A semantic rule checking environment for building performance checking

Starting from the outlined possibilities of combining semantic web technology with IFC, we conceived a semantic rule checking environment for construction industry, thereby investigating how the rule checking process may be improved with a dedicated rule language and rule engine. This environment was conceived and developed as a part of the architectural information modeling (AIM) framework previously presented in Pauwels et al. [2009a]. We solely document the rule checking environment as the discussion of the AIM framework falls outside the scope of this paper. The concluding test case will provide clear examples of the conceptual methodology outlined in this section.

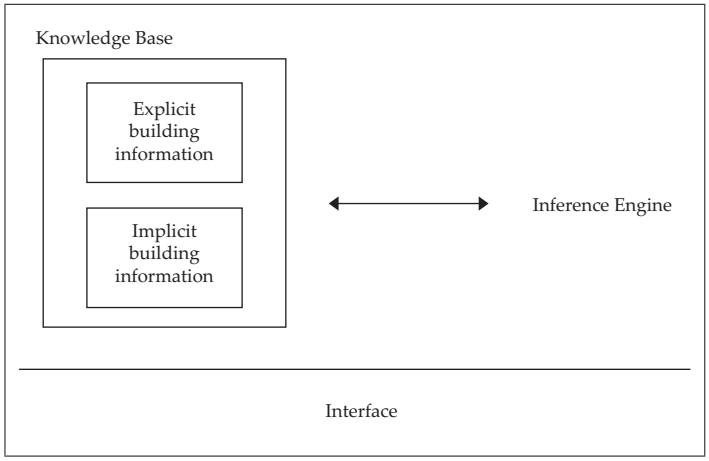


Figure C.1: Conceptual overview of the rule checking environment proposed in this paper.

In our discussion we first document how a knowledge base can be conceived, containing both explicit building information (facts) in RDF graphs and implicit building information (rules) in N3Logic. We present how an inference engine is able to combine information in this knowledge base to retrieve the information desired. An interface completes the rule checking environment for construction industry (Fig. C.1). We point out improvements over existing rule checking engines for construction industry in our brief overview.

C.1.1 Explicit building information: RDF graphs

Several vocabularies are currently being constructed as OWL ontologies, thereby enabling the description of building information in an RDF graph using structured vocabularies. We will only mention two of our OWL ontologies here, namely an IFC ontology and a construction element ontology. Other ontologies focus on information about architectural theory, design, actors, geographical locations, and so forth.

The IFC ontology is largely similar to the ontology discussed in Beetz et al. [2009]. The EXPRESS schema was transformed into an OWL ontology. In this transformation process, every EXPRESS element that has a direct equivalent in OWL is mapped onto this equivalent. For each ENTITY element in EXPRESS a corresponding OWL class is generated, EXPRESS attributes are converted into the appropriate OWL properties, and so forth. Having the eventual IFC ontology at our disposal, we are able to generate IFC/RDF instances as part of the explicit information in our knowledge base. We are currently relying on existing BIM applications to generate an IFC model that can subsequently be converted into an equivalent IFC/RDF graph using our in-house IFC-to-RDF converter [UGent Multimedia Lab, 2012a]. A part of an IFC/RDF graph is displayed in Fig. C.2.

The construction element ontology is constructed by applying a D2R server mapping [Bizer and Cyganiak, 2010] onto an existing database of construction elements. This relational database collects test information about construction elements produced in Belgium. This test information mainly concerns acoustic and thermal characteristics, such as sound reduction indexes, thermal resistances, and so forth. On top of this database, RDF graphs describing construction elements (CONSTR/RDF instances) are generated real-time using the D2R mapping. The resulting CONSTR/RDF instances are in this way available as part of the explicit information in the knowledge base.

References between the different graphs of the knowledge base, under which the IFC/RDF graph and the CONSTR/RDF graph, can be created using explicit RDF statements. This allows creating a building information graph containing both IFC/RDF information and CONSTR/RDF information for instance. Figure C.2 displays how an instance in an IFC/RDF graph can be linked to an instance of the CONSTR/RDF graph. The instances in the building information graph can similarly be linked to graphs containing architectural history, geographical or design information, thereby resulting in a far richer information model than the IFC schema can provide.

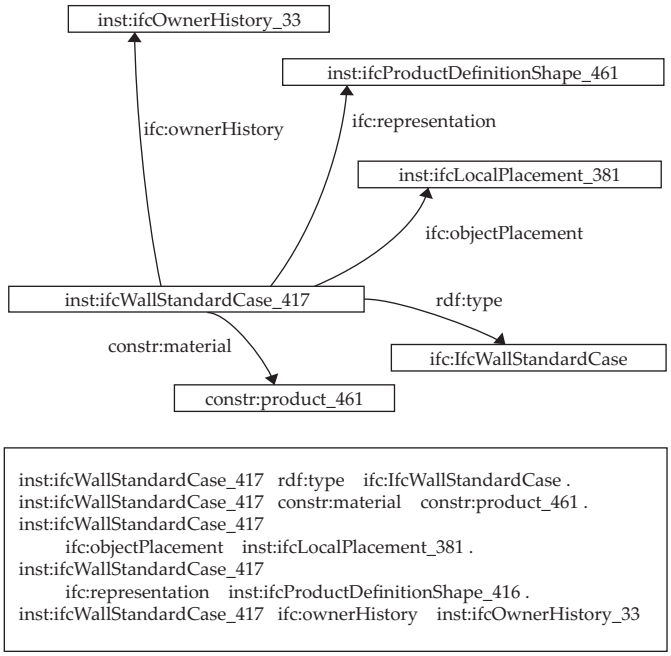


Figure C.2: Linking instances of an IFC/RDF graph to the appropriate construction element information available in a CONSTR/RDF graph.

C.1.2 Implicit building information: N3Logic rules

Parallel to the explicit building information, a range of rule sets is developed in N3Logic and added to the knowledge base. They enable the description of information that is implicitly available in the building model. Every rule set contains rules that can be applied to RDF graphs in order to infer this implicit information. They are thus closely related to the ontologies deployed in describing these RDF graphs. We briefly discuss how these rules are described and how they are related to the explicit building information available in the knowledge base, thereby making the comparison with existing approaches in the rule checking process as they were outlined in Eastman et al. [2009].

Similar to how they are described in natural language, rules tend to deploy their own vocabulary, consisting of concepts that apply specifically to the rule set at hand and that are not available in the vocabularies used to describe other information, such as the information in a building information graph. Geometric and material-based concepts, for instance, do not

map directly on the concepts deployed in an acoustic or thermal standard rule set. We therefore consider it useful to create a separate rule set vocabulary using OWL to describe the concepts deployed within each rule set. One can easily see how the rule displayed in Fig. C.3, for instance, deploys only concepts stemming from its rule set vocabulary in OWL (that is, `ruleOnt:`).

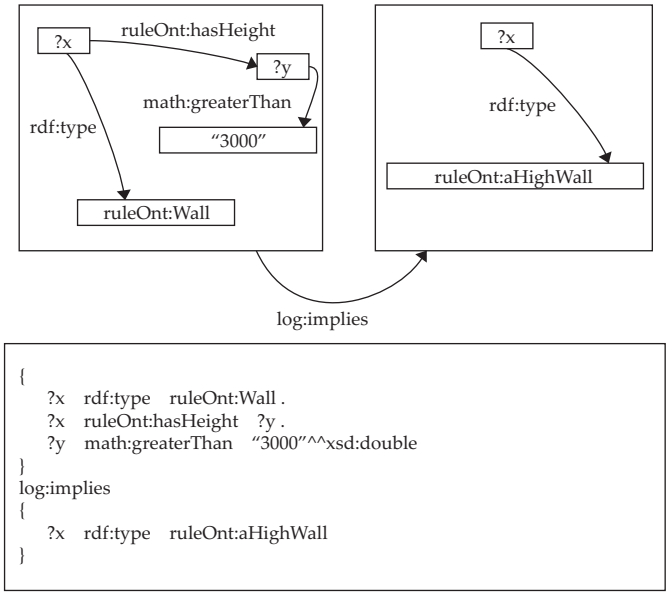


Figure C.3: Example of a rule described using its own rule ontology in OWL.

An explicit relation is needed between the different descriptions of both the explicit building information and the implicit building information. Otherwise, none of the rules described will be able to infer any information from the explicit RDF graph of the building model. When deploying semantic web technology, however, this is fairly easily dealt with. A separate 'conversion rule set' can explicitly link the ontology structure used in the RDF graph to the ontology structure of a rule set. When one tries to infer information with an acoustic performance rule set for instance, this conversion rule set explicitly describes how the running inference engine can interpret the information described in the inputted RDF graph containing building information.

This approach guarantees that both the explicit information contained in the building information graph and the implicit information contained

in the rule set remain manageable and allow changes. When, for instance, the structure of the building information graph changes, it is only necessary to update the conversion rule sets without having to deal with the logic of the rule set itself. An overview of the resulting knowledge base that is to be addressed by the semantic rule checking environment, is shown in Fig. C.4.

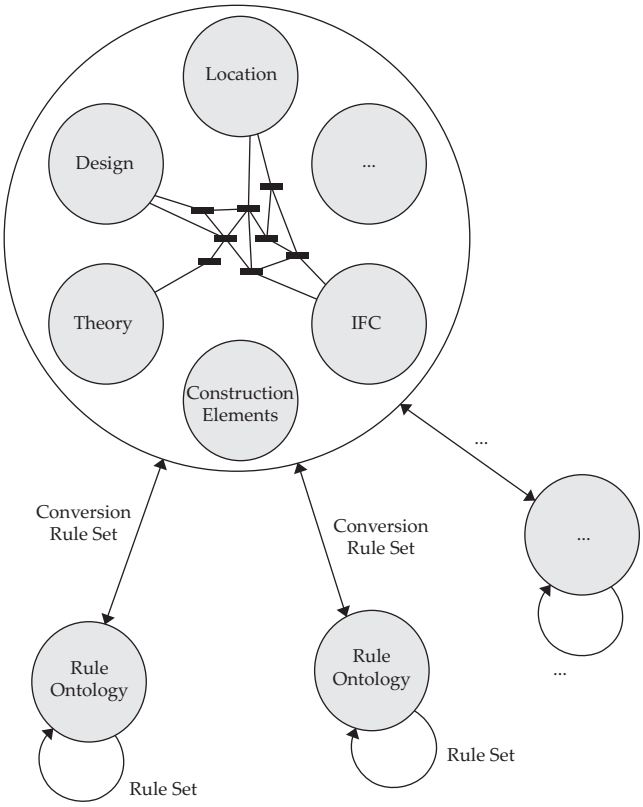


Figure C.4: Rule sets linked to the core knowledge base deploy a separate ontology definition and a conversion rule set.

As can be seen in Fig. C.4, each rule set can thus be considered as a separate addition of implicit information, each time for a very specific domain of knowledge. Since they rely on the same central information base, these rule sets are nevertheless related and thus require a sound organization or structure. One could, for instance, organize these rule sets hierarchically (for instance, construction industry >structural - performance - ar-

chitectural >...), in distinct groups according to geographical parameters, or maybe completely intertwined at diverse levels. How this relation is organized, requires further attention in future design and implementation phases, as such a decision is hard to make in this early stage of research.

C.2 Test case: the acoustic performance regulations

C.2.1 The setup of the test case

A test case was set up for the investigation of how the configuration given above may improve existing rule checking processes in architectural design and construction. As a subject we chose the newly emerging acoustic performance regulations both on a European level as on a national level in Belgium, assuming that the methodology deployed can be extended to other calculation and simulation processes as well. Note that we merely aim at documenting a proof of concept, and not a fully implemented and widely usable application. We merely show the basic concepts in order to make clear to what extent this may form a valid alternative to existing approaches.

C.2.2 European and national standards

March 2000 was the publication date of the EN12354 European standard [European Committee for Standardization (CEN), 2000] for the estimation of the acoustic performance of buildings from the performance of elements. This standard consists of several parts, specifying the estimation process, for instance, for the airborne sound insulation against outdoor sound, or for the transmission of indoor sound to the outside. We solely addressed the third part of the standard in this test case, namely 'EN 12354-3:2000 : Airborne sound insulation against outdoor sound' [European Committee for Standardization (CEN), 2000].

The EN 12354-3:2000 standard documents the way in which the relevant quantities for expressing an acoustic building performance can be both derived from quantities measured on-site or calculated from a given calculation model and more basic rules of thumb. Figure C.5, for instance, shows how to calculate the sound power ratio of radiated sound power by a façade element i due to direct transmission of sound incident on this element, relative to incident sound power on the total façade [European Committee for Standardization (CEN), 2000].

4.2 Determination of direct transmission from acoustic data on elements

All elements of the façade shall be included in the calculation. The sound power ratio is calculated according to the following, where the distinction between small and other elements is in accordance with EN 20140-10.

4.2.1 Small elements

$$\tau_{e,i} = \frac{A_0}{S} 10^{-D_{n,e,i}/10}$$

$$A_0 = 10\ m^2$$

where in the input data

$D_{n,e,i}$ is the element normalized sound level difference of small element i, in decibels ;

S is the total area of the façade as seen from the inside (i.e. the sum of the area of all elements), in square metres.

4.2.2 Other elements

$$\tau_{e,i} = \frac{S_i}{S} 10^{-R_i/10}$$

where in the input data

R_i is the sound reduction index of element i, in decibels ;

S_i is the area of element i, in square metres.

Figure C.5: The calculation procedure for several acoustic performance properties of buildings is given in European Committee for Standardization (CEN) [2000].

As the European standard only provides a means to derive the required quantities for expressing an acoustic building performance, it leaves out the process of evaluating these quantities resulting in a 'pass', 'fail', 'warning' or 'unknown' message for building elements or for the building itself. This part of the evaluation process is to be addressed by the national standards organizations of the involved countries in Europe [European Committee for Standardization (CEN), 2000].

In response to the European standard, the National Bureau for Normalization (NBN) in Belgium published the national regulation NBN S

01-400-1 in January 2008 [Bureau voor normalisatie (Commissie: Geluidsleer - algemeen), 2008]. This regulation specifies the acoustic performance requirements to be met in the construction of new dwellings, regarding the insulation against airborne, contact and outdoor sound. The acoustic performance level against outdoor sound hereby specifies explicit references to the acoustic quantities that can be obtained from the European EN 12354-3:2000 standard. An extract of NBN S 01-400-1:2008 given in Fig. C.6 shows, for instance, the conditions that have to be met in order for a building to reach a normal (Dutch: 'normaal akoestisch comfort') or an increased (Dutch: 'verhoogd akoestisch comfort') acoustic performance level.

Normaal akoestisch comfort	Verhoogd akoestisch comfort
$D_{Atr} \geq L_A$ - 34 + m dB (1) *en* $D_{Atr} \geq$ 26 dB	$D_{Atr} \geq L_A$ - 30 + m dB (1) *en* $D_{Atr} \geq$ 30 dB

Figure C.6: Bureau voor normalisatie (Commissie: Geluidsleer - algemeen) [2008] describes what conditions need to be fulfilled in order to reach a certain acoustic performance level.

C.2.3 Explicit building information: RDF graphs

A building information graph was constructed for this test case, following the methodology given above. A simple building model was created in a BIM application, namely Autodesk Revit Architecture 2010. This model was subsequently exported into an IFC representation, which was then converted into an IFC/RDF graph using our in-house IFC-to-RDF converter [UGent Multimedia Lab, 2012b]. This graph was extended with information from the CONSTR/RDF graph, resulting in an RDF graph of which a part was previously presented in Fig. C.2.

C.2.4 Implicit building information: N3Logic rules

Rule description

The two above regulations had to be formalized into rules. We chose to explicitly separate the specifications described in both standards and generate both an EN 12354-3:2000 and an NBN S 01-400-1:2008 rule set. This enables, for instance, the derivation of the relevant acoustic quantities defined by the European standard, keeping the possibility to subsequently combine them with other relevant rule sets, for example, the European equivalents to the Belgian NBN S 01-400-1:2008 standard.

Both standards are manually interpreted and translated into a rule ontology in OWL, describing the concepts used within the standard, a rule set in N3Logic, describing the logic in the standard, and a conversion rule set making the references required to the building information graph. The code fragment in Fig. C.7 illustrates, for instance, how the second part of the textual resource ('Other elements') given in Fig. C.5 is translated into one of the rules in the EN 12354-3:2000 rule set.

```
{
    ?BE   rdf:type   EN12354:BoundaryElement .
    ?BE   EN12354:elementSurfaceArea   ?a .
    ?BE   EN12354:partOfRoomBoundary   ?RB .
    ?RB   rdf:type   EN12354:RoomBoundary .
    ?RB   EN12354:facadeSurfaceArea   ?atot .
    ?BE   EN12354:soundReductionIndex_4000Hz   ?R_4000 .

    ?a   math:notLessThan   1 .

    (?a   ?atot)   math:quotient   ?value_i .
    (?R_4000   -10)   math:quotient   ?value_j .
    (10   ?value_j)   math:exponentiation   ?value_k .
    (?value_i   ?value_k)   math:product   ?value_l
}
log:implies
{
    ?BE   EN12354:directSoundPowerRatio_4000Hz   ?value_l
}
```

Figure C.7: Example rule in N3Logic from the EN12354-3:2000 rule set [European Committee for Standardization (CEN), 2000]. This rule specifies how the `EN12354:directSoundPowerRatio_4000Hz` *property can be derived from a building model. The* `EN12354` *prefix refers to the base URI of the rule ontology for this standard.*

Similar to making translations between natural languages, slight discrepancies between the original text and its translation were found to be inevitable. However, having described the standards in an interoperable computer- and human-readable syntax enables an improved understanding of the contents of the standards, thereby helping involved parties to more easily make improvements to the contents of the standards.

An identical approach was followed for the NBN S 01-400-1:2008 standard, resulting in a rule ontology, a conversion rule set and the actual rule set for the standard. Starting from the concepts derived from both rule sets and their ontologies, one may thus eventually derive an explicit statement about whether or not the building model applies to a certain acoustic performance level, defined according to the Belgian NBN S 01-400-1 standard.

Relation with explicit building information

The defined rules deploy concepts stemming from a rule ontology, separately described in OWL. An inference engine needs to be able to relate these concepts to the concepts described in the building information graph. A dedicated conversion rule set is developed for this purpose, containing concepts from the RDF graph in its `IF` clauses and concepts from the rule ontology in its `THEN` clauses. In this case, the rule engine will consequently know how to translate or convert certain concepts with base URIs represented by the `ifc:` and `constr:` prefixes into concepts with the `EN12354:` prefix, which represents the base URI of the rule ontology (Fig. C.8).

```
{
    ?SS   ifc:spaceBoundary   _:675 .
    _:675  ifc:relatedBuildingElement   ?BE .
    ?BE   rdf:type   ifc:IfcWallStandardCase .
    ?BE   constr:material   ?mat .
    ?mat   rdf:type   constr:product .

    ?mat   constr:SRI63Hz   ?63Hz
}
log:implies
{
    ?SS   EN12354:soundReductionIndex_63Hz   ?63Hz
}
```

Figure C.8: Building model preparation starting from the `ifc:` and the `constr:` namespaces used in the test case.

With this conversion rule set, the concepts required for the acoustic calculations can be automatically inferred for any building information graph containing the required concepts. This can be considered being implicit information available for the building information graph at hand. This implicit information describes, for instance, an acoustic performance level acquired according to the EN12354-3:2000 and the NBN S 01-400-1:2008 standards.

C.2.5 The rule checking environment

The inference process

The information described by the rules in the knowledge base can be considered implicit information concerning the designed building. As the rules describing the two acoustic standards enable the logical inference

of an acoustic performance level, this information can be considered implicitly available building information. The inference process thus solely makes this implicit information explicit. As both the RDF graph and the N3Logic rule sets are based on logic theory, this is a standard task for a logical inference engine such as EYE. After providing the engine with the RDF graph and the rule set at hand the required inferences can be made.

Several commands can be given to the EYE reasoner, each time resulting in a different backward-chaining process with its own specific output. It is possible to reason through all information given and reproduce all inferences found in the output result. Using specific queries, however, one is able to only infer and retrieve the information needed. As we are primarily interested in the acoustic performance level of the building model, we constructed a specific query that specifically returns this information. The eventually resulting output is given in the code fragment in Fig. C.9.

```
#Processed by $Id: euler.yap 3098 2009-10-24 20:31:17Z josd $

@prefix : [...]

inst:RoomBoundary_1   NBNS014001:ComfortLevel   "normaal"^^xsd:string .
inst:RoomBoundary_2   NBNS014001:ComfortLevel   "verhoogd"^^xsd:string .
inst:RoomBoundary_3   NBNS014001:ComfortLevel   "normaal"^^xsd:string .
inst:RoomBoundary_4   NBNS014001:ComfortLevel   "normaal"^^xsd:string .


#ENDS 16 msec
#Trunk : 94/326 = 28.8343558282209 %
#Branch: 1/93 = 1.0752688172043 %
```

Figure C.9: The output of the query on a given building information graph and a set of rules indicates how four `inst:RoomBoundary` *concepts (equivalent to four wall elements) each reach a certain* `NBNS014001:ComfortLevel`*, distinguishing between 'normaal' (Dutch for 'normal') and 'verhoogd' (Dutch for 'increased'). More details on the calculation can be generated as well, but are left out here for reasons of brevity.*

Reporting of rule checking results

As can be seen in Fig. C.9, the results of the rule checking process are returned by the inference engine in the form of new RDF statements. By making these statements available in an RDF graph, for instance, within an online triple store, they can easily be reused in any other information environment for further processing. Generating such an RDF graph of the results is of course only a beginning. A substantial amount of additional research, design and implementation efforts is needed to enable a suffi-

ciently functional visualization of rule checking reports, as was indicated earlier.

D

Visualization of semantic architectural information within a game engine environment

P. Pauwels, R. De Meyer, and J. Van Campenhout

This is a summary of the work that was published in Pauwels et al. [2010b]. ***Visualisation of semantic architectural information within a game engine environment, in Proceedings of the 10th International Conference on Construction Applications of Virtual Reality, pages 219-228, 2010. http://hdl.handle.net/1854/LU-714009.*** **For more details, please consult the original article.**

Abstract Pauwels et al. [2010b] presents how architectural information that is stored in an online RDF graph can be accessed from within a game engine and visualized by this game engine. In this paper, we presented an overview and comparison of the most prominent visualization techniques, including (1) a comparison between VR and MR environments in the AEC domain, (2) a comparison between VR environments and game engines in the AEC domain, and (3) a comparison of game engines in the AEC domain. Additionally, Pauwels et al. [2010b] briefly documents how building information can be stored in RDF graphs using semantic web technologies. A short test case finally illustrates how both can be combined to enhance

information visualization for architectural design and construction. The comparisons between visualization techniques and the documentation of the test case are repeated in the sections below.

D.1 Suggested strategy for the visualization of semantic architectural information

D.1.1 Strategy

MR applications are thus typically developed for contexts in which real objects matter for the building design. In an early design context, this includes in many cases only an empty construction site. In these cases, this MR visualization does not seem such a valuable addition to the design process, because design decisions rely almost exclusively on the virtual parts of the design. Certain, more exceptional design tasks may profit from the usage of an MR visualization, including the design of underground structures [Roberts et al., 2002] and exterior architectural design [Thomas et al., 1999], for instance, but in mainstream design projects the percentage and value of real objects, such as the site, tends to be rather low in comparison with what is imagined and is essentially virtual. Because VR visualizations, on the other hand, typically involve only virtual elements, they tend to be far more useful in the early stages of the design process, in which the complete design is still mainly virtual in the designer's mind. VR visualization environments thus typically focus on visualizing as much of this internal mental image of the design to generate an improved image of the state of the design at hand.

In our research, we target the improvement of the initial stages of the design process. Following the above considerations, we focus our research on VR visualization systems and will handle only these systems in the remainder of this paper. VR visualization systems can typically be subdivided in two types, depending on the software deployed to build the virtual environments. On the one hand, we distinguish more traditional VR systems that rely on more low-level software libraries for optimizing the interface between the application logic and the VR hardware, including tracker systems, stereoscopic displays, interactive input devices, and so forth. Alternatively, we distinguish VR environments that rely on standard game engines that focus on an out-of-the-box environment for a fast and intuitive development of virtual worlds that may subsequently be optimized for certain VR hardware.

A brief and comprehensive comparison between game engines and 'virtual reality engines' is given in Al-Najdawi [2007]. This overview in-

dicates the most significant reasons why one typically chooses for a visualization based on a game engine. Comparing to other research efforts [Marks et al., 2007, Moloney et al., 2003, Wünsche et al., 2005], we extract the following main reasons.

- Solid, high-level, and out-of-the-box functionality enables a considerably fast development of functional virtual worlds
- Low cost
- Mature tools for the development of extra functionality: networking utilities, physics engine, AI engine, and so forth
- More compelling results in terms of interactivity and 3D graphics
- Designed for a remarkably good minimal performance on a whole range of operating systems and hardware configurations

Comparing this strategy overview with our focus on the initial architectural design stages, we decided to focus first on VR visualization environments based on game engine technology. Considering the large number of available game engines, it is nearly impossible to give even a brief overview of all game engines. We therefore focus on the most promising game engines in relation to visualizations in the AEC domain. An overview and qualitative analysis of such engines was recently given in Koehler et al. [2008]. Following developments in game engine technology, we give a short overview of the game engines Unity3D, Quest3D, Shiva, Virtools and Creator. We analyze to what extent they are appropriate for the visualization of semantic architectural information in a virtual interactive 3D environment, and in how far they compare or compete with the visualization environments discussed earlier. Traditionally very popular game engines, such as Doom, Unreal, and so forth, are not considered as they do not provide the required interoperability with the CAD tools typically deployed in the AEC industry, and usually require users to (re-)model all 3D in an in-house game editor to obtain sufficiently qualitative graphics.

D.1.2 Comparison of game engines

Creator

Esperient Creator is one of the newly emerged 3D engines explicitly targeting architectural design processes. The product's whitepaper indicates how the product fits perfectly into the typical architectural design process,

indicating the position of the application in relation to major architectural design tools such as Google SketchUp, Autodesk Revit, GraphiSoft ArchiCAD, Bentley v8i and Autodesk 3DS Max [Esperient, 2009]. A closer look at the work flow from CAD tools in the AEC domain to Esperient Creator, however, shows a not so intuitive process, involving a link to the relational database underlying the original CAD tool or connecting trough Microsoft's Distributed Component Object Model (DCOM) [Esperient, 2012]. Alternative file-based processes seem to require a detour via 3DS Max, which is a specialized 3D visualization tool and not a standard CAD tool possessed by any AEC specialist, and the usage of the Right Hemisphere (.rh) file format, which seems not to produce the best visual graphics when testing.

On the other hand, Esperient Creator provides a significant and useful set of standard GUI components to its users, and looks a fairly intuitive tool to use [Esperient, 2012]. Apart from the two available built-in scripting languages, it also provides an extensive C++ API for the development of advanced functionality in the visualization. Extended with the availability of ODBC database connectivity and the necessary networking functionality, Esperient Creator provides a solid basis for the advanced information visualization environment we targeted earlier.

Virtools

The 3DVIA Virtools visualization engine of Dassault Systèmes has evolved into a complex, but highly functional platform for 3D authoring and 3D visualization. The presented process 'Import - Create - Publish' [Virtools, 2012] gives an appropriate idea of the visualization process typically gone through when using Virtools for building a virtual world. Extensive functionality is provided for the creation of the eventually resulting world, extending the main platform with many additional functionality libraries, including a Physics Library, a Multiuser Library, a VR Library, and so forth. Combined with a highly functional and well-documented Software Development Kit (SDK), any user is thus provided with all the required functionality to build compelling 3D worlds.

Virtools primarily focuses on the connection with Dynamic Content Creation (DCC) applications, thereby mainly targeting animation software such as Autodesk 3DS Max. The bridge towards Product Lifecycle Management (PLM) and BIM software is provided through the 3D XML plug-in for Virtools [Virtools, 2012]. This plug-in is, however, mainly available for PLM applications by Dassault Systèmes, thereby somewhat excluding the easy usage of Virtools outside the Dassault Systèmes product suite. Combined with the high purchase cost, we tend not to consider the Virtools

platform as an appropriate tool for developing the targeted information visualization.

Unity3D

The Unity3D game engine is a recent game engine under development by Unity Technologies. The engine focuses on a fast and intuitive game development for diverse hardware and software environments, including iPhone and iPad applications, immersive installations, and so forth [Unity Technologies, 2012]. After testing, it was considered as one of the best game engines in terms of usability, intuitiveness, and resulting quality in graphics and interactivity. The game engine relies mostly on import through the FBX file format, analogous to other, similar game engines. Unity3D provides a useful API that is accessible through C# scripts and JavaScript for basic game engine functionality [Unity Technologies, 2012]. The more advanced API components, such as a VR library, a physics engine, a multiuser library, and so forth, are not available out-of-the-box, resulting in a compact, functional API for fast application development.

In our investigation of Unity3D, we experienced a remarkable support, elaborate documentation and an active user community, indicating a high level of user satisfaction. The fact that other initiatives for VR visualization, including goBIM [Keough, 2009], for instance, rely on the Unity3D engine, indicates the appropriateness of the game engine. The availability of a free version adds to these advantages.

Quest3D

The Quest3D engine, developed by Act-3D, differs considerably with the other game engines described here. This difference is mainly caused by the heavy reliance on 'channel graphs' to express interactivity in the resulting virtual world [Quest3D, 2012]. Channel graphs are graphs that are continuously called when running the virtual world, and depending on the 'channels' contained in the graph, one or another action / interaction is triggered.

The connection with existing CAD tools relies heavily on import / export / conversion plug-ins developed by third parties. As Quest3D information is described in an in-house format, any external 3D description needs to be converted into this data structure. An important information loss is experienced in this conversion, not to mention the hard balancing exercise between advantages and disadvantages of the several conversion alternatives. After experimenting, the Quest3D engine proved not as intuitive as was originally expected. Notwithstanding the promising charac-

ter and the nice results, we found the game engine inappropriate for the targeted fast and intuitive visualization of 3D building models and their information.

ShiVa

The newly emerging StoneTrip ShiVa game engine [ShiVa, 2012] focuses on compatibility with a whole range of existing DCC applications through the DAE and FBX file formats, including Blender, 3DS Max, Maya, Cinema4D, and so forth. In order to get a building model from a BIM modeling environment into the ShiVa visualization environment, one is thus required to get the model in any of these DCC applications and export it again in a DAE or FBX format, which might cause a certain amount of information. The provided import/export functionality via DWG provides an alternative work flow, but this is not the ideal approach considering the richness of the IFC format, for instance.

Furthermore, ShiVa seems to provide all the basic functionality one could expect from a game engine nowadays. Scripting possibilities via Lua are provided, an API provides developer access to the engine, and the environment can be published into a myriad of environments, including mobile hardware systems such as an iPhone and iPad. To conclude, ShiVa can provide a solid basis for an information visualization environment. Being a young and small company compared to the companies considered above, a somewhat lesser support and maintenance may nonetheless be expected for the moment.

D.1.3 The best choice?

Purely based on the functionality provided by each of the game engines, only relatively small differences were found between the engines, of which the extent can only be experienced through a far more detailed study *and* usage of each of these engines. Every game engine can to some extent be used as a basis for implementing a visualization environment for the architectural information linked to the building models on a semantic web, and none of them seems perfectly fit. Based on the extra characteristics of the game engines, such as cost, development support and popularity, we found the Unity3D engine most promising. We therefore chose to use this engine for building an example virtual environment in which a 3D building model is visualized and the semantic architectural information can be interactively accessed in real-time for each of the elements of the building model.

D.2 Accessing the architectural semantic web from within the Unity3D game engine

D.2.1 Information exchange

For testing the visualization of a building and the information available on the semantic web for this building, we modeled a design built in Antwerp by architects R. De Meyer and F. Van Hulle, using Revit Architecture 2010. Building information concerning the building design was added to the model as in every usual BIM model using the Revit BIM environment. This information was exported using the IFC export utility standard provided in any mature BIM application, including Revit Architecture 2010. The information in this IFC file was then converted into an online RDF graph using our in-house IFC-to-RDF converter [UGent Multimedia Lab, 2012a]. References were added to other information in the online open linked data cloud, including more detailed material information and geographic information [Pauwels et al., 2010a]. This information is now accessible through an online SPARQL endpoint. Through this endpoint, any information processing system may access the information described.

We did not found any game engine able to build a virtual world based on an IFC description. As the IFC format targets interoperability mainly in construction industry, it is near to useless for a game engine. As an alternative approach, one could choose to exchange the 3D information using the triangulated mesh representation made available through the API of the modeling application, as was done in Keough [2009]. We, however, chose to rely on a well used file format, ideally a formal or informal standard. Industry standards in animation industry include formats such as FBX and DAE. As Unity3D provides standard 3D exchange through FBX, this was chosen as the communication medium of the 3D information. Note that the amount of information that can be described using the IFC format exceeds that of FBX, which focuses solely on graphical information.

D.2.2 Creation of the virtual world

After import of the 3D model in FBX into Unity3D, a few actions enable the creation of an interactive virtual world. The mere inclusion of a terrain, a global directional light and a First Person Controller enable a fast production of a basic virtual world. After the import of the FBX file, every building element that was modeled in Revit Architecture is available as a separate object, identified by the original CAD object ID. As this CAD object ID is also present in the IFC description and thus also in the semantic

web graph (Fig. D.1), a connection between the object in the virtual world and the information in the semantic web graph is available.

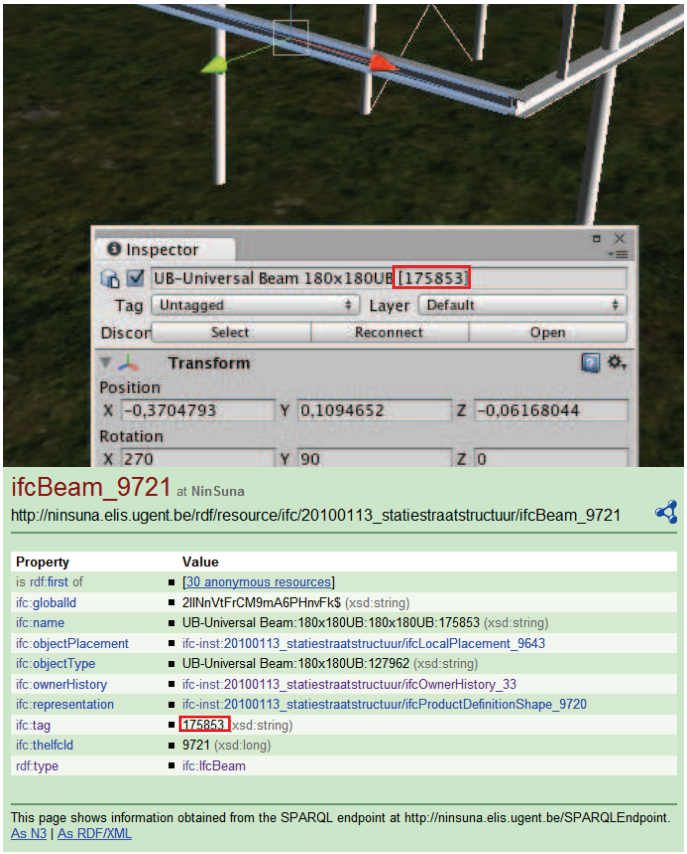


Figure D.1: The unique CAD Object ID is available in Unity3D (top) and in the semantic IFC/RDF graph (bottom).

D.2.3 Implemented functionality

Several additional scripts allow a connection from within the Unity game environment with the SPARQL endpoint on the server. When connected with this endpoint, any possible query can be sent through over HTTP. The query result received by the script can be processed or visualized as desired or required in the virtual world. At the moment, we implemented functionality so that any AEC specialist can intuitively walk around in the virtual 3D world (Fig. D.2, left) and select any object using the available

pointer. Upon selecting an object, a query is fired towards two parallel semantic web servers, the one containing a number of IFC models in one semantic web graph, and the other containing extra architectural information (topological information, material information, and so forth) added to these original IFC models. The user view changes into a separate focus view that isolates the selected element from the rest of the building and merely displays the information available for this element (Fig. D.2, right). This information is displayed as a three-dimensional graph that can be further explored by selecting the nodes in this graph.

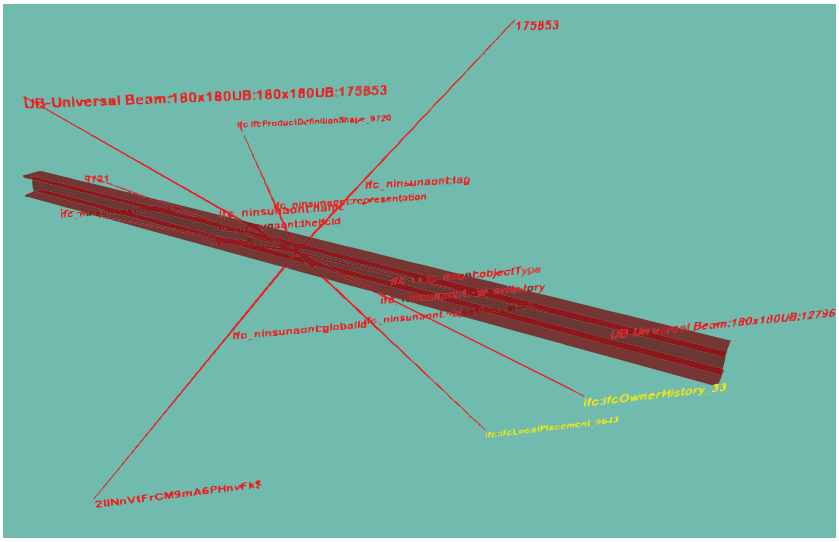


Figure D.2: Virtual world with the building structure of the visualized design, both in an overall First Person View (left) and in an isolated view for the building element highlighted earlier in Fig. D.1 (right).

D.2.4 Evaluation

The main objective of this research was to find an appropriate virtual environment allowing easy visualization of 3D building models typically developed within the AEC domain and the information adhered to these models using semantic web technology. The Unity3D game engine has indicated this functionality, and an initial implementation of the targeted functionality was implemented.

However, because research primarily focuses on the connection between a game engine and this semantic building information, we did not

elaborate on the interface displaying the information. As most AEC specialists are not familiar with a 3D graph representation in a virtual world, it proved not the most intuitive interface for the targeted functionality. As an alternative, the information might be better visualized in a form-like view, similar to existing CAD applications. We also recognized the importance of selective views on the information of the building elements. No one benefits from an 3D information 'explosion', so further interface optimization is needed also on this topic. As this is ongoing research, further reports on these topics are to be expected.

E

Information systems and reasoning for architectural design thinking support

P. Pauwels, R. De Meyer, and J. Van Campenhout

This is a summary of the work that was published in Pauwels et al. [2012a]. *A Critical Evaluation of Information Systems and Reasoning for Architectural Design Thinking Support, in Design Issues, 2012 (in press).* For more details, please consult the original article.

Abstract *Numerous attempts have been made to conceive and implement appropriate information systems to support architectural designers in their creative design thinking processes. These information systems aim at providing support in varying ways: enabling designers to make diverse kinds of visual representations of a design; enabling them to make complex calculations and simulations that take into account various relevant parameters in the design context; providing them with relevant information and knowledge from all over the world, and so forth. Despite the continuing efforts to develop these information systems, they still fail at this point to provide essential support in the core creative activities of architectural designers. Seeking to understand why an appropriately effective support from information systems is so hard to realize, we began to look into the nature of design thinking and on how reasoning processes are at play in this design thinking.*

Our investigation suggests that creative designing rests on a cyclic combination of abductive, deductive, and inductive reasoning processes. However, traditional information systems typically target only one of these reasoning processes at a time, which might explain their limited applicability and usefulness. As research in information technology increasingly targets the combination of these reasoning modes, improvements in design thinking support by information systems might be within reach.

E.1 Information system support for design

Understanding how an information system can provide support in the design process requires sufficient understanding of how design thinking occurs in this process. Constructing such an understanding has been the goal of many research initiatives during previous decades. Several appropriate overviews are available that describe the historical evolutions in these research initiatives and their outcomes [Bayazit, 2004, Cross, 2007a]. We elaborate here on some of the key points in this domain of research, focusing on the theories outlined by Nigel Cross, Bryan Lawson, Donald Schön, and Herbert Simon. Central elements in these theories are (1) the intense interaction between designer and design context, and (2) the reflective, 'learning-while-doing' character of the design thinking process. In learning-while-doing, designers build up knowledge in direct reference to concrete experiences. This knowledge is often referred to as 'a designerly way of knowing' [Cross, 1985, 2006]. Designers make design decisions in newly encountered design contexts on the basis of this kind of knowledge. Through their ongoing interaction with new design contexts, designers continuously modify or adjust their designerly way of knowing. These adjustments obviously have a significant effect on future design decisions.

Over the years, the design research community has recognized how reasoning based on a set of previous experiences is crucial for creative thinking and for mental activity in general. Several theories refer to this kind of reasoning as 'abductive reasoning' [Cross, 1990]. These and comparable theories typically refer to the work of Charles Sanders Peirce, and more specifically to his work that combines abductive reasoning with deductive and inductive reasoning. Peirce argues that the combination of these three reasoning modes underlies any process of scientific inquiry [Peirce, 1958]. Many researchers have tried to simulate these reasoning modes in a computer environment. Whether or not these attempts were successful, or could possibly and eventually be successful, is not of central concern for this article. Instead, we hope to evaluate to what extent and in which way(s) information systems might support these reasoning modes.

A digital repository of architectural information can be considered an example of a support system because it makes a limited version of other people's architectural experiences available to any reasoning process all over the world. One such repository is the repository that resulted from the initiative Metadata for Architectural Contents in Europe (MACE). This repository relies on metadata to interconnect and disseminate digital information about architecture [Stefaner et al., 2007]. However, examples of how such repositories are usefully deployed in existing architectural design practices are lacking, which might indicate that they are not fully suited to support the reasoning processes of a designer. Similar limitations exist for other types of support by information systems, essentially indicating a mismatch between design thinking and the support provided by information systems.

To understand the basic causes of such an apparent mismatch, we compare in this article how design thinking appears to occur in the human mind and how information systems aim to provide support to an architectural designer. We first give a brief overview of the most significant theories in design thinking. Second, we discuss natural human reasoning processes and how they relate to theories of design thinking. Third, we outline the supposed mismatch between the theories of design thinking and information system support, thereby considering three application development approaches. Based on this analysis, we offer suggestions for addressing the mismatch.

E.2 Theories of design thinking

A lot of theories of design thinking exist. We focus in this section on (elements of) these theories that are related to the actual reasoning processes of the designer, and to design representations and guiding principles in design.

E.2.1 Reasoning in design

Several researchers have made a case to identify design as a separate discipline in human thinking, apart from the disciplines of the arts and the sciences. With his theory of 'a designerly way of knowing' [Cross, 2006], Nigel Cross can be counted among these researchers. His theory essentially distinguishes design knowledge from the kinds of knowledge typically at play in the disciplines of the arts and the sciences [Cross, 1982]. Throughout his publications, Cross strongly associates this kind of knowledge with the specific nature of design thinking. His reference to the research of Dou-

glas and Isherwood illustrates his understanding of this kind of thinking: *"For too long a narrow idea of human reasoning has prevailed which only accepts simple induction and deduction as worthy of the name of thinking. But there is a prior and pervasive kind of reasoning that scans a scene and sizes it up, packing into one instant's survey a process of matching, classifying and comparing. [...] Metaphoric appreciation, as all the words we have used suggest, is a work of approximate measurement, scaling, and comparison between like and unlike elements in a pattern."* [Douglas and Isherwood, 1979] (p. viii).

Later on, Cross refers to several other research initiatives that distinguish a very similar kind of reasoning as fundamental for design thinking; here, he mentions the terms abductive reasoning, productive reasoning, and appositional reasoning [Cross, 1990] (p. 132), which are the terms assigned by their respective originators, Peirce, March, and Bogen [Bogen, 1969, March, 1976, Peirce, 1958].

The 'work of approximate measurement' or 'metaphoric appreciation' is closely related to analogical reasoning and the reasoning involved in pattern recognition. Analogical reasoning is often explained as the cognitive ability to think about relational patterns [Holyoak et al., 2001]. It allows one to find a structural alignment or mapping between a base and a target pattern residing in (partially) different domains [Gentner et al., 2001]. By making such an analogical mapping, familiar knowledge about the base pattern can be related to the target pattern, thereby filling in the gaps of the target pattern and creating new knowledge. In a context of architectural design, analogical reasoning often occurs between a new design-related experience (e.g., a building, sketch, 3D model, conversation, etc.) and a previous design experience. But also in the very act of sketching, analogical reasoning is critical because it allows reinterpreting or 'seeing as', as Goldschmidt [1991] puts it. In 'seeing as', the designer reinterprets the sketch and adds new and unique meaning to it, thereby generating new ideas.

Another argument for the recognition of design as a separate discipline, apart from the disciplines of the arts and the sciences, is formulated by Bryan Lawson in his theory of 'how designers think' [Lawson, 2005a]. An important basis for his argument stems from his experimental observations in which he saw differences between the thinking processes of architects (closer to 'imagining') and of engineers (closer to 'reasoning') [Lawson, 1979]. Reasoning in this case is considered more purposive and directed toward a particular conclusion, whereas imagining is said to draw from an individual's own experiences, combining material in a relatively unstructured and perhaps aimless way. Note that Lawson does not exclude the coexistence of imagining and reasoning in one mind. Instead, he consid-

ers the control of the delicate balance between rational and imaginative thought as one of the most important skills of a designer [Lawson, 2005a] (p. 138). Note also that this definition of imagining is again closely tied to analogical reasoning. Because analogical reasoning is guided by encountered target patterns, which are out of the designer's grasp, the designer appears to proceed *"in an unstructured and perhaps aimless way"*. A similar characterization is given by Boden's research on 'the creative mind' [Boden, 2004] (p. 29-36). She stresses the importance of the incubation phase in creative thinking. In this phase, the conscious mind focuses on other domains, problems, or projects, thus enabling the creative mind to make diverse analogies with the (design) situation at hand.

A second characteristic of reasoning in design is given by Donald Schön. He characterizes design thinking as a sense-making process, in which the designer *"must make sense of an uncertain situation that initially makes no sense"* [Schön, 1983] (p. 29-36). This type of reasoning process can be distinguished from more 'traditional' reasoning processes, in which problems are typically represented as well-confined and fixed givens, and one merely has to select the most appropriate method available to get to a solution. This distinction is also made by other researchers in design thinking by stating that designers do not follow a straight-forward path from problem to solution, but instead oscillate between problem(s) and solution(s). For instance, Cross indicates that designing does not proceed in a sequence of stages that targets each one of the (partial) problems initially identified or outlined. Instead, designing appears to proceed through an iterative form of interplay between partial problems and their solutions. *"During the design process, partial models of the problem and of its solution are constructed side-by-side, as it were. The crucial factor, however, is the bridging of these two partial models by the articulation of an apposite concept [...] which enables the models to be mapped onto each other."* [Cross, 1997] (p. 439). Dorst and Cross [2001] link the interplay between partial problems and solutions to the notion of 'co-evolution' of problem and solution. This notion of co-evolution was suggested previously by Maher and Poon [1996]. According to the concept of co-evolution, both problem and solution evolve during a combined process of exploration (see Fig. E.1).

Finally, a similar approach is also put forward by Herbert Simon in his 'sciences of the artificial' [Simon, 1996]. In his discussion of the problem-solving process preceding any artifact, Simon dissociates himself from the traditional, rational viewpoint, in which the outer environment would be modeled as a combination of, for example, cost and revenue curves, so that one can 'easily' calculate the optimum through a process of 'substantive rationality' [Simon, 1996](p. 25-30). In reality, the problem is far more

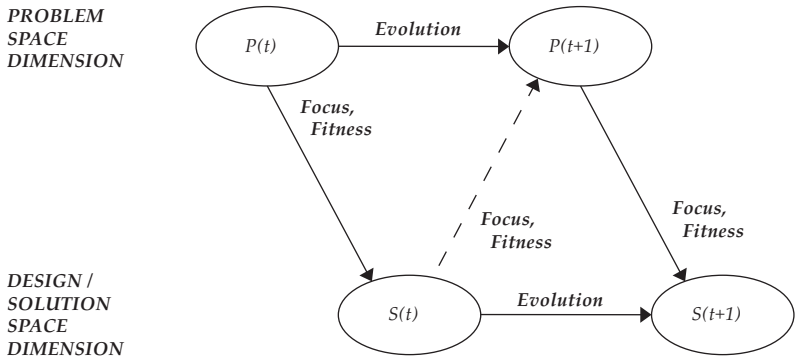


Figure E.1: Maher and Poon's 'problem-design exploration model' (courtesy of Mary L. Maher and Josiah Poon) [Poon and Maher, 1997] (p. 441)).

complex, says Simon, because one is required to cope with numerous elements of uncertainty and with quality (vs. quantity) criteria. Thus, *"at each step toward realism, the problem gradually changes from choosing the right course of action (substantive rationality) to finding a way of calculating, very approximately, where a good course of action lies (procedural rationality)."* [Simon, 1996] (p. 27). As a result, Simon eventually concludes that a designer is essentially a 'satisficer' —a person who aims at a good enough alternative, starting from a set of combinations of possible problem descriptions and corresponding solutions.

E.2.2 Design representations and guiding principles

Two elements are crucial in the sense-making process of designers: (1) what designers experience at design time (task environment/target patterns) and (2) what designers have already experienced before (background knowledge/base patterns). Both elements are also considered central in the notion of a 'design problem space', as defined by Goel and Pirolli [1992] (p. 399): *"a formalization of the structure of processing molded by the characteristics of the information-processing system [the designer's knowledge], and more importantly, the task environment [the designer's experiences]."*. By endlessly combining both elements, designers continuously form renewed understandings of their design (target pattern) and of their background knowledge (base pattern).

Sketches are among the most obvious examples of experiences that designers go through at design time [Purcell and Gero, 1998]. As Goldschmidt [1991] (p. 123-143) indicates, sketches are not only visual expressions of what one wants to express; they also are elements for reinterpretation and thus for generating all kinds of new knowledge. Cross [1999] similarly refers to the importance of sketching because it enables a designer to explore several solutions and problems to a certain design situation at once, thereby considering several levels of detail at once. Schön, in turn, refers to the habit of many a designer to continuously make representations of the design situation at hand in documents, plans, and sketches, thereby allowing a designer both to answer a previously generated problem situation or design situation, and to frame the design situation anew into an alternative perspective.

A valuable theory of the role of background knowledge of a designer is formed by Lawson's theory of so-called 'guiding principles' or 'design philosophies' [Lawson, 2005a] (p. 159-180). These concepts can be understood as the background knowledge or the knowledge by experience of a designer —the set of familiar design patterns that the designer might use in making analogies with new design experiences. According to Lawson, these guiding principles include not just objective, factual information but also information involving, for instance, motivations, beliefs, values, and attitudes. Guiding principles may be very intense and clearly structured, and they might be vague and unclear, but they always influence design decisions one way or another. In some research initiatives, they are almost considered part of a 'personal religion', thereby implicitly redefining design as *"a very complicated act of faith"* [Jones, 1970] (p. 3). With sometimes profound intensity, designers hold to these personal guiding principles, believing it 'morally right' to follow them.

E.2.3 A reasoning- and principal-based design process

From the investigation of existing theories, we can construct a possible outline of the design process. As displayed in the outline in Fig. E.2, the design process proceeds by making analogies between encountered situations in the physical world and guiding principles in the human mind. The resulting analogies can be considered the designer's interpretations of encountered situations. By making an analogy, the designer generates hypotheses and predicts that the rest of the familiar pattern also applies to the encountered situation. In other words, new knowledge is created by the analogy. The designer finally tests the prediction made, thereby creating a new situation or experience that either confirms or refutes the

original analogy. When refuted, an alternative analogy is sought. When confirmed, the pattern is added to the background knowledge, thereby indirectly changing the guiding principles of the designer. This process continuously starts anew: *"[The designers] shape the situation, in accordance with [their]initial appreciation of it, the situation 'talks back,' and [they]respond to the situation's back-talk."* [Schön, 1983] (p. 79).

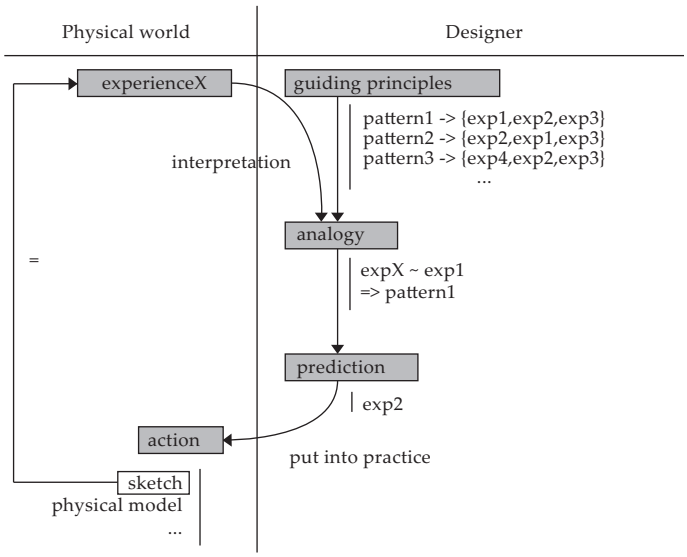


Figure E.2: Possible outline of the design process.

E.3 A reasonable explanation of human design thinking

Several parallels can be made between the schema in Fig. E.2 and theories of human reasoning processes in general. Earlier work by Purcell and Gero [1998] draws similar parallels with cognitive psychology, in which they aim at learning and refining theories in design thinking with concepts from cognitive psychology and vice versa. We mark out four main elements that can serve as a basis for documenting such parallels. Based on these parallels, we try to find a reasonable explanation of design thinking:

1. The importance of analogical or abductive reasoning in producing creative ideas,

2. The concept of co-evolution of problems and solutions,

3. The physical world with which the designer interacts (experiences), and

4. Guiding principles or background theories built up by experiences.

E.3.1 Rational problem solving

Many theoretical views exist on the reasoning mechanisms underlying natural human thinking. Several of these views originate from research on discovery and explanation in science. Aliseda [2006] (p. xii) and Aliseda [2004] indicate how 'traditional' views on this topic mainly focus on a 'context of justification' and maintain that topics like creativity, idea generation, and hypothesizing belong to a separate 'context of discovery', which is considered outside the realm of philosophical reflection and cannot be subject to any formal treatment. From this point of view, explanations are *"logical substitutes rather than real processes"* or *"rational reconstructions"*, or they are *"thinking processes in a way in which they ought to occur if they are to be ranged in a consistent system."* [Reichenbach, 1938] (p. 5). This approach forms the basis for the early work on problem solving in the 1960s. When following this approach, researchers focus mainly on the context of justification, thereby limiting their investigations to the 'rational reconstructions' in problem solving (i.e., well-structured problems) [Newell and Simon, 1972, Newell et al., 1963]. Such well-structured problems can supposedly be solved by selecting the appropriate heuristic methods.

By focusing solely on the context of justification, the first step in the problem-solving process is largely excluded. In this step, problem solvers convert a real-world problem into a well-structured problem, or, in other words, they construct a 'problem space' through a process of problem structuring. In the context of design (see Fig. E.3), this step is the one in which designers creatively interpret physical experiences using their background knowledge or guiding principles. In the traditional problem-solving approach, the element of creativity thus remains left out, resulting only in methods for solving artificial, well-structured problems. In terms of the four main elements outlined previously, this rational problem solving approach considers only the element of co-evolution (2). Analogical reasoning for producing creative ideas (1), the physical world or task environment (3), and the guiding principles (4) are largely left out.

E.3.2 Peirce's process of scientific inquiry

More recent theoretical views on human thinking try to recombine both contexts (i.e., discovery and justification) again into one 'process of inquiry'

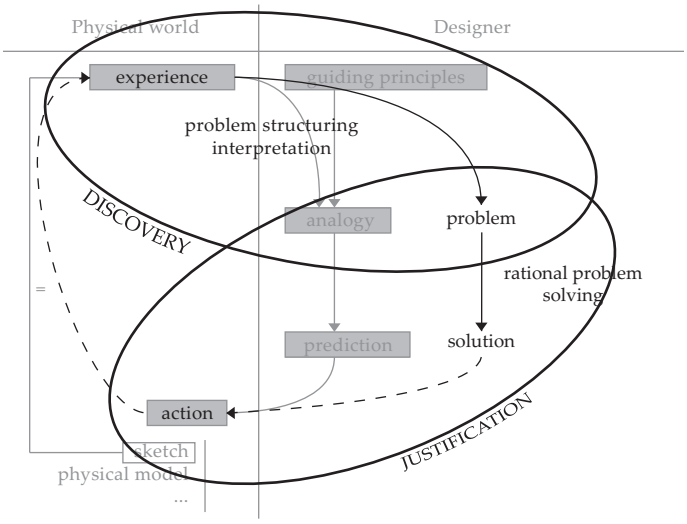


Figure E.3: The context of discovery and the context of justification in design thinking.

and try to reassess to what extent the diverse aspects or stages in this process can be subject to a formal treatment [Aliseda, 2006, Fischer, 2001, Ray, 2007]. Charles Sanders Peirce, whose work informs most of these theories, was one of the first to reconsider the process of coming to discoveries and producing justifications as one 'process of (scientific) inquiry'. Peirce's theories of reasoning and human thinking change a lot during his research, which is reflected throughout the eight volumes of Peirce's papers collected from 1931 (vol. 1-6) to 1958 (vol. 7 and 8) [Peirce, 1958]. That Peirce differentiates between three types of reasoning —namely, deductive, inductive, and abductive reasoning— is now more or less accepted. According to Peirce, reasoning should not be limited solely to a 'correct' or 'rational' kind of reasoning (i.e., deduction and induction), but instead should reflect a combination of all possible thought processes of the human mind (which includes abduction). In comparison with traditional viewpoints, Peirce thus suggests the addition of this third kind of reasoning, abductive, which encompasses our ability to generate hypotheses about the world and to choose one of them as a possible explanation.

The three reasoning modes are typically explained in the context of scientific inquiry. A good review of Peirce's understanding of this process of scientific inquiry is given by Flach and Kakas: *"When confronted with a number of observations [they] seek to explain, the scientists comes up with an initial hypothesis; then [they] investigate what other consequences this theory, were*

it true, would have; and finally [they] evaluate the extent to which these predicted consequences agree with reality. Peirce calls the first stage, coming up with a hypothesis to explain the initial observations, abduction; predictions are derived from a suggested hypothesis by deduction; and the credibility of that hypothesis is estimated through its predictions by induction." [Flach and Kakas, 2000] (p. 6).

This description illustrates how the three reasoning modes should be understood as parts of one whole —parts that continuously flow into each other and that underlie human thought, including problem solving and design. In this understanding, Peirce's theory of human thinking is in accordance with our schema of the design thinking process (see Fig. E.4), including each of the four elements previously outlined.

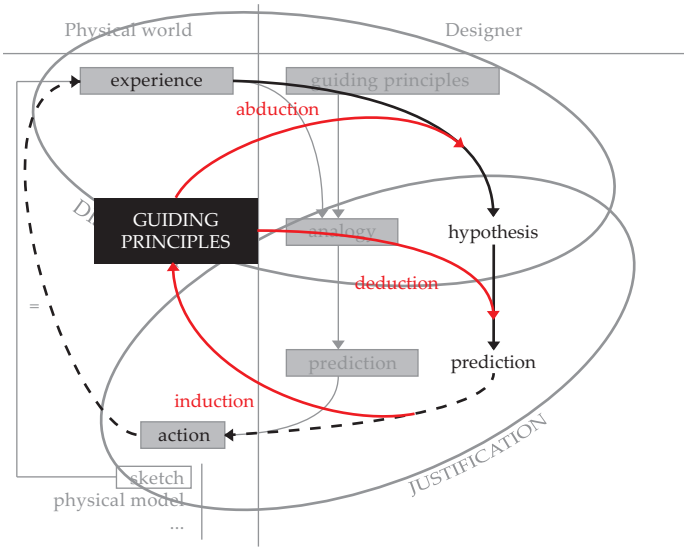


Figure E.4: The intertwining of abductive, deductive, and inductive reasoning in the context of design thinking.

E.3.3 Abduction, deduction, and induction

Peirce's understanding of each of the three reasoning modes differs substantially from the 'traditional' understanding that focuses solely on the context of justification. We give a brief summary of Peirce's discourse on these three reasoning modes in the following paragraphs.

Abduction is described by Peirce as the process of generating and choosing hypotheses in the process of inquiry: *"From a collection of observations which are judged according to some background information to be similar*

or related, we draw hypotheses that generalize this observed behavior to other as yet unseen cases." [Flach and Kakas, 2000] (p. 9). This process is most often explained using a visual observation example, in which a person makes hypotheses or tries to explain a visual observation. It is very often also associated with the 'flash of insight' or the 'eureka-moment' in a discovery [Peirce, 1958] (p. 5.181) [Aliseda, 2006](p. 171) [Flach and Kakas, 2000] (p. 7). The important caveat to keep in mind is that inferences resulting from abductive reasoning are hypotheses and not absolute truths, and they can be refuted through a series of contradicting observations.

Deduction plays an important role in the overall reasoning process. Considered separately, however, its breadth of influence appears limited in Peirce's understanding because he points out that all deductive conclusions are determined by the premises. In the context of Peirce's process of scientific inquiry, these premises are the background knowledge of the reasoning instance and the hypothesis obtained through abduction. In the terms displayed in Fig. E.4, a set of guiding principles and a hypothesis determine the prediction, but one still has to provide these premises. This part of the process can be recognized in the creation and usage of more traditional computer applications. Most of the application logic is written beforehand in a complex combination of objects, classes, and relations. Based on some user input, the application generates a certain result. The quality of this result directly depends on the quality of the premises —in this case, the application logic and the user input. The most challenging part in producing and using such applications thus lies in creating a sufficiently intricate network of premises, and not in the deduction itself.

Induction, finally, has several definitions but is most often identified with 'enumerative induction': making generalizations from a set of similar observations [Shapiro, 1991]. In this case, induction supposedly starts from a series of recurring observations, in which one sees a pattern and subsequently learns the rule behind this pattern. This understanding naturally includes a probabilistic or statistical flavor. Peirce, however, upholds a different definition following directly from his discussion of the process of inquiry (refer again to Fig. E.4). In his understanding, induction is identified with *"a course of experimental investigation"* [Peirce, 1958] (p. 5.168). This 'experimental investigation' should be understood in a broad sense, namely, any *"question put to nature"* [Peirce, 1958] (p. 5.168). For instance, such questions include scientific experiments, design tryouts, medical diagnoses, talking and listening, etc. Note that in this understanding, abductive reasoning is the only ampliative reasoning type because it offers the only moment where new, original hypotheses or "suppositions" are put forward. Induction according to Peirce's definition does not generate new

knowledge; it only confirms or refutes hypotheses that were produced previously through abduction: *"Like any interrogatory, it is based on a supposition. If that supposition be correct, a certain sensible result is to be expected under certain circumstances which can be created, or at any rate are to be met with. The question is, will this be the result?"* [Peirce, 1958] (p. 5.168).

We want to stress that Peirce's theory is not confined to scientific reasoning but can be used in several application domains, including commonsense reasoning, idea generation, architectural design, health care, and more. A recent overview of how abductive, deductive, and inductive reasoning processes are at play in design thinking, including several real world examples, can be found both in Gardner [2009] and Kolko [2010].

E.3.4 Criticism of Peirce's theory

Peirce's theory is not the only existing theory explaining human reasoning. In fact, notable criticism exists that argues against this theory. This criticism focuses mainly on the abductive reasoning mode. The main argument against Peirce's theory is the one given by Frankfurt [1958], who argues that Peirce's definition of abduction is paradoxical because it is proclaimed as the sole ampliative reasoning mode, yet it typically contains its conclusion(s) already in its premises. Frankfurt hereby refers to the logical form of an abductive inference, which indeed contains its conclusion already in its premises.

The surprising fact, C, is observed

But if A were true, C would be a matter of course

Hence, there is reason to suspect that A is true [Peirce, 1958] (p. 5.189)

If we compare this argument with Goldschmidt's terms of 'seeing that' and 'seeing as', this argument would state that 'seeing as' does not generate any new ideas because both the sketches and the human mind already had all the components necessary to produce the new ideas. An in-depth discussion of this argument is important but is beyond the scope of this article. In our opinion, novelty is generated by the unique combination of an external element in the physical world and a specific body of knowledge. Because this combination appears to be one of the main arguments of Peirce's process of inquiry, we consider this process of considerable value for the domain of architectural design thinking.

E.4 Analysis of information system support for design thinking

In this part of the article, we analyze how information technology currently provides support for design thinking, based on the theoretical overview provided. We try to indicate the main causes of the apparent mismatch between information system support and design thinking processes, so that we might be able to address it more appropriately. In this analysis, we distinguish between three application development approaches: (1) applications as surrogates for reasoning modes, (2) applications as tools for experimenting, and (3) applications as autonomous reasoning agents.

E.4.1 Surrogates of reasoning modes

Applications that might be understood as surrogates for the abductive reasoning mode in architectural design thinking are applications that supposedly create useful analogies between descriptions of a given situation and situations in memory. This process is the main driver behind the retrieval phase of case-based reasoning (CBR) applications [Aamodt and Plaza, 1994]. In CBR applications, new 'cases' are compared with a large collection of 'known cases' to find a solution to a problem by analogy. The MACE metadata archive could be considered a recent application with a similar objective —a case base described by metadata [Stefaner et al., 2007]. By specifying queries in the search or browse window, users present new cases to the system, which are then used to retrieve the most appropriate analogies from the cases in the case base.

This approach is indeed very similar to the experience-based nature of architectural design thinking. The main issue in realizing such an experience-based retrieval system is the structure used to describe the cases or experiences. As we indicated earlier, architectural knowledge is formed through personal experiences and is of a highly dynamic nature. Designing and implementing a structure to describe this kind of knowledge is difficult, or even nearly impossible, because of the ill-structured nature of architectural 'problem' situations [Goel and Pirolli, 1992, Simon, 1973]. The situation is shaped by so many parameters that one single best solution does not exist; instead, many different re-solutions are available, depending on the parameters to which one attends. When molding an architectural 'case' into an information structure, one similarly chooses certain parameters as characteristic of the case —called 'features' in the context of CBR— thereby losing characteristics that might be crucial when retrieving cases in other contexts.

Applications that might be understood as surrogates for the deductive reasoning part of the architectural design thinking process are simulation and calculation environments. As suggested, the application logic of such environments is typically written beforehand and can be considered relatively static. A calculation result is obtained based on this application logic and some user input (e.g., a CAD model). The main difficulty is creating this network of premises —input model and application logic— from which the deductive reasoning starts. Creating such a network suffers from the same issue already described: Too many essential parameters are lost in translating ill-structured real-world knowledge into a well-defined information structure.

Applications that might be understood as surrogates for the inductive reasoning part of architectural design thinking —as described in Peirce's theory— are modeling applications. Such applications enable the production of visual design tryouts in far more diverse ways than is traditionally the case. Although designers were previously 'limited' to paper-based sketching or to building physical models, they can now rely on a myriad of modeling and visualization applications to conduct experiments or create design tryouts. This process replaces the inductive reasoning mode in the sense that original hypotheses can be tested and then confirmed or refuted. However, it does not replace the inductive reasoning part in the sense that a knowledge base behind the application is adjusted according to the result of the test.

E.4.2 Tools for experimenting

An alternate possibility is to consider each of the outlined information systems or applications as nothing more than parts of the physical world with which designers interact. In this approach, applications are similar to other elements in the physical world, and interaction with the applications occurs similarly to the way that Purcell and Gero articulate interaction with sketches, diagrams, and drawings [Purcell and Gero, 1998] (p. 389-430). When reconsidering the schema discussed earlier, such applications thus lie in the lower left corner (see Fig. E.5). In this part of the overall reasoning process, they provide extra environments for producing inductive experiments.

In this application development approach, the reasoning cycle outlined by Peirce takes place entirely in the human mind, and the application is just a tool in which to conduct experiments. The assumption is that designers produce the hypothesis and prediction autonomously using their own 'designerly knowledge' and abductive and deductive reasoning. The

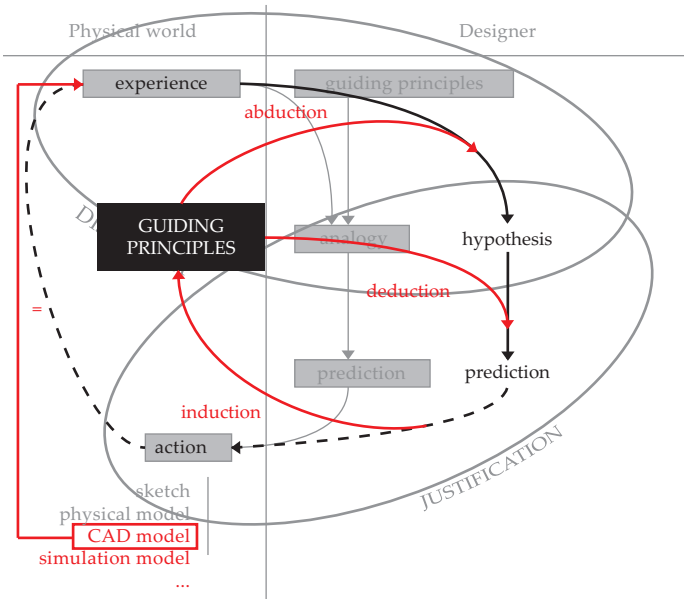


Figure E.5: Information systems as extra environments for producing inductive experiments or design tryouts.

prediction generates an expectation that can be tested in an experiment or design tryout. Modeling environments are perfect examples of how applications can act as tools for experimenting. However, the experience-based retrieval systems and the calculation and simulation applications also can be considered tools for experimenting. In a system like MACE, for instance, designers already have the kind of results in mind that they expect to see when searching or browsing. The result produced by the system confirms or refutes this expectation. Similarly, when preparing a model in an application to calculate, for instance, the energy performance level, designers already have in mind what kind of energy performance level they expect the system to produce. This implicitly known result is constituted by the abductive reasoning step taking place in the human mind. Again, the result produced by the application confirms or refutes this expectation. Note that the inductive learning also takes place in the human mind in this case.

When developing and using applications in accordance with this approach, one should bear in mind that the information structure in the application has only limited use. The actual reasoning takes place in the human mind, which is perfectly fit to handle ill-structured problems and is

thus far more powerful than any of the possible information structures of the application. The merit of producing applications using this approach is that the number of tools for experimenting in the physical world notably increases.

E.4.3 Autonomous reasoning agents

A third and last application development approach we consider is that of autonomous reasoning agents. In such an approach, all three reasoning modes outlined by Peirce are implemented and combined in a dynamic information system. In this setting, the information structure is completely dynamic: it evolves step by step through every single observation or experiment made by the reasoning agent. Combining the diverse reasoning modes in a continuous cyclical process, instead of focusing on each of these reasoning modes separately, theoretically allows for the development of an information system that can make hypotheses, make predictions, devise experiments, and learn, all based on the observations and experiments the system continuously goes through.

Research on such systems is ongoing, but early results are emerging. One of the most notable results is the 'robot scientist' system, which was developed for doing scientific research semi-autonomously in a particular sub-field of yeast micro-biology [King et al., 2009, Ray, 2007, Ray et al., 2009]. This system implements a reasoning process similar, but not identical, to the process outlined in Fig. E.4 or to Peirce's process. The result is a robot that can start from an experimental observation; interpret what it sees based on the background information that was, in this case, formalized in ontologies and inserted in the robot; make hypotheses that explain the observation at hand; devise experiments to test these hypotheses; do these experiments; and learn from these experiments. This process goes on in continuous cycles. Because Peirce's process of inquiry is supposedly also at play in other application domains, a similar approach could theoretically be developed for design thinking support. For instance, semantic web technologies theoretically enable software developers to build information systems dynamically from the output of reasoning engines [Berners-Lee et al., 2001, Bizer et al., 2009]. This could eventually result in autonomous reasoning agents. Other approaches might be equally feasible if they combine the three reasoning modes into one system, following the outline of Peirce. This combination is necessary because, as we have seen, working with separate surrogates for reasoning modes brings insufficient added value.

This kind of support would be completely different from any traditional kind of support currently provided by information systems. Similarly to the way in which it happens in the robot scientist project, the reasoning system would evolve into an independent agent reasoning about a design situation, and it would thus not directly interfere with reasoning processes of the human designer. The main support it could give to a human would presumably be similar to the support any designer gives to any other designer, namely, by simple dialogue and discussion of design alternatives, from which both generate their own interpretations and start their own reasoning processes. A similar agent-based role for information systems was suggested by Lawson [2005b].

We go through a simple example, starting from a design brief, to show how such a reasoning agent might work in an architectural design environment. The design brief is given to the reasoning agent. Similar to the human process, the reasoning agent goes through the brief line by line, word by word, at each step interpreting the contents of the brief using its own personal background knowledge. In every step, the reasoning agent actually goes through the entire reasoning cycle, thereby making hypotheses about the meaning of words in the brief (interpretation), predicting what it will read next, and testing its prediction (projection) by actually reading through the next word. At every step in this reading process, a —hypothetical and fallible— understanding of both 'problem' and 'solution' is constructed (cfr. co-evolution). After reading through the design brief, the reasoning agent proceeds in the way it assumes is best. This decision is again made using one or more reasoning cycles. For instance, the agent might hypothesize that the best way to proceed is to construct a 3D model of a part of the problem. Starting from this hypothesis, a whole set of additional reasoning cycles starts, enabling the reasoning agent to continuously undertake an action, reflect on the action, and make new hypotheses based on the action. Through these reasoning cycles toward a complete 3D model, the agent not only reflects on and learns about 3D modeling; it also adjusts its initial understanding of the design problem and solution into a new and more refined understanding. And the cycles continue.

Such a situation is far from being realized, and whether it can ever be realized is unknown. In the robot scientist project, a significant amount of information was formalized and inserted as a background information model for the robot. With this formalized knowledge in place, 6.6 million optical density measurements were made, eventually resulting in a formalized scientific argumentation structure that includes more than 10,000 different research units in a hierarchical structure of 10 levels. These research units are representations of segments of experimental research, including

studies, cycles, trials, tests, and replicates. Note, however, that the system ultimately addresses only a very small subdomain in functional genomics [King et al., 2009] (p. 85-89). Building a similar model for all "designerly" information is practically infeasible, especially when taking into account Lawson's remark that designerly knowledge also includes motivations, beliefs, values, and attitudes. For such an approach to work, the information model used by the reasoning agent will thus have to be built up by the agent. Although this model is theoretically possible using a combination of abductive, deductive, and inductive reasoning, a more challenging objective would be difficult to imagine.

A crucial issue in building such a system is that the theory requires the reasoning agent to be actively embedded in a physical world if it wants to learn anything of realistic 'size'. In the case of architectural design, the reasoning agent would have to actually go through realistic architectural design processes by itself. That anyone would ever actually allow the agent to do so, is highly unlikely. But even before such a scenario could come within reach, the reasoning agent would have to be able to communicate reliably with the surrounding physical world according to the theory. In fact, such an agent would need to be able to sense (i.e., in auditory, olfactory, gustatory, visual, and tactile ways) and act similarly to the way humans do to construct useful knowledge autonomously and produce useful input to anyone. Such a development would be problematic even for the most basic sensory communication because little is known about how this process of sensing, constructing knowledge and acting accordingly occurs in the human body. Consequently, this third approach appears highly unlikely to be implemented anytime soon.

E.5 Discussion and concluding remarks

We have presented in this article a critical evaluation of information system support for architectural design thinking. This evaluation was closely tied to a significant number of theories, both in design thinking and philosophy. We chose to use Peirce's process of (scientific) inquiry as a helpful explanatory framework for several of the phenomena identified in architectural design thinking. This process of inquiry continuously proceeds by iteration through a cycle of abductive, deductive, and inductive reasoning, with a dynamic background knowledge and parts of the physical world as its premises.

As a result, we distinguish three main development approaches in information system support for architectural designers. In the first approach, applications are designed and implemented as surrogates for each of the

reasoning modes included in Peirce's process of inquiry. Such an approach appears to be of less value because it requires both application developers and users to endlessly convert real-world problem situations, which are essentially ill-structured, into artificial, well-defined problem situations. This process is tedious and time-consuming, and compared to our own reasoning capabilities, its results are not as reliable and useful as they should be. In the second approach, all information systems providing support to a designer are considered additional parts of the physical world. Similar to a paper and a pencil, a CAD system or a simulation environment allows a designer to make inductive experiments or design tryouts. The main disadvantage of this approach is that resulting support systems are of limited use because the actual reasoning process remains completely out of the application environment. The main value of this approach is that an important number of additional, previously unavailable 'test environments' can be provided to designers. This second approach is the one generally used these days, although it might not always be recognized as such. The third and last approach is to build reasoning agents that autonomously go through the reasoning cycle outlined by Peirce. This radically different approach, to our knowledge, has never been implemented in its complete form. In this case, all three reasoning modes are combined in a dynamic information system, changing continuously in response to the experiences of the reasoning agent. With this combination, such an agent theoretically should be able to make hypotheses, make predictions, devise experiments, and learn, all based on the observations and experiments the system continuously goes through. The main barrier towards this approach, assuming that it is even feasible, is that the reasoning agent needs to be actively embedded in a physical world and needs to be able to communicate reliably with this world. Without such embeddedness and communication, it would never be capable of building up an adequate knowledge base, nor of communicating this knowledge to designers.

Of the three approaches, the second is the most —perhaps the only— feasible strategy toward information system. What both designers and software developers have to bear in mind, however, is the limited effect of this approach on architectural design thinking. This approach contradicts many new and innovative proposals in information system support for design thinking that claim to provide all kinds of automation features and generative mechanisms. As we indicated in this article, such systems remain to be used mainly as useful environments for design tryouts.