



Vrije Universiteit Brussel

Faculteit Wetenschappen en Bio-ingenieurswetenschappen
Vakgroep Computerwetenschappen

Dynamic Content in Fluid Cross-Media Documents

Proefschrift ingediend met het oog op het behalen van de graad van
Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Jochen François

Promotor: Prof. Dr. Beat Signer
Begeleider: Prof. Dr. Bruno Dumas

2013-2014





Vrije Universiteit Brussel

Faculty of Science and Bio-Engineering Sciences
Department of Computer Science

Dynamic Content in Fluid Cross-Media Documents

Graduation thesis submitted in partial fulfillment of the requirements for
the degree of Master of Science in Applied Science and Engineering: Computer Science

Jochen François

Promoter: Prof. Dr. Beat Signer
Advisor: Prof. Dr. Bruno Dumas

2013-2014



Abstract

This dissertation presents a conceptual model of dynamic content. This model serves as the foundation for the development of dynamic documents. The need for dynamic documents is clarified in the overview of the multidisciplinary domain of ubiquitous computing. This background information presents the state of the art of *The Internet of Things*, sensor systems, context-aware systems and ambient intelligence. Furthermore, it also discusses the need for dynamic documents in ubiquitous environments.

The creation of the conceptual model is based on an inductive approach, where the model was extracted from the implementation of three dynamic documents. These dynamic documents were chosen from a set of scenarios that were devised through an analysis of previous research with regard to dynamic documents and an analysis of document formats. The analysis of previous research has revealed that there is no consensus on what functionalities a dynamic document should possess. Furthermore, the analysis of document formats has shown that, besides some exceptions, most document formats do not support dynamic content and that they have to rely on other technologies to enable it.

Along with a conceptual model, a formal definition is introduced for the term “*dynamic content*”. Furthermore, based on this definition we were able to define the term “*dynamic documents*”. This definition prevents further misconceptions on the term and forms a common ground for further developments. In order to facilitate the comparison of different dynamic document solutions, a categorisation is proposed with regard to content adaptation. Afterwards, this categorisation was applied to a set of scenarios to proof its efficiency.

To demonstrate the usability of the proposal (i.e. the conceptual model) in practice, the conceptual model was applied to develop an online document editor for creating dynamic documents. This document editor relies on a “*pipe and filter*” design to facilitate the creation of dynamic documents. Hereby, the editor distinguishes itself from the traditional WYSIWYG (i.e. *What You See Is What You Get*) editors. The comparison of our editor with several related systems, has shown us that the developed editor stands out by hiding the technical details of the dynamic content from the end user. The editor is targeted towards regular end users without a technical background. The primary goal of the developed editor, is to find a balance between usability and expressiveness.

Samenvatting

Deze thesis introduceert een conceptueel model voor dynamische content. Dit model dient als basis voor de ontwikkelen van dynamische documenten. De nood aan dynamische documenten is verduidelijkt in het algemeen overzicht van het multidisciplinaire domain van *ubiquitous computing*. Dit overzicht verduidelijkt de begrippen en de huidige stand van zaken aangaande *The Internet of Things*, sensor systemen, context-aware systemen en *ambient intelligence*. Verder wordt ook verduidelijkt hoe digitale documenten van dienst kunnen zijn in ubiquitous omgevingen.

Het conceptueel model werd ontwikkeld aan de hand van een inductieve onderzoeksmethode waarbij het model is opgebouwd op basis van de ontwikkeling van enkele scenario's van dynamische documenten. Deze scenario's zijn tot stand gekomen dankzij het analyseren van vorige onderzoeken naar dynamische documenten en het analyseren van een reeks van "*document formats*". Uit een eerste analyse is gebleken dat er geen consensus bestaat over welke functionaliteiten een dynamisch document moet bezitten. Daarnaast heeft de analyse van verscheidene "*document formats*" ook aangetoond dat, behoudens enkele uitzonderingen, de meeste "*document formats*" geen ondersteuning bieden naar dynamische content toe. De meeste "*document formats*" vertrouwen op externe technologieën om de documenten dynamisch te maken.

Naast een conceptueel model wordt ook een formele definitie voorgesteld voor de term "*dynamische content*". Deze definitie is ontwikkeld om vervolgens de term "*dynamische documenten*" te definiëren. Deze definitie heeft als doel verdere misverstanden te vermijden en om als basis te dienen voor verdere ontwikkelingen. Daarnaast is er een categorisatie ontwikkeld die toelaat om verschillende dynamische documenten met elkaar te vergelijken aan de hand van bepaalde karakteristieken. De voorgestelde scenarios werden, overeenkomstig deze categorisatie, geclassificeerd om de bruikbaarheid te verduidelijken.

Om de inzetbaarheid van de voorgestelde oplossing in de praktijk aan te tonen, werd het conceptueel model gebruikt als basis voor het ontwikkelen van een online document editor. Deze editor ondersteunt het creëren van dynamische content door gebruik te maken van een "*pipes and filter*" design. De editor onderscheidt zich hierdoor van de traditionele "*WYSIWYG*" editors. De vergelijking van deze editor met enkele andere systemen, leert ons dat de ontwikkelde editor zich onderscheidt van de andere systemen doordat zij de technische details van de dynamische content tracht te verbergen. De editor is ontworpen voor de gewone gebruiker zonder technische achtergrond. De principiële doelstelling van de voorgestelde editor is om een balans te vinden tussen gebruiksvriendelijkheid en expressiviteit.

Acknowledgements

I would like to express the deepest appreciation to my promoter Prof. Dr. Beat Signer and to my supervisor Dr. Bruno Dumas, for their continuous support of my Master study and research, for their patience, motivation and extraordinary knowledge. I would also like to thank Ahmed A. O. Tayeh for his advice on my research.

In addition, a special thanks to my family and friends. The support of my mother, father, sister and grandparents were invaluable for this thesis. I also want to thank them for all of the sacrifices they made for my behalf.

Contents

1	Introduction	1
1.1	Context	1
1.2	Research Question	3
1.2.1	Sub-Questions	3
1.3	Research Approach	4
1.3.1	Review Existing Solutions	4
1.3.2	Analyse Document Formats	4
1.3.3	Develop Prototypes	4
1.3.4	Define the Term “Dynamic Content”	5
1.3.5	Develop a Conceptual Model for Dynamic Content	5
1.3.6	Develop a Document Editor as Proof of Concept	5
1.4	Thesis Structure	5
2	Background	7
2.1	Ubiquitous Computing	7
2.1.1	The Internet of Things	8
2.2	Sensors and Smart Environments	10
2.2.1	The Evolution of Sensors	10
2.2.2	Sensor Data Analysis	12
2.2.3	Semantic Sensor Data	13
2.3	Context Awareness	14
2.3.1	There is More to Context than Location	14
2.4	Middleware for Pervasive Systems	15
2.4.1	Requirements for Middleware	16
2.4.2	Available Middlewares	17
2.5	The Evolution of The Internet of Things	18
2.5.1	Analysis of a Research Project: Smart Kindergarten	18
2.5.2	Commercial Solutions	20
2.5.3	Open Research Problem in Ubiquitous Computing	24
2.6	Ambient Intelligence	25
2.6.1	Ambient Documents	26

2.7	Scope of the Thesis	29
3	Review of Dynamic Content in Digital Documents	31
3.1	Analysis of the State of the Art	31
3.1.1	Active Tioga Documents - 1990	32
3.1.2	Multivalent Documents - 1996	35
3.1.3	Stick-e Documents - 1996	36
3.1.4	Live Documents - 2002	37
3.1.5	Active Documents - 2003	40
3.1.6	Minerva Documents - 2005	41
3.1.7	Multimodal Documents - 2006	44
3.1.8	WOAD Dynamic Documents - 2011	45
3.1.9	Conclusion	47
3.2	Review of Document Formats	48
3.2.1	L ^A T _E X	48
3.2.2	HyperText Markup Language	50
3.2.3	Portable Document Format (PDF)	51
3.2.4	Extensive Markup Language	53
3.2.5	OpenDocument	54
3.2.6	DocBook	55
3.2.7	Office Open XML	56
3.2.8	Electronic Publication (EPUB)	56
3.2.9	iBook	57
3.2.10	Conclusion	58
4	Dynamic Document Scenarios	61
4.1	Scenarios	61
4.1.1	Categorisation of Scenarios	64
4.2	Prototypes	68
4.2.1	RSL Metamodel	68
4.2.2	Objectives of the Prototypes	75
4.2.3	Infrastructure	75
4.2.4	Prototype 1 - Interactive Vocabulary	75
4.2.5	Prototype 2 - Adaptive Travel Guide	79
4.2.6	Prototype 3 - Live Travel Guide	82
4.3	Conclusion	84
5	Conceptual Model of Dynamic Content	85
5.1	A Formal Definition of Dynamic Content	85
5.2	Conceptual Model of Dynamic Content	86
5.2.1	Elements of the Conceptual Model	87

5.2.2	Tubes as First Class Objects	91
5.2.3	Conceptual Model	93
5.3	Conclusion	95
6	Proof of Concept: Document Editor	97
6.1	Objectives of the Proof of Concept	97
6.2	Related Systems	97
6.2.1	Squidy	98
6.2.2	Yahoo Pipes	99
6.2.3	iBooks Author	101
6.3	Document Editor	103
6.3.1	Approach of the Document Editor	104
6.3.2	Architecture of the Document Editor	112
6.3.3	Comparison to Related Systems	115
6.4	Conclusion	117
7	Conclusion and Future Work	119
7.1	Summary of the Research	119
7.1.1	Research Questions and Main Contributions	121
7.1.2	Limitations	123
7.2	Future Work	124
A	Scenarios	1
A.1	Active Storytelling	1
A.2	Dynamic Restaurant Menu	2
A.3	Adaptive Tour Guide	2
A.4	Smart Content	3
A.5	Finger Reader	4
A.6	Fluid Reading	4
A.7	Fluid Font	5
A.8	Interactive Vocabulary	5
A.9	Live catalogue	6
A.10	Live Travel Guide	6

List of Figures

2.1	An iPhone 4 with its integrated sensors. [39]	10
2.2	The Mediacup. [24]	12
2.3	Smart Table	19
2.4	iBadge	19
2.5	Nest Thermostat's discrete design	21
2.6	Nest Thermostat's mobile application	21
2.7	Philips Hue lamp and mobile application	22
2.8	Lockitron's lock and mobile application	23
2.9	Relationship between AmI and other areas [4].	25
3.1	Generated document representation of the telephone directory document [65]	33
3.2	The logical structure of a document representing telephone numbers with dynamic query nodes [65]	33
3.3	Procedures constituting a node transformation activity [65]	33
3.4	LiveDocuments adaptation characteristic [74]	38
3.5	Document B is "transcluded" in document A, P and Q ¹	39
3.6	The network is the document [51]	42
3.7	The architecture of a context-aware document adaptation system [15]	44
3.8	The result of invoking <code>getEvents@TimeOut.com</code> [1]	54
3.9	Architecture: OpenOffice combined with Python-Uno and MySQL to create real-time slides	55
3.10	iBooks Author: Available Widgets	58
4.1	Categorisation of dynamic document scenarios	67
4.2	Core Link Metamodel [45]	70
4.3	User management [45]	71
4.4	Layers [45]	72
4.5	Navigational and structural links [45]	73
4.6	Prototype 1: Learning table - <i>Fully exposed to light</i>	76
4.7	Prototype 1: Learning table - <i>Fully covered from light</i>	76

4.8	Prototype 1: Learning table - <i>Partially covered from light</i>	76
4.9	Prototype 1: Quiz - <i>Fully exposed to light</i>	77
4.10	Prototype 1: Quiz - <i>Covered from light</i>	77
4.11	Prototype 1: Shake it - <i>Device Shuffled</i>	78
4.12	Prototype 1: Rotate left	78
4.13	Prototype 1: Rotate right	78
4.14	Prototype 1: RSL - Content Presentation	79
4.15	Prototype 2: Informational page over Sydney	80
4.16	Prototype 2: Informational page over Brussels	80
4.17	Prototype 2: RSL - web service and observable resource	81
4.18	Prototype 3: Travel guide information of Brussels	82
4.19	Prototype 3: Travel guide information of Sint-Pieters-Leeuw	82
4.20	Geofence with latitude 50.784, longitude: 4.250 and radius 15 meter	83
4.21	Prototype 1: RSL - Content Presentation	83
5.1	Dynamic content shell	86
5.2	Conceptual model: Component	88
5.3	Conceptual model: Components with crosslets	88
5.4	Conceptual model: Components of content example	89
5.5	Conceptual model: Tubes	90
5.6	Conceptual model: Example of digital content - tubes	91
5.7	Conceptual model: Text specific access rights	92
5.8	Approach 1: Access rights on the text	92
5.9	Approach 2: Access rights on the tubes	93
5.10	Conceptual model of dynamic content	94
6.1	Squidy Design Environment [36]	98
6.2	Squidy Design Environment: Zoomed in Kalman filter [36]	99
6.3	Yahoo Pipes: YouTube links for the top 10 song on iTunes	100
6.4	iBooks Author: Inspector window	102
6.5	Document editor: Start view	104
6.6	Document editor: Start view	105
6.7	Document editor: Details of a document	106
6.8	Document editor: Template view	107
6.9	Document editor: Editing phase	108
6.10	Document editor: Placeholder 4 composition	110
6.11	Document editor: Web service configuration	111
6.12	Document editor: Tube's "Visible To" property	112
6.13	Architecture of the document editor	113
6.14	A general class diagram of the extended RSL metamodel implementation	114

6.15 Filter Component	116
---------------------------------	-----

List of Tables

3.1	Summary of analysed documents formats	59
4.1	Tools and Platform	75

List of Listings

3.1	Example of a Stick-e note in SGML	37
4.1	Observable resource	80

Chapter 1

Introduction

1.1 Context

The craft of making paper is a Chinese invention. From there it found its way into the Muslim world and Europe. However, the invention of paper did not directly lead to the invention of printing.

In 1454, Johannes Gutenberg invented the very first printing press for mass production. The printing press was used to print the first major book called “*The Gutenberg Bible*”². The invention of Gutenberg made it possible to reproduce books on paper without having to copy them by hand or by printing them into engraved wooden blocks. This invention has led to the explosive expansion of paper. As a result, paper has gained an unassailable position in distributing large amounts of information to a wide audience.

Technological advances of the 19th century allow us to capture and process data that updates over time. Unfortunately, since paper cannot rewrite itself, it is unable to present such information. This evolution has resulted in a first competitor of paper documents, namely digital documents. Unlike paper documents, digital documents can make use of the computational power of a computer to update their content.

However, despite the ability of digital documents to make use of the available computation power, we notice that apart from some exceptions most digital documents imitate their paper counterpart. In the light of this thesis we emphasise on their incompetence for integrating dynamic content.

²<http://www.gutenberg-bible.com/history.html> last accessed on 09/05/2014

An important concept that may benefit from enriching digital documents with dynamic content is ubiquitous computing [75]. The world of ubiquitous computing can be seen as a world where computation is all surrounding wherefore they are able to obtain an enormous amount of information. The intention of ubiquitous computing is to obtain a world where we interact with computers without thinking about them, in other words *“computers vanish into the background of our lives”*.

Since computers will be surrounding us in every possible way, they can provide us with an enormous amount of data. This data can be used to create digital documents that act as intelligent agents for presenting the right information. However, in order for the document to modify itself, it must support the integration of dynamic content.

A limited set of document formats also highlighted the need for the integration of dynamic content. Nevertheless, besides the support offered by the document format, we must facilitate the creation process of digital documents with dynamic content. Up to now, a software or web developer is responsible for extending a digital document with dynamic content. This approach introduces a level of complexity that is unacceptable for regular users. Even more, the line between document authors and programmers will become blurred.

In order to facilitate the integration and creation of dynamic content in digital documents, we need to capture its essence. We may have a mental model of what dynamic content means but in order to make use of this model, we need to refine it and translate the mental model into a conceptual model.

This thesis makes an effort in solving one of the problems that we encounter with digital documents. By analysing the concept of dynamic content, we facilitate their integration in digital documents. Furthermore, this thesis makes an effort to provide an interface, targeted towards regular end users, that should facilitate the creation of digital documents with dynamic content. This interface seeks to provide a fine balance between usability and expressiveness.

1.2 Research Question

Despite the computational power available to digital documents, we have yet to see their support for dynamic content.

What is “dynamic content” and how can we assist its integration into digital documents by regular end users?

Before we can answer these questions, we need to answer a set of sub-questions.

1.2.1 Sub-Questions

1. **What does the term “dynamic content” mean?** The term “*dynamic content*” lacks a precise definition. As a result, the term “*dynamic documents*” is also unclear. Not only will a definition bring a sense of unity amongst researchers, we also believe that a definition will provide a guidance throughout this thesis and future work.
2. **Were there any attempts made in supporting dynamic content and why were they not successful?** In order to find a proper solution, previous research must be analysed: *What was their approach?; What were the mistakes they made?; What were the limitations of their solution?*.
3. **To what extent do the current document formats enable the specification of dynamic content?** By reviewing a set of representative document formats, we are able to deduce the current state of progress on how digital documents provide support for dynamic content.
4. **What are the struggles and challenges that come along when integrating dynamic content?** Dynamic content extends the document with elements that will change over time. These unpredictable changes may entail other obstacles in the editing phase and the viewing phase. By identifying these obstacles, we are able to foresee what measures should be taken when editing and presenting such content.
5. **How can we attain a fine balance between expressiveness and usability with regard to the creation of dynamic content?** Document editors are tools developed to assist the end users in the process of editing digital documents. It is interesting to see which approach is the most desirable for creating dynamic content?

1.3 Research Approach

In order to answer the research question of this thesis and its aforementioned sub-questions, an inductive approach was used.

We start by examining the current state of the art of digital documents with support for dynamic content. Next, we analyse a set of representative document formats towards their support for integrating dynamic content. Afterwards, we discuss several use cases of documents containing dynamic content. We also describe how we have implemented a prototype for three of these use cases.

After gaining the necessary insights on the subject, we define the term “*dynamic content*”. Next, we develop a conceptual model of dynamic content based on our experience with the prototypes. Eventually, we put our conceptual model to the test by implementing a proof of concept. This proof of concept is a document editor that supports the creation of digital documents with dynamic content.

This approach can be divided into the following steps:

1.3.1 Review Existing Solutions

Previous attempts have been made to integrate dynamic content into digital documents. Unfortunately, most of these solutions do not seem to fully solve the problem. As a result, we review these solutions according to the following criteria: support for dynamic content (i.e. expressiveness), used approach and editing process (i.e. usability). This phase will contribute to the understanding of the term “*dynamic content*”.

1.3.2 Analyse Document Formats

Over the years, a myriad of document formats have emerged. In order to get an idea to which degree these document formats currently support dynamic content, an analysis must be performed. These document formats may have their own way of specifying dynamic content or they may have none. By analysing these formats, we are able to create a conceptual model that is general enough to satisfy most document formats.

1.3.3 Develop Prototypes

A set of scenarios with regard to dynamic content are devised to investigate wherefore dynamic content can be used and what its expectations are. These scenarios

are then categorised according to their characteristics. Next, three of these prototypes are created to provide an insight in their development process.

1.3.4 Define the Term “Dynamic Content”

After investigating the idea behind dynamic content, we are able to specify a formal definition for it. This definition helps us to develop a general definition for the term “*dynamic document*”. Not only does this definition become useful for further research, it also opens new questions.

1.3.5 Develop a Conceptual Model for Dynamic Content

Once we have determined the definition and the scope of the dynamic content, we are able to develop a conceptual model of it. This conceptual model makes an effort in capturing its components by means of entities and relations between those entities.

1.3.6 Develop a Document Editor as Proof of Concept

After creating a conceptual model for dynamic content, it is put to the test by developing a document editor. This editor is used to verify the generality of our conceptual model. The editor is developed with the intention to simplify the creation process of digital documents with dynamic content. In order to justify the editor’s approach, three relevant systems are analysed.

The developed editor is built around the RSL metamodel, a conceptual model for hypermedia systems.

1.4 Thesis Structure

The **first chapter** describes the context of this thesis. Afterwards, the different research questions of this thesis are proposed. The chapter continues by discussing the chosen research approach. Finally, this chapter presents the structure of the thesis.

The **second chapter** serves as introduction to the age ubiquitous computing and its relevant technologies. Afterwards, the link between digital document and this new age is emphasised.

The **third chapter** contains a literature review of the related work on digital documents with dynamic content. Along with the literature review, an analysis is presented of representative document formats and their support for integrating dynamic content.

Several scenarios of digital documents containing dynamic content are discussed in the **fourth chapter** along with a categorisation. Next, the development of three scenarios are discussed in more detail.

A formal definition and a conceptual model of “*dynamic content*” is presented in the **fifth chapter**. Subsequently, a definition of the term “*dynamic document*” is proposed. Furthermore, this chapter explains how the Resource-Selector-Link metamodel was extended to support the creation of dynamic content.

The proof of concept of this thesis is presented in the **sixth chapter**. The proof of concept is an editor that facilitates the process of creating and managing dynamic content in digital documents. This chapter also explains the architecture of our editor and some technical details. A review of related systems is presented to justify the approach taken by the proposed document editor.

Finally in the **seventh chapter**, a conclusion is presented to discuss how the research questions were answered and how this thesis contributes to its respective field of research along with some insights for future work.

Chapter 2

Background

This chapter serves as an introduction to the landscape of *ubiquitous computing*. We discuss the different technologies that enable this new age and present their overall progress.

Afterwards, we discuss how digital documents must evolve in order to keep up with this new age of computing. This discussion is preceded with an introduction to *ambient intelligence*.

2.1 Ubiquitous Computing

Currently, we are evolving from the era of personal computing towards the era of ubiquitous computing. In the first era, people are sitting in front of their computers and use them in a conscious manner. Indeed, there exists a barrier between the digital world and the physical world. This means that computers are unaware of the world in which they are used.

The era of ubiquitous computing, also called the age of “*calm technology*” or “*pervasive computing*”, was coined in 1988 by Mark Weiser [75]. Weiser was a chief scientist at Xerox Parc and performed many of his researches on the subject. Some of his best known papers on ubiquitous computing are the following: “*The Computer for the 21st Century*”[77], “*The ParcTab Ubiquitous Computing Experiment*” [73] and “*Some Computer Science Issues in Ubiquitous Computing*” [76].

In “*The computer for the 21st Century*”[75] (one of Weiser’s first papers on ubiquitous computing), it is stated that the most profound technologies are the ones that disappear in our daily lives. In order to achieve this, he states that we need

to reconsider the way we think about computers. In order for computers to completely disappear in the background, they must take the natural environment into account. Then and only then are we able to obtain the real potential of information technology.

Fifteen years later, after Mark Weiser's vision, electronics and computers have evolved and found a way into our homes and infiltrated into our lives. In the year 2006, Mark Wieser has noted³ that the average amount of microprocessors of an American family house reached the milestone of forty units and this number shall only increase over time. Since computation will take place in various devices, places and forms, we are likely to notice a change in the way we work and interact with computers.

Since computers are surrounding us, they will become aware of our presence, habits, desires, etc. This extra layer of knowledge creates a myriad of opportunities towards computing. Although, in order to make computing a truly invisible part of our lives, we cannot rely on the technological advances alone.

According to Weiser, the disappearance of computing in our daily lives will largely depend on human psychology. A well-known example of this phenomenon is the act of writing and reading (some consider these actions as the first form of information technology³). Written information has become so entangled with our world (billboards, books, tickets, etc.), that we absorb written information without consciously performing the act of reading. From this, we may derive that written information is freed from thinking, allowing us to focus on reaching other goals. Therefore, it is preferable that we seek methods to reach the same degree of consciousness when working with current and future information technology.

2.1.1 The Internet of Things

Another concept that relates to ubiquitous computing is the "*Internet of Things*". The term was introduced in 2009 by Ashton Kevin in [2]. The *Internet of Things* represents a vision where the Internet is no longer limited to entities of the digital world. According to Friedemann Mattern and Christian Floerkemeier [43], the *Internet of Things* will be primary responsible for making computing truly ubiquitous.

Today, most of the information on the Internet is provided by humans. However,

³<http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm> last accessed on 05/06/2014

humans have a limited amount of time available to create information, are likely to make errors and have limited accuracy. According to Kevin Ashton, we should empower computers to have their own senses. That way, computers on their own could collect data for themselves without being restricted to the limitations of the human-entered data.

This means that we have to create a link between the physical environment and the digital world. By digitalising objects in the physical world, we obtain smart objects and smart environments⁴. A manner of digitalising physical objects is by tagging them. Once an object is tagged, it can be identified by computers. From then, physical objects can be included in the digital world. Radio Frequency Identification (RFID) tags [70, 72], Barcode scanners, QR codes (2D barcodes) [52], etc. are possible forms of tagging. Also sensors⁵ play an important role in digitalising information about the environment.

When we look at the state of the art of the *Internet of Things*, we notice that a lot of research has been performed in converting the physical environment into a smart environment. Diane J. Cook and Sajal K. Das performed an analysis in [17], on how “*smart*” our environments are. According to their findings, it is likely that our houses and other environments such as workplaces, stores, etc. will only increase in intelligence.

Despite the fact that smart objects and smart environments have gained a lot of attention, until five years ago, they never broke through commercially. According to Patrick Holroyd, Phil Watten and Paul Newbury [31], the lack of commercial breakthrough was due to both the complex installation and the lack of a user friendly interface. Subsequently, from their research we can deduce that the success of smart environments relies heavily on the development of new paradigms concerning user-centric data extraction and data presentation.

Inexpensive sensors, embedded processors combined with sensor fusion, networking and pervasive computing lie at the root for the advancement of smart environments and smart devices. As a result, we present an overview of the state of the art of sensors and context aware systems.

⁴Smart environment: “*a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network*” quoted from [77]

⁵Sensor: “*a device that implements the physical sensing of environmental phenomena and reporting of measurements. Typically, it consists of five components: sensing hardware, memory, battery, embedded processor, and transceiver.*” quoted from [67]

2.2 Sensors and Smart Environments

One of the first ubiquitous computing environments is *The Active Badge Location System* created by Ro Want, Andy Hopper, Veronica Falcão and Jonathan Gibbons [71]. The goal of their research was to efficiently locate and coordinate the staff of a large organisation. Instead of using the traditional phone or beeper, they made use of badges and a network of sensors that were deployed around the building.

By relying on the badges and the sensor network they gained a lot of benefits. Some benefits for receptionists were: easily tell whether a person is in or not, identify visitors of an organisation, etc. From then on, other researchers [7, 69, 37] also made use of sensor networks to create ubiquitous environments. Since sensors are often used to infer information about the environment, we discuss them in more detail. Afterwards we also describe the evolution and prospects of sensors.

2.2.1 The Evolution of Sensors

Sensing technologies are important tools for measuring physical phenomena. These technologies have progressed from clunky uncomfortable devices towards slim, ergonomic and convenient units. Since these sensors have become so compact, they are widely spread amongst us. When we look at our mobile phones, see Figure 2.1, we see that they already accommodate a range of sensors. Examples of sensors that are currently provided with (most) smartphones are the following: accelerometer, digital compass, gyroscope, GPS, microphone, and camera.

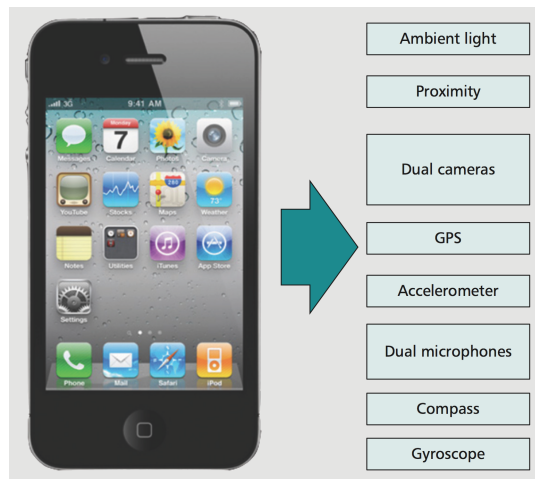


Figure 2.1: An iPhone 4 with its integrated sensors. [39]

In our opinion, it is likely to see that future smartphones will be extended with more advanced sensors. Nicholas D. Lane, Emiliano Miluzzo, et al. present in [39] how sensor-equipped mobile phones will revolutionise many sectors in healthcare, social networks, transportation, etc. The authors of the paper predict that the primary obstacle with sensors will not be the lack of infrastructure (because most mobile phones are already equipped with sensors), it will rather be the technical barriers that are related to privacy and the lack of resource sensitive reasoning with noisy data that may hinder the actual usage of sensor systems.

Another important requirement that has to be taken into account when working with a myriad of sensors is the need for a common interface for the sensors that serves as an access point. A good example of such an interface are the *phidgets* presented in [25]. The *phidgets* arose from a research project of Saul Greenberg and Chester Fitchett. The *phidgets interfacekit* provides a common gateway to access sensor data from a diversity of sensor types. Another initiative is *Arduino*⁶. *Arduino* is an electronics open source platform that allows common people (i.e. artists, designers, hobbyists, etc.) to create interactive objects or environments by providing access to a myriad of hardware (sensors, micro controllers, etc.).

Besides providing a software infrastructure to communicate with physical sensors, there is also the need to effectively distribute sensors in the environment. Previous research has indicated that sensor deployment in the environment has gained more and more attention. The authors of [54] present a method to identify the key sensors (i.e. most important sensors to detect an interest) in a smart environment.

Other researchers such as Emmanuel Munguia Tapia, Stephen S. Intille and Kent Larson take a different turn. During their research [63] they focus on how sensors could be used without becoming perceived as invasive. By abandoning traditional sensors such as cameras and microphones they were able to employ a sensor network without having to modify or damage the environment and this while keeping an accurate detection-rate.

In a more recent work by Stephen S. Intille, Jonathan Lester, et al. [32] is stated that the current technology behind sensors is emerging. Their study shows that we should not expect the development of fundamentally new types of sensors but rather sensors and devices which have an overall better performance. Besides this insight, they give an overview of the current trends in sensor technology such as: *raw data processing, multiple sensor data fusion, activity/context inference using statistical pattern recognition*, etc. Afterwards, the authors of the paper also

⁶<http://arduino.cc/> last accessed on 30/01/2014

provide the reader with some best practices and the future trends that could be expected for the following five years. Most of their predictions are concerned with the deployment of sensors and their physical characteristics.

2.2.2 Sensor Data Analysis

Most sensor devices provide us with raw data. This data on its own does not provide much knowledge. As shown in a survey of mobile phone sensing [39], sensor data has to be interpreted. Several approaches to interpret sensor data present themselves. Examples of sensor data interpretation are the following: *interpreting audio data* [41], *visual interpretation* [50], etc. Naturally, we can ask ourselves the following question: “*How will these techniques progress when the field of available sensors grows?*”.

By interpreting sensor data we allow applications to perceive information about their environment. An example of such an application is the one described by Jorge Cardoso and Rui Jos in [14]. In their research, they adapt the display of a public kiosk according to the current context. Another example is presented by Hans W. Gellersen, Albercht Schmidt and Michael Beigl in [24]. By making use of multiple diverse sensors, they create smart objects out of ordinary objects. The *Mediacup* was one of these objects. The *Mediacup*, as presented in Figure 2.2, is an ordinary coffee cup enriched with sensors, processing and communication.



Figure 2.2: The Mediacup. [24]

Unfortunately, understanding the physical context from sensors demands a new

proactive, automated model. Besides a model there is also a need for an infrastructure to cope with diverse concerns of context recognition such as the precision of the sensed context, battery life, computational resources, etc. An example of a platform which leveraged these challenges is Mobicon. Mobicon is a mobile context-monitoring platform, presented in [40].

When sensors are used to retrieve context, we enter the research field of context awareness. Context awareness can be provided by different approaches. However it is also possible to combine different approaches and fuse them together to fully enable context awareness. This was done by the authors of [6]. By combining the three key data streams (mobile, sensor and social), they made an effort at effectively creating context aware applications.

In Section 2.3 we will provide a more detailed explanation on context awareness. First, we give an overview on how sensor data can be shared and reasoned about.

2.2.3 Semantic Sensor Data

As with other types of datasources, the aspect of sharing and interoperability is important. Some propose semantic annotated sensor data in order to provide interoperability. As proposed in [56], they rely on the SSW-framework (Semantic Sensor Web) for adding meaning to the sensed observations to enable situation awareness. Besides the annotations themselves, this approach also provides a means to reason about heterogeneous multimodal sensor data.

Another approach to reason about sensor data is the one proposed in [78]. Their proposed framework is based on a semantic streams programming model. The framework allows non-technical users to execute queries over semantic interpretations of sensor data by using a declarative language.

We think that the progress of the semantic web⁷ will require more advanced frameworks that support the association of semantic data with sensor data. We believe that the ability of exchanging sensor data will have a positive effect on the progress of ubiquitous computing. Sensors should all be connected to the web with associated metadata. Furthermore, it must be possible to read and control the sensors remotely.

⁷Semantic Web [8]: An extension of the web where concepts such as reusability and sharing stand central.

2.3 Context Awareness

As already discussed in the previous section, sensors can be used to obtain contextual information about the environment. Therefore, it is important to know what is meant by the term “*context*”.

Context awareness is a term that arose from ubiquitous computing. A well accepted definition for “*context*” is the one introduced by Anind K. Dey in [19]. The presented definition of context will also be used throughout this thesis.

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves.”[19]

This means that context provides us with extra information about the current situation. This information can for example be used by applications to provide a better form of human computer interaction (hereinafter named HCI).

The following section will describe how researchers have embraced the concept of “*context*” in their respective fields of research.

2.3.1 There is More to Context than Location

It is interesting to see how researchers have incorporated the notion of “*context*” in their field of research. Fortunately this task was already performed in “*There is More to Context than Location*” [55], by Hans W. Gellersen, Albercht Schmidt and Michael Beigl. The authors of this paper provide us with facts about mobile computing research concerning context aware applications.

During their research, they have observed that location is the dominant parameter to deduce context in an application. Secondly they point out that the use of sensors may improve context awareness up to a certain level. They noticed that sensors provide either abstract context such as noise level, temperature, etc. or very specific context such as the user’s attention level.

Moreover, they state that most discussions on context awareness suffer from a general model that should allow comparison of the different approaches. Therefore, they introduce the following model concerning the structure of context:

- “A context describes a situation and the environment a device or user is in.
- A context is identified by a unique name.

- *For each context a set of features is relevant.*
- *For each relevant feature a range of values is determined (implicit or explicit) by the context.” [55]*

In our opinion, this model of context closely aligns with the characteristics of a sensor. A sensor in a network is also identified by a specific identification number. Furthermore, a sensor is able to retrieve a set of features instead of only one. A microphone can for example retrieve the pitch, noise, etc and this with a specific range determined by the device’s characteristics.

From this model they also describe the feature space to which context can be derived. The feature space is split into two parts namely, human factors⁸ and the physical environment⁹.

Using this model and the described feature space, the authors of the paper also developed two prototypes based on sensors to provide context. The first prototype was a light sensitive display and the second prototype an orientation aware PDA interface.

With respect to the process of deriving context, we share the same point of view. Context is best derived from multiple sensors in order to maximise the success-rate and to minimise the amount of “*false positives*”. Besides that, we also agree on the fact that the type of sensor should be chosen carefully in order to obtain contextual information. We also believe that applications can benefit largely from when they are presented with contextual information.

The next section will discuss the need for appropriate middlewares for the employment of sensors and context aware systems.

2.4 Middlewares for Pervasive Systems

This section discusses a set of middlewares that have been developed by researchers to facilitate the use of sensors and to retrieve context from sensor enriched environments. Before reviewing these different middlewares, we discuss the pa-

⁸According to [55], human factors are the collection of information on the user (knowledge, habits, etc.), the user’s social environment (group cooperation, social link, etc) and the user’s task (goals, spontaneous activities, etc)

⁹According to [55], the physical space can be subdivided into location, infrastructure and physical conditions

per of Billibon Yoshimi [9]. His work serves as an introduction to justify the need for appropriate middleware for sensors for pervasive systems.

2.4.1 Requirements for Middleware

Billibon Yoshimi studied the use of sensors for context-retrieval in [9] by examining the concept of location and situation awareness. First of all, he noticed that most system designers make the mistake of waiting until the end of their development phase to choose the appropriate sensors. He states that it is important for any system design process to examine the integration of sensors before starting the implementation of a prototype.

As already mentioned, he studied the subject from a low level point of view. Thereby, he divided the set of sensors into two major types. The first type of sensors are the *active sensors*. This type of sensor interacts with the environment and observes how its actions affect the environment. Examples of active sensor systems are: RFID tags, an infrared transmitter, a laser fluorosensor, etc. The second type of sensors are the *passive sensors*. Passive sensors are designed to make use of an external energy source to observe an interested object. Examples of passive sensors are GPS, ambient audio, etc. Both types of sensors have their advantages and disadvantages. Active sensors, for example, help to reduce ambiguities from passive sensors, while passive sensors are useful in situations where changing the environment is undesirable.

In most cases sensor data must be combined in order to get a unified view of the world. This phase is called *sensor fusion*. This phase requires expertise in knowledge engineering. Some methods that are at our service to perform sensor fusion rely on the following techniques: *bayesian networks* [38], *genetic algorithms* [21], *neural networks* [3].

During Billibon's research, a problem intrinsic to sensor system design was identified. The problem is as follows: when users interact with a computer they expect that a computer performs certain actions. On the contrary, when the users do not interact with the computer, but the computer performs an action, the users become confused. Since ubiquitous computing enables computers to perform actions autonomously, we believe that this problem may be directly linked to the success of ubiquitous computing.

This discovery implies that ubiquitous computing will require novel ways of HCI. Billibon proposes a solution to this phenomena. By creating an evolving user model we enable the system to make predictions of when a user has difficulties

when working with the interface. Based on this model, extra measures can be made to inform the user.

From this work we may deduce that, when working with sensors, we are also introduced to new sensing paradigms, architectural frameworks, etc. Sensor frameworks and context retrieval systems will help us in tackling these issues and challenges. The following subsection presents some available middleware that were developed to cope with these challenges.

2.4.2 Available Middlewares

In order to help us in the process of creating context aware applications and in particular by means of sensors, we will have to rely on architectural frameworks and middlewares. These will become the building blocks that will have to cope with challenges such as: available resources, distribution, reasoning, etc.

Therefore, we describe some middleware that were recently developed:

- Middlewares for Sensor Networks:
 - **Shared Phidgets** [42]: a software extension built upon the commercial Phidgets platform¹⁰ for rapidly prototyping distributed interfaces. The shared phidgets project allows developers to integrate distributed components without having to be concerned about the distributed systems aspects. This allows developers to leverage some of their concerns on the framework allowing them to focus on their main goals.
 - **SENSEI Real World Internet Architecture** [68]: a solution that provides an architecture that enables the foundation of the *Internet of Things*. They propose a framework that is responsible for leveraging distributed system aspects. Furthermore, they provide an architecture for efficiently integrating a variety of heterogeneous sensors to build a homogeneous framework to obtain real world information.

For more information on middleware concerning sensors networks, we refer to the work of Karen Henriksen and Ricky Robinson [30]. Their paper presents an analysis of the state of the art middlewares and highlights some open research challenges.

- Context-aware Middlewares:

¹⁰www.phidgets.com last accessed on 09/05/2014

- **Context aware Middleware Architecture for Smart Home Environments [29]:** a middleware that facilitates sharing contextual information of diverse sensors that are available in a smart environment. The middleware uses a publish/subscribe mechanism which allows software developers to share their context elements. The intention of this project is to allow developers to create more sophisticated context types by using the context elements that are shared by other developers.
- **Contextual Interfacing: a Sensor and Actuator Framework [28]:** the middleware makes the connection between sensors and actuators. Besides that, it also provides abstract and contextual information. The framework is a layered model that is built with the following concepts in mind: *synchronisation, aggregation, abstraction, and integration*. The components of this model are explained in depth in [28].
- **Context-aware Middleware for Pervasive and Ad Hoc Environments [60]:** a middleware architecture that supports both pervasive and ad hoc computing. The approach is based on a number of component frameworks that allow pervasive dissemination for both real time systems and normal systems. Even more, the authors of the paper present a *sentient object model*. This model presents the key requirements for enabling both contextual awareness behaviour as autonomous intelligent behaviour.

Recent research with regard to context aware middleware has resulted in several requirements wherefore middleware must comply to. These requirements were explored in [34].

2.5 The Evolution of The Internet of Things

The *Internet of Things* and smart environments have been researched for a reasonable period of time. Therefore many prototypes and applications have arisen over time. We present an overview of this evolution by reviewing a research project of 2001, where they developed a smart environment. Next, we present three commercial *Internet of Things* devices that have been developed over the past two years.

2.5.1 Analysis of a Research Project: Smart Kindergarten

In 2001, Mani Srivastava, Richard Muntz and Miodrag Potkonjak have developed a smart kindergarten [62]. The kindergarten was empowered with sensors with the

intention to create a problem solving environment for early childhood education. Figure 2.3¹¹ and Figure 2.4¹² present the sensor enhanced table and badges for the children.

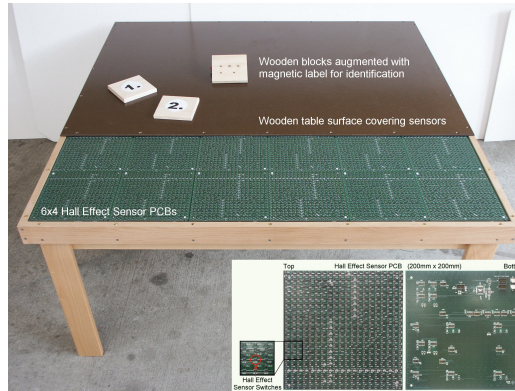


Figure 2.3: Smart Table

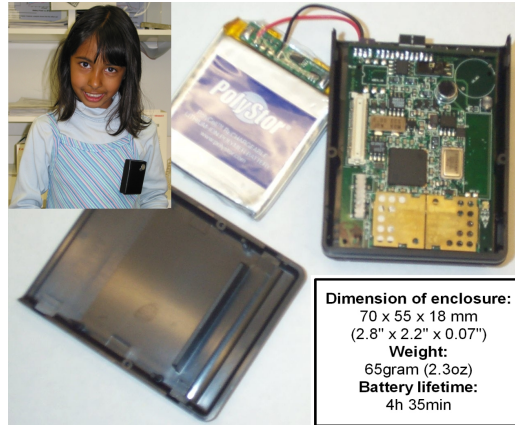


Figure 2.4: iBadge

The smart learning environment is developed to enhance the learning process of children on an individual level by adapting the environment towards each child

¹¹http://nesl.ee.ucla.edu/projects/smartkg/photos/SmartTable_Complete.jpg last accessed on 09/05/2014

¹²http://nesl.ee.ucla.edu/projects/smartkg/photos/iBadge_Complete.jpg last accessed on 09/05/2014

individual needs and by adapting to the current context. Furthermore, this smart environment is not only useful for children. The teacher was also able to unobtrusively evaluate the learning process of the child and this at a continuous rate. Their solution is built upon smart objects that take the shape of sensor-enhanced toys and classroom objects. These sensor-enabled devices are connected through a specific middleware service that stores the retrieved information in a database.

Unfortunately, the authors did not mention in any way how the teacher is presented with the information. We can only assume that the teacher was limited to the data presented by the database viewer. This approach makes it a hard process for “*non-experts*” to access, manage and process this information.

From this project, we may conclude that the smart environment helps children and their teacher in the learning process. By employing sensors over the environment and physical objects, computers are able to collect a large amount of data. However, if this data is not filtered or is not presented efficiently to the teacher, it will be difficult to make use of the benefits of their efforts. Therefore, we can conclude that they did not pay much attention to the presentation of the collected data.

We believe that their solution could benefit from an interactive display. This display could present the filtered information in an intelligent way. Instead of presenting a large amount of data to the teacher, a computer could decide autonomously which information should be shown in which situation.

2.5.2 Commercial Solutions

- **Nest Thermostat:** Nest¹³ is a company bought by Google and is viewed as the entering point for Google towards the *Internet of Things* and to a “*connected home*”. Nest’s top product is their “*smart*” thermostat. According to several sources¹⁴ this device is probably the most popular *Internet of Things* device at the moment.

¹³<https://nest.com/thermostat/life-with-nest-thermostat> last accessed on 2/02/2014

¹⁴<http://www.economist.com/blogs/schumpeter/2014/01/google-and-Internet-things>
http://www.slate.com/blogs/future_tense/2014/01/14/google_nest_acquisition_Internet_of_things_is_next_frontier_for_data_machine.html
<http://www.theguardian.com/technology/2014/jan/20/google-nest-aquisition-learning-thermostat> last accessed on 03/02/2014

After installing and using the thermostat, it will learn the routines of the homeowners. Afterwards, the device will be able to adjust the thermostat independently. Another feature of the device is its ability to save energy.

One of the most useful things about Nest's thermostat is its mobile application (see Figure 2.6). This application provides a user friendly interface that allows the homeowners to view the thermostat's settings and adjust them accordingly. An extra advantage of Nest's thermostat is that it seamlessly integrates into our homes with its discrete design (see Figure 2.5) without becoming intrusive. The thermostat works autonomously without disturbing the homeowners.

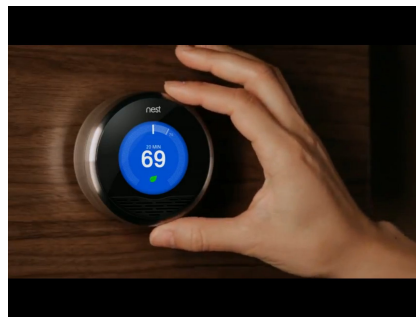


Figure 2.5: Nest Thermostat's discrete design

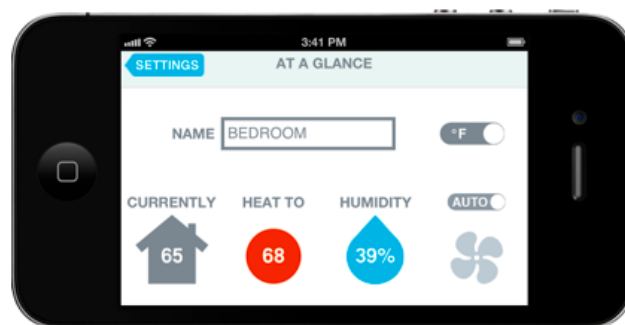


Figure 2.6: Nest Thermostat's mobile application

Still, another important aspect that we want to emphasise is the clear presence of an interface on top of the device. Therefore, the device does not become fully “invisible” to the homeowners. There is still a way to consciously interact with the thermostat. Since homeowners are still aware of

the presence of the electronic device, it does not disappear in the background as envisioned by Mark Weiser.

- **Philips Hue:** Hue¹⁵ is a personal lighting system that enables users to manage the lighting of their house by using their mobile device. Hue is less intelligent as Nest's thermostat, as it does not regulate the lighting autonomously. Rather, it relies on the user to create an illumination schedule.

The most important feature of Hue is the possibility to change the illumination while you are not at home. Furthermore, Hue also has the option to create relationships between an illumination schedule and one of your photos or events. Just like Nest's thermostat, Hue also has a mobile application which allows you to manage the smart device. In contrast to Nest's thermostat, the “*digital*” lamp (see Figure 2.7) does not have an interface of its own, nor does it collect information about the environment.

The lamp merges well with the environment, as a regular lamp would do. However, it still requires human interaction in order for it to work properly.



Figure 2.7: Philips Hue lamp and mobile application

- **Lockitron:** Lockitron¹⁶ is a kickstarted project that allows keyless entry to the user's home by using a smartphone and a smart lock (see Figure 2.8). Besides opening and locking doors, it also sends notifications when for example someone knocks on the door or when a child opens the door using their own phone. Another advantage of Lockitron is the ability to grant access to other people when you are not at home. Lockitron relies

¹⁵<http://www.meethue.com> last accessed on 02/02/2014

¹⁶<https://lockitron.com> last accessed on 02/02/2014

hardly on the Bluetooth abilities of the mobile device that controls it. Their latest product uses Bluetooth 4.0¹⁷ that enables people to open their doors without having to manually unlock it. Just like the two previous *Internet of Things* devices, this smart device is managed by using a mobile device such as a tablet or a smartphone.



Figure 2.8: Lockitron's lock and mobile application

From the aforementioned smart devices, we can deduce three factors to which they owe their success. First of all, we notice that all of these artifacts are employed in the home environment without invasively modifying the house and without too much physical effort. Secondly we notice that all of these devices rely on a mobile application. The mobile application enables users to truly benefit from the devices's capabilities i.e either to manage the smart artifacts or to present information that they have collected. Thirdly and maybe most surprising, is that all the aforementioned devices have completely replaced their "*predecessor*".

The traditional lamp, thermostat and locks were completely removed from the environment and replaced by their successor. In other words, we can conclude

¹⁷Bluetooth 4.0: a new version of Bluetooth that focusses on low energy consumption

that these environments were made “*smarter*” by replacing them instead of upgrading the existing devices. Naturally, we can imagine scenarios where sensor deployment in environments will still be more advantageous, for example when artifacts have an emotional value, are expensive, etc.

Based on these smart devices, we can make a distinction between two types of *Internet of Things* devices. We have noticed that there exist artifacts that are connected to the digital world, that do not act “*intelligent*”. We assign these artifacts to a first category of *Internet of Things* devices.

The Hue lighting system is part of this first category. The Hue Lamp only performs an action when they are controlled by the user. They do not perform any action autonomously, therefore we can not consider them to be truly “*intelligent*”.

A second category of *Internet of Things* devices are the artifacts which perform “*intelligent*” actions. Both Nest’s thermostat and Lockitron’s locks are examples of this second class. In a way, we can consider them to be smarter than the Hue lamps, as they are able to perform actions autonomously (raising the thermostat as temperature drops, closing doors when leaving a room).

To our opinion, both categories have earned the title of *Internet of Things* devices. However, the devices that belong to the second category i.e devices that act “*intelligent*”, are a degree higher in reaching the purposes of ubiquitous computing or even more the purpose of ambient intelligence as described in Section 2.6.

2.5.3 Open Research Problem in Ubiquitous Computing

Both the development of *ubiquitous computing* and *Internet of Things* create huge opportunities towards economics and individuals. As highlighted in Section 2.2, the physical environments will change drastically. Physical objects will be tagged creating a smart environment. Sensors, together with tagged objects, will become the entering point for computers to our physical world.

However, since computing will happen everywhere and anywhere, we are presented with a large amount of data. As a result, we may expect that we will require innovative ways to manage this data overload. From reviewing three recent *Internet of Things* devices, we noticed that it is important for the user to have an intelligent interface that forms a bridge between the user and the collected data.

In order for computers to determine what information is relevant for the user and what is not, the computer must act intelligently. The following section will present

a new paradigm that makes an effort in solving this problem, namely, ambient intelligence.

2.6 Ambient Intelligence

In the previous sections, we have introduced the concepts behind ubiquitous computing. We have emphasised on the technologies (sensors, context aware systems, wireless networking technologies, etc.) that enable ubiquitous computing. Unfortunately, we cannot say that all problems concerning this concept are resolved. Issues such as *data access* and *usability* were left unaddressed. The ambient intelligence (hereinafter named AmI) [53, 16, 5] paradigm is an initiative that makes an effort to take both accessibility and usability into account.

In a way, AmI can be seen as the successor of ubiquitous computing. Juan Carlos Augusto defined AmI as the confluence of ubiquitous computing and artificial intelligence [4]. To their opinion, we may not confuse AmI with other areas such as ubiquitous computing, sensors, wireless networks and artificial intelligence (hereafter named AI), but rather look at it as a fusion of these areas. Whereby, AI is seen as encompassing areas such as agent-based software and robotic. Juan Carlos Augusto illustrates his point of view on AmI as follows, see Figure 2.9:

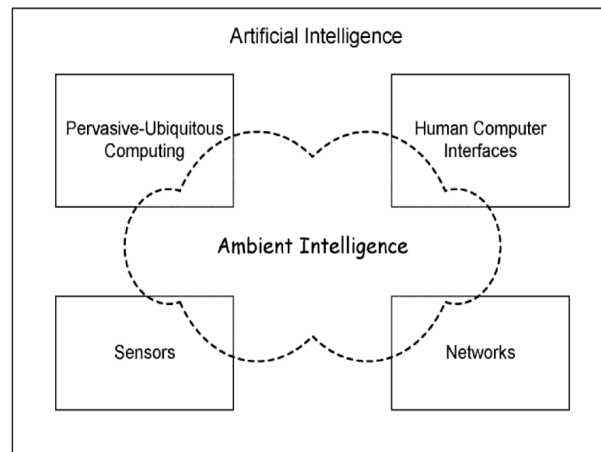


Figure 2.9: Relationship between AmI and other areas [4].

When we look closer at the definition of ubiquitous systems and AmI we see that there are some alignments and differences. Ubiquitous systems are built to emphasise the physical presence and availability of resources while missing an essential element, namely, the explicit requirement of “*intelligence*”. When we

look at the definition of AmI given by Raffler in [49], we notice that this idea aligns with the concept behind ubiquitous computing:

“A digital environment that supports people in their daily lives in a non-intrusive way.”

This definition was slightly modified by Juan Carlos Augusto. His definition emphasises on the the essential element of intelligence as a fundamental component of AmI systems:

*“A digital environment that **proactively**, but **sensibly**, supports people in their daily lives.”*

The terms “*proactively*” and “*sensibly*” represent the “*intelligence*” requirement that could help ubiquitous computing in becoming truly useful. As discussed in Section 2.5.3, current realisations of ubiquitous systems can easily become unusable because computing will appear everywhere and anywhere. This will result in smart environments that will compete for the user’s attention, resulting in an information overload that may overwhelm the user.

The overload of information may tempt the user in turning down the computing system or just ignore it, losing its purpose. Therefore, the element of intelligence will be responsible for mediating between the user and the services provided by the ubiquitous environment.

The following section will explain how digital documents can become intelligent agents that act as mediator between the user’s needs and the data obtained from ubiquitous environments.

2.6.1 Ambient Documents

The advent of electronic media has opened opportunities that would never be possible with traditional paper documents. For a large extent, the World Wide Web has been responsible for challenging the properties of traditional documents i.e. static document consisting of text and/or images.

Over the years the document metaphor has been extended to include multimedia items such as video, graphics, audio, etc. Since we were no longer restricted to media that should be rendered on paper, digital documents have not only changed the way we think about documents, but also how we create, view and interact with them. As a result of this emergence, researchers and companies have thought about ways to reinvent the traditional document metaphor with the intention to

create a “*true digital document*”.

The contribution of this thesis lies in expanding the AmI paradigm by enabling the concept of “*ambient documents*”. Ambient documents are an extension of the traditional digital document metaphor. Gregory M. P. O’Hare, Michael J. O’Grady, et al. describe ambient documents as follows:

“Documents that offer a dynamic and radical view of document access, production and dissemination.” [47]

The authors of [47] consider these documents as an intelligent subclass of traditional documents. They describe the documents as intelligent artifacts that make use of a suite of intelligent techniques for content identification and filtering to cater current and future needs of the user.

To their opinion, if digital documents want to become truly useful in ubiquitous environment, we need to radically change the assumptions we have with traditional documents. Furthermore, the authors also present the following five assumptions that must be challenged and overcome with ambient documents:

- “*Document retrieval is based on user request (pull technology);*
- *Documents are generic;*
- *Document are static;*
- *Document are primarily comprised of text and images;*
- *Document content is always regulated.” [47]*

Unfortunately, to our knowledge, the writers of the paper have never made an effort to investigate this path nor did they create a prototype of their ambient documents. However, they did provide the following seven characteristics to which an ambient document will be subjected. We made the effort to describe them in more detail:

1. Document Retrieval

Traditional digital documents rely on a pull model i.e the user independently searches and opens the document. However with ambient documents this model will be extended towards a pull/subscribe model. This model allows for documents to be pushed towards the users according to certain aspects of the user’s context. Even more this pull/subscribe mechanism can also be domain dependent.

2. Document Audience

Most of the time, documents are written for a general audience (i.e digital documents present the same information regardless of who the reader is). Although this may seem a good approach for some cases, it also contains some significant disadvantages.

Imagine that you have created a document that contains some personal information and you want to exchange it with a colleague. Then you have two options: Option 1) Exchange the document with the personal information with the risk that it can be misused, or Option 2) Modify the existing document which is error prone and takes time. With ambient documents this problem will not occur. As documents are personalised for their audience, it can filter the sensitive information.

3. Document Content

Ambient documents can be seen as dynamic objects that can change over time. The content that is presented depends on the current context. A document may change its content according to the needs of the user and may depend on the situation (e.g environment, task, user, etc.).

4. Document Media

Images and text are the most common type of media for traditional documents. The WWW has changed this assumption by allowing various types of multimedia elements such as audio, video, 3D objects, etc. However, ambient documents will extend the assumption even more by allowing more media types such as real time sensor information, web service data, etc. Furthermore, the selection of media elements will highly depend on the user's context. For example if a user wants to read a document while driving a car, it is likely that he is not capable of performing the act of reading. As a result, the documents could intelligently decide to change the media representation from text to audio.

5. Document Ethos

Traditional documents are static in the way they present the content and how they are structured. Ambient documents are unregulated towards these properties. Forums are a perfect examples of non traditional documents, as their content is provided continuously and unpredictably while their structure adapts accordingly.

6. Document Repository

Ambient documents are not restricted to a central repository, such as a digital library, for retrieval. Ambient documents can be deployed on devices

with limited computational capabilities such as sensors.

7. Document Presentation

Ambient documents are not restricted to standalone computers with a WIMP interface. Neither are these documents restricted to particular interaction devices such as a mouse and a keyboard. Ambient documents can be viewed and accessed in a multimodal fashion.

2.7 Scope of the Thesis

In this chapter we have discussed two concepts, namely ubiquitous computing and *Internet of Things*. Along with these concepts we have discussed the different technologies that are necessary to enable them.

We gave a detailed overview of how sensors can be used and what we might expect from them in the future. Next, we discussed how sensor data can be used by computers to derive context, along with some relevant examples and available middlewares. This has shown us that these technologies are more or less mature enough to be used in applications.

However, we came to the conclusion that there are still some open challenges with ubiquitous computing such as data access and usability. In order to solve these challenges, an initiative was proposed by researches namely, ambient intelligence, a confluence of ubiquitous computing and artificial intelligence.

Along with this concept we have discussed the idea behind “*ambient documents*”. These documents are intelligent agents that provide the right information at the right time by making use of collected data from the environment. However, despite their interesting proposal they never investigated this trail.

We believe that this thesis contributes to the era of ubiquitous computing and even more to the era of ambient intelligence, by investigating the development of these visionary documents. We believe that by enriching documents with dynamic content, we can obtain ambient documents. Since we have evaluated the current state of the art on sensors and context awareness, we are able to create dynamic content that makes use of these technologies and concepts.

The following chapter will review the state of the art on dynamic documents along with an analysis of representative document formats with regard to their support for dynamic content.

Chapter 3

Review of Dynamic Content in Digital Documents

This chapter discusses the research training of this thesis and is divided into two main sections. In the first section, we discuss the related literature of this thesis. We consider several attempts in empowering digital documents with dynamic content. Besides summarising their approach, we provide a critical analysis for each solution and the relations amongst them.

In the second section, we analyse a set of document formats. This analysis reviews each document format with regard to their ability to specify dynamic content.

For both sections, we present a conclusion that discusses the current state of progress and we elaborate on the problems at hand.

3.1 Analysis of the State of the Art

Over the past two decades, attempts have been made by both researchers and companies [10, 65, 13, 33], to develop so-called “*dynamic documents*”¹⁸ i.e digital documents that contain dynamic content. Their research has resulted in a myriad of diverse perspectives on which kind of functionalities these documents must comply to.

These different schools of thought have become the ground for arguing on the term “*dynamic documents*”. Since most researchers had their own perspective on the subject, they were driven towards alternative concepts, resulting in sev-

¹⁸The term *dynamic document* was first introduced in [61] to describe documents that have a set of associated behaviors (activities)

eral naming conventions, such as: *active documents* [65], *live documents* [74], *multivalent documents* [48], etc. Since this thesis makes an effort in empowering digital documents with dynamic content, it is recommended to explore some of these approaches in more detail. Furthermore, by reviewing these different schools of thought on dynamic documents, we can easily verify the generality of our solution.

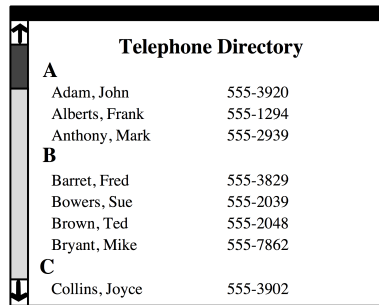
Since the concept of dynamic content has been studied for more than twenty years, we present our review in a chronological order to clarify the evolution.

3.1.1 Active Tioga Documents - 1990

In 1990, Douglas B. Terry and Donald G. Baker focussed on the creation of digital documents whose content was derived from external data sources. In their opinion these external data sources can be very diverse, ranging from databases to continually changing sources such as clocks, weather forecasts, etc. The justification behind their research was the fact that documents, containing information from a source that changes over time, should also update accordingly and this without any form of human intervention.

Their research resulted in two paradigms which they implemented using a document editor, named Tioga. Their editor and their approach can be found in [65]. The first paradigm that they employed involves dynamically computing the content of a document as it is displayed. Their second paradigm is based on a notification system that informs interested applications of the changes made to the document.

Instead of going straight into the details of their approach, it is advised to know how the Tioga editor manages documents. Tioga documents are logically structured as a tree of nodes that present some text of the document. An example of such tree and its output is presented in Figure 3.2 and Figure 3.1, respectively. This tree contains the nodes of a telephone directory document. Besides the text that a node represents, they are also able to obtain properties in the form of key-value pairs.



Telephone Directory		
A		
Adam, John	555-3920	
Alberts, Frank	555-1294	
Anthony, Mark	555-2939	
B		
Barret, Fred	555-3829	
Bowers, Sue	555-2039	
Brown, Ted	555-2048	
Bryant, Mike	555-7862	
C		
Collins, Joyce	555-3902	

Figure 3.1: Generated document representation of the telephone directory document [65]

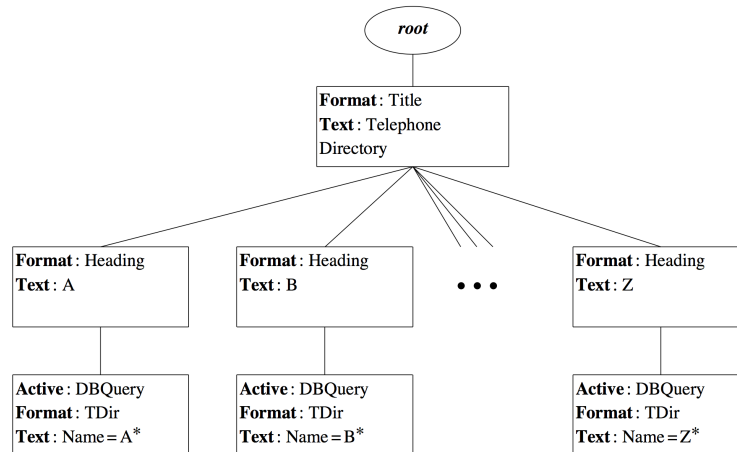


Figure 3.2: The logical structure of a document representing telephone numbers with dynamic query nodes [65]

Dynamic content is represented with a node and must contain a property named “Activity”. The associated value of this property indicates the *class* of the activity. So instead of keeping the activity’s code in the node itself, the property shall keep a reference towards the associated activity class. This class will be responsible for dynamically computing the content. The implementer of such an activity class must define two procedures, namely, a transform procedure and a size procedure. An example of such two procedures is presented in Figure 3.3.

```

Transform: PROCEDURE [current: Node] RETURNS [new: Node];
Size: PROCEDURE [current: Node] RETURNS [size: INTEGER];

```

Figure 3.3: Procedures constituting a node transformation activity [65]

The current textual content of the node is used as input by the transform procedure to produce the new content for the node. The size procedure is responsible for estimating the desired size of the transformed node. Note that the size procedure makes an estimate of the size. This is an important design choice, since no actual transformation must be performed for this task. As a result, their solution obtains the ultimate form of lazy evaluation of the dynamic content. This means that no computational power is required until the dynamic content is shown to the user.

The Tioga editor relies on the “*TextNode*” module for executing the associated code. This module performs on the lowest level of the Tioga editor and manages the tree of nodes for a document. The functionality of the *TextNode* module was extended in such a way that it triggers the activities that are associated with the nodes. It is important to note that these activities are only triggered when the content of the node is requested. Therefore, we may conclude that documents that contain dynamic content will be able to exploit the advantages of a lazy evaluation.

A second dynamic document mechanism that was implemented was motivated by the idea that documents should be transparent towards their external sources. This means that external sources should be notified when a user modifies content of the document that is linked with an external source. An example could be changing some database data in a document that was obtained from an external database. In the authors opinion, edits to the document should be propagated towards these external sources. This mechanism introduces a two-way interaction between the document’s dynamic content and the external sources. Naturally, we can imagine that this two-way interaction may also have some disadvantages such as: unintentionally changing the external source, data corruption, etc. As a result they have opted to keep a list of changes made to the content. These modifications are only propagated when the user saves his edits. Another option that they provide is the ability to reset the document. The reset functionality of the document restores the original content and leaves the external sources untouched.

To implement the second dynamic document mechanism, they also relied on the Tioga editor. This implementation is based on a *edit notification dispatcher*. The dispatcher is responsible for keeping track of all the edits made while working on the document. Afterwards, the dispatcher will be responsible for propagating the modifications towards the interested applications.

Critical Analysis: During their research they have created several of what they call “*dynamic documents*”. After examining their work, we may conclude that they have two perspectives on what dynamic content is about. On one hand,

dynamic content is content that transforms according to a particular predefined procedure. On the other hand, dynamic content is content that is interactive i.e the document has become the interface for external sources.

In our opinion, these two paradigms seem acceptable but they lack expressiveness. Their solution only supports a limited set of the possibilities that could be possible with dynamic content. Additionally, their solution is a pure ad-hoc implementation that will only work with the Tioga editor. Therefore, it will be difficult to apply their approach to other document formats or editors. Apart from that, we agree that a document could benefit from the notion of *actions* that change the content/structure of the document.

We also agree that documents could become the interface for external resources. However, by enabling digital documents to modify the data of external resources, we can easily violate data integrity. As a result, it will be a complex process to enable such a feature.

3.1.2 Multivalent Documents - 1996

Another approach towards dynamic content in documents is the one taken by Thomas A. Phelps and Robert Wilensk in [48]. According to the authors, “*true digital documents*” should provide an interface to potentially complex content. Therefore, they introduce a novel solution, named “*multivalent documents*”. According to the authors, multivalent documents would become the new general paradigm for the organisation of complex digital document content and functionality. In order to accomplish their vision, they propose a model with an object-oriented perspective on digital documents.

According to the authors, these documents are particularly interesting for documents containing complex content with a variety of interaction styles. Authors of such documents should divide their content into different layers. These layers are bound to certain *behaviours* that define when a corresponding layer should become activated. A simple example of a layer type are annotations in a PDF document. An annotated document has two views: either we enable the annotations which makes it possible to see them, or we have the possibility to deactivate them, leaving us only with the original document. A more complex use of these layers is for example an image with an additional layer that contains the OCR result¹⁹ of the image. This would allow the user to see both the original image and

¹⁹OCR (Optical Character Recognition): a technique that converts the characters on the page into ASCII text.

give the user the possibility to select, copy and paste a certain geometric region of the scanned image. This would be very useful when the original image is of a particular value.

Critical Analysis: This solution defines behaviours that are the glue to bind the different components and their associated layers together. The combination of the different layers forms a single coherent document. If we compare this approach to the one mentioned before and especially the first paradigm, we may conclude that besides some differences they also share some similarities. Both types of dynamic documents adapt their visual appearance of the document towards a particular interest. The first type of dynamic documents proposed in the Tioga editor, merely updates the current content. While on the other hand the multivalent documents are about activating a particular layer, where layers provide the user with the most appropriate information in combination with the associated interaction possibilities.

To our opinion, this approach is too limited. We believe that the opportunities of dynamic content can go beyond the concept of changing layers. For instance, their approach does not support the integration of data that is able to change over time. The layers merely consist of static parts that are replaced when necessary. Furthermore, their approach resulted in an ad-hoc solution that is not easily extended to support other features.

3.1.3 Stick-e Documents - 1996

Until now, none of the discussed solutions have taken the physical environment into account. From our background research in Section 2.3 we can conclude that over the last years, a lot of research has been performed on pervasive environments and context-aware systems. This field of research also influenced the exploration of digital documents. The stick-e document presented by Jason Brown Peter, in [10] is such an example.

A stick-e document acts as a container for smaller components which are called stick-e notes. A stick-e note consists of two main components, namely, its content and the context. The author of the paper has chosen to represent stick-e documents in the SGML²⁰ format to make them portable. The following code snippet demonstrates how a stick-e note is encoded:

²⁰SGML (Standard generalised Markup Language): a platform independent ISO-standard that describes the syntax of markup languages for documents (<http://www.w3.org/MarkUp/SGML> last accessed on 16/04/2014)

Listing 3.1: Example of a Stick-e note in SGML

```
<note>
  <required>
    <with> Jochen Francois <or> Gus Windey
    <at> <long>38.5323</long> <lat>77.0027</lat>
    <content>
      This content adapts to the user
</note>
```

Wherein the `<required>` tag defines the context and the `<content>` tag defines the content that must be presented for that particular context.

The key to create a context-aware document is by creating a relation between a contextual element and a stick-e note. When a contextual element occurs, the associated stick-e note is displayed. This approach is inspired by the idea of AIR (Activity-based Information Retrieval). This specialisation of AIR differs from other approaches as it focusses on the future instead of the past events for retrieval.

Critical Analysis: When we compare this approach to the ones of the previous solutions, we notice that it is the only one that takes the context into account to change the content of the document. This is an interesting approach since the age of ubiquitous computing will collect an enormous amount of data about the physical environment. This vast amount of data can then be used to the benefit of document authors in creating digital documents that intelligently adapt themselves to the needs of the reader.

We believe that future documents can really benefit from the notion of context. Therefore, we consider the context as an essential requirement for which dynamic content should provide support for.

However, the downside of this solution is that they do not consider the support for content that may change over time, such as: data from a web service, sensor data, etc.

3.1.4 Live Documents - 2002

Software documentation is associated with computer software to explain how the software operates or how it should be used. Writing software documentation is

a daunting task since it should always represent the current state of the software. Therefore, one would rather have this task automated.

Anke Weber, Holger M. Kienle and Hausi A. Müller also noticed the need for documents that rewrite themselves autonomously. As a result, they introduce the concept of “live-document” [74]. They describe live documents as follows: data driven, context-aware, interdynamic and adaptation centric.

Note that their adaptation characteristic is not associated with the context awareness characteristic of the document. The adaptation characteristic deals with the fact that a document should autonomously rewrite itself towards multiple output versions and formats coming from a single source, see Figure 3.4.

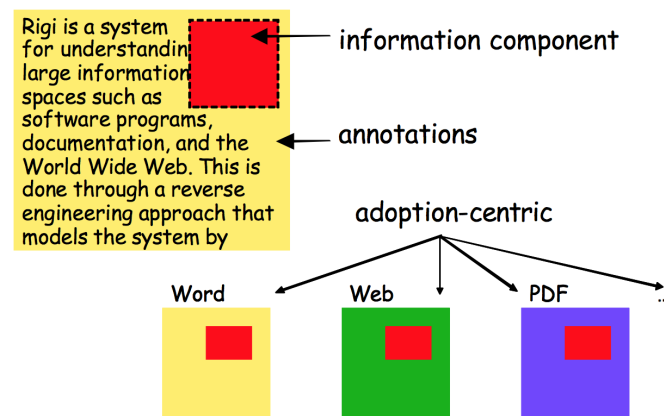


Figure 3.4: LiveDocuments adaptation characteristic [74]

During their research, they have developed a prototype document that presents software documentation in real time. This document is interlinked to the actual software in order for it to rewrite itself by the so-called “live links”. The implemented prototype was developed by making use of SVG (Scalable Vector Graphics)²¹ and Microsoft Office Automation²².

Live links are necessary to detect changes in the source code. Afterwards, these updates are reflected in the content of the document. Unfortunately, the authors of the paper do not explain how these live links work.

²¹<http://www.w3.org/Graphics/SVG> last accessed on 16/04/2014

²²[http://msdn.microsoft.com/en-us/library/ms173024\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/ms173024(v=vs.90).aspx) last accessed on 16/04/2014

The authors of the paper also present eight requirements to which live documents should conform. To our opinion these eight requirements are also useful beyond their solution. Therefore, we will discuss them briefly: The first requirement is that live documents should have a state, the document itself should keep track of the modifications of its state. A second requirement is that live documents should manage their own state autonomously. This means that when a documents represents something that has changed over time, it should adapt its state according to these changes. The third requirement is oriented towards reusability. The elements from which a live documents exists, should be reusable on an individual level.

They use the term “*reusable*”, not in the sense of a copy/paste operation, but rather in the sense of *transclusion*, introduced by Ted Nelson in [44]. Ted Nelson assumes that documents or parts of documents should be reused by means of a reference, see Figure 3.5.

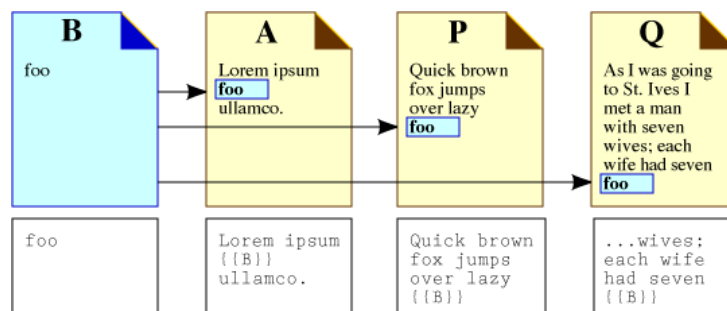


Figure 3.5: Document B is “transcluded” in document A, P and Q ²³

The fourth requirement is the ability to adapt intelligently to the presentation media. An example of such use could be adapting the document when it is printed on paper and contains a video element. Then an image could be shown instead of the video. The fifth requirement deals with the ability to adapt the content towards the interest of the reader. The sixth requirement is about information visualisation. The document should adapt the visualisation to facilitate content comprehension. The seventh requirement involves the ability to search the document beyond the content of the document itself. This is possible since the document has access to external sources. The last requirement is that living documents should support scripts. These scripts should be used for repetitive tasks such as going back to the previous state of a document.

²³<http://elinux.org/Help:Transclusion> last accessed on 3/02/2014

Critical Analysis: The notion of Live Documents is very interesting since it provides support for both context awareness and changing data. However, their proposed solution and the development of their prototype lacks precise detail on how they were achieved.

In our opinion, we would have preferred another prototype. In their implementation they make use of links between the software code and the document. Nevertheless, the authors do not mention how these links are specified. As a result, we cannot know how much effort is needed to create such documents. We would rather have seen a prototype of a document that is used by regular people. Then they could present how a regular user can specify links between the document and external sources.

Still, their vision of taking both context and changing data into account is an interesting trail for further exploration.

3.1.5 Active Documents - 2003

When we focus less on a specific implementation, we can conclude that documents with dynamic content can and will change the way enterprises work. This point of view was discussed by Joshua Duhl and Susan Feldman, in [33].

In their research, they did not develop a prototype or any other implementation of such kind. However, since the authors are more commercially oriented, we find it interesting to discuss their point of view on the subject. In their work, they define the role of these active documents as follows, we quote:

“Active documents change the role of information and how it interacts with both software applications and people.” [33]

Their vision is based on the thought that specific applications may gradually disappear over the years. As a result these application-specific functions will be integrated into what they call “*active documents*”. By doing so, they will affect business processes and workflows. It will change the way in which companies currently share and exchange their information. As a result the separation between dynamic documents and specific programs will disappear. This will lead to an advent of a new category of applications which will be responsible for integrating systems, analysing information and personalising content.

When giving form to their opinion, they try to define what an active document should be. In their quest of searching *the right definition*, they stumble on the

same problem as we did, namely, not finding a complete definition of dynamic documents:

“Active documents are so new that there is no standard for what they look like or what they do.” [33]

However they make an effort to define the broad lines of what it should be. Their vision on dynamic documents finds its origin in the fact that it should be active in its own representation, interaction and presentation. According to the authors, this could be achieved by adding code and actions to the document programmatically.

Critical Analysis: We agree with the idea that digital documents with dynamic content can take over some functionalities provided by a software application. However, we believe that their proposed approach for achieving this is unacceptable. By imposing the requirement for document creators to have some limited programming knowledge, a lot of people are excluded to create dynamic documents.

In a way, their solution is comparable with the last requirement of the Live Documents approach. This solution also relies on scripts to make the document more dynamic. We believe that the authoring process of a document should be as simple as possible. By imposing the skill of programming, we do not obtain an authoring process that is user friendly. Therefore, we can conclude that the editing process of digital documents with dynamic content will bring along some challenges. These challenges must be addressed in order for regular people to adopt these envisioned digital documents.

3.1.6 Minerva Documents - 2005

Markus Reitz and Christian Stenzel consider dynamic documents as the next generation document technology. According to them, these documents come to the aid of the problems endured when working with the current information distribution techniques. In their paper [51] they propose *Minerva*, a testbed for dynamic documents and applications. One of the first remarks they make during their study is the following:

“With documents there is always one core concept that remains unchanged, namely, the fact that every component of the document is static i.e. the content always represents a snapshot taken by the author during the creation of the document.” [51]

According to Markus Reitz and Christian Stenzel, the static property of digital documents forms the ground for two problems. Firstly, the documents are stateless. Therefore, it becomes impossible to show up-to-date information to users. Secondly, the interaction possibilities between the user and the document viewer are limited.

The goal of their research is to overcome the aforementioned limitations in order to escape a snapshot system wherein documents stand still in time.

During their research, the authors of the paper make the remark that the domain of dynamic documents undergoes a flow of constant fluctuation. They note that only a few frameworks exist which allow researchers to experiment on the concepts of dynamic documents. In order to cope with these constant changes, the authors propose their own testbed.

Their proposed testbed acts as a framework for rapid prototyping of new concepts on dynamic documents. A key element of their testbed is extensibility. Extensibility is provided through the following characteristics of the framework: flexibility, modularity and component-based.

While examining the notion of dynamic documents, they follow the paradigm of “*the network is the document*” as presented in Figure 3.6. As a result, documents are constructed on the basis of components found in the network of information. Furthermore the components in the network can also interact with the environment. This can lead to changes in both the content and the navigational behaviour of the document.

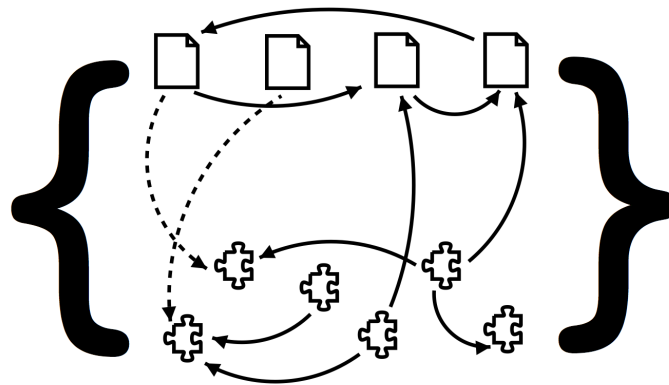


Figure 3.6: The network is the document [51]

During their research, they give an in-depth look at the shortcomings of classic documents technology. Furthermore by analysing these shortcomings, they propose a list of required properties to which a dynamic document framework should comply to. Since these requirements are of help during our research, we discuss them here in more detail:

- **Polymorphism:** Documents have a stateful representation depending on the current user and the history of use.
- **Statefulness:** This property helps conserving polymorphism towards different user sessions and user groups.
- **Non-linearity:** Using the document according to personal preference, instead of following a predetermined path through the document, which was defined during creation by the author.
- **Web of components:** The documents exist out of a web of interacting objects combined together to form a coherent document.
- **Environment awareness:** The building blocks of the document become aware of their environment. Corresponding actions are triggered when certain phenomena in the environment occur.

The rest of their paper is centred around the implementation details of their framework and how the testbed could be used in practice. In the end they also included a brief evaluation accompanied by their conclusions.

Critical Analysis: Personally, we think that this point of view on dynamic documents aligns the most with ours. We agree that digital documents should not stand still in time. They should be able to adapt themselves after the point of creation and with or without human intervention.

While we reviewed their proposed testbed, we noticed that they do not have a clear document model. This makes it difficult to reuse their proposed solution to other document formats. Additionally, their solution only considers context obtained from the direct environment of the document. We believe that it is necessary to provide support for contextual information from users and the physical environment that is not directly linked to the document. Hereby, documents are aware of even more information on which they can perform appropriate adaptations.

We join their vision of “*a document can be characterized as a web of components*”. By using the RSL metamodel to describe our document model, we shall attain the same notion of the “*web of components*”.

3.1.7 Multimodal Documents - 2006

Augusto Celentano and Ombretta Gaggi noticed the growth of information service providers. According to them, this information could be used to adapt user interfaces to the current context. The goal of their research [15] was to develop a model and adaptation architecture for context-aware multimodal documents.

Their developed architecture is presented in Figure 6.13. The system enables context awareness by relying on a rule-based system. The rule-based system uses external contextual information in order to select the appropriate media elements. The media elements are annotated with properties that are used by the rule-based system. When all the media elements have been chosen, the system collects them in the desired virtual document. A virtual document acts as a template to produce the *real* digital document. Afterwards, the instantiated document is presented to the user.

By using this technique, we can obtain different representations and interaction possibilities for a particular document while still conveying the same information.

To describe their context-aware documents, they made use of SMIL²⁴, a standard language for describing synchronised multimedia documents.

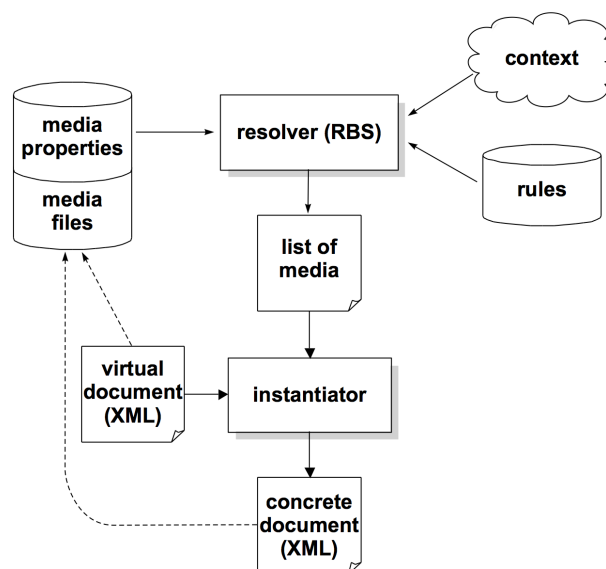


Figure 3.7: The architecture of a context-aware document adaptation system [15]

²⁴<http://www.w3.org/TR/smil> last accessed on 03/02/2014

It is interesting to note that their adaptation approach is based on two viewpoints. On the one hand, the adaptation can happen on a high level, modifying the logical structure of the multimedia documents, e.g. a video file can be replaced by a series of images as long as they convey the same meaning. On the other hand, adaptation can be performed on the selection of media that has the largest cognitive impact on the user. It is interesting to think about the two approaches and how we can find the right balance. Unfortunately, the authors of the paper do not describe how the contextual information is derived from this external source.

The authors of the paper also make an interesting remark that their solution only “enables” the user to create dynamic documents. In order to create “well designed” adaptable documents, we will have to rely on suitable guidelines and methodologies.

Critical Analysis: This solution operates on the basis of a virtual document. It is fascinating to see the clear presence of a document that acts as a placeholder for the final document. We like to remark that the other reviewed solutions did not take this element into account, or at least not that explicitly.

We believe that a virtual document is a very useful tool to specify the structure and the content of a document during the editing phase. Despite the fact that we are unable to know which content will be displayed in the document, it is still possible to specify where content should be displayed when it is actually chosen to be presented.

As a final remark, we like to note that we agree with their statement: *creating a well designed dynamic documents will be an interesting challenge*. We believe that it will be a challenge for the people that create design guidelines, but also for the development of future document editors that will assist us in creating these documents.

3.1.8 WOAD Dynamic Documents - 2011

When reviewing the Minerva documents, see 3.1.6, we have seen that their solution relies on a network of components from which they form their documents. Federico Cabitza and Lade Gesso also rely on this approach in [13]. Their components are named “didgets”. Hereby, they define the term “dynamic document” as follows:

“A dynamic document is an electronic document that users can easily build by aggregating smaller data modules, called “didgets”,

to mimic their paper-based templates and provide them with proactive behaviors in support of daily practice.” [13]

The goal of their research was to build modular and flexible Electronic Patient Records (hereafter EPR), by making use of the dynamic document metaphor. First, they have compared the traditional Paper Patient Records to the electronic ones. They have concluded that with EPRs they lose some advantages. The first flaw they encounter is the lack of flexibility when customising and modifying the templates of official paper-based forms. Secondly, the flexible workflow, that they once had with paper, has now been restricted to a predefined flow.

In order to solve these problems, they have developed a new system named ProDoc. ProDoc provides a set of persistent documents and forms without imposing a certain workflow. The key to developing ProDoc is the concept of the “*Web of Active Documents*” (hereafter WOAD).

WOAD is an interconnected document system that consists of documents and forms that practitioners consult on a regular basis. The building blocks for WOAD are the dynamic documents. A document may in turn be composed out of passive parts and dynamic parts. The passive parts form a container for the content which has a specific structure. The dynamic part empowers the passive part with context-aware behavior by adding executable code.

In turn, the dynamic and passive parts are divided into the following elements: *datoms*, *didgets*, *templates* and *mechanisms*. The smallest atom from which a dynamic document can exist is called a *datom*. A *datom* represents some content of interest that can be used throughout different documents. The representation of these *datoms* are handled by *didgets*. *Didgets* can be reused throughout different document-templates. By using only these three components it would not be possible to make the document dynamic. Indeed, *mechanisms* are required to support the dynamic parts of the document. *Mechanisms* are a collection of *if-then* statements that apply to *datoms*, *didgets* and their corresponding content. These *mechanisms* make it possible for the content of the document to become dynamic and proactive. WOAD’s architecture is further described in [13].

Critical Analysis: As in the case of the Minerva documents, this solution also prefers to consider documents as the composition of components in a collection. It seems that this approach enhances the reuse of content and facilitates the development of dynamic content.

We also prefer their simple classification of components and the notion of *mech-*

anisms to enable the dynamic parts of the document. By using simple *if-then* statements it is possible to specify how the document should adapt. On the other hand, we were missing the support for data that changes over time such as sensor data, web service data, etc. We also believe that their approach can easily be extended beyond the use of EPR's.

Despite their well thought out approach, we were asking ourselves whether it is possible for regular user to design such a document. We would have liked to see an evaluation of how easy users can use these *if-then* statements to specify the dynamic parts of a document. Still, overall we must conclude that their approach looks very promising.

3.1.9 Conclusion

In Section 3.1, we analysed a set of research projects that present the state of the art in dynamic documents. Despite the proposed definitions for the term “*dynamic documents*”, none of them were seen as “*generally accepted*”. This has resulted in different schools of thought that are hard to compare. In general, we could categorise digital documents as follows:

- Digital documents that are context-aware;
- Digital documents that present data that can change over time;
- Digital documents that mix the two aforementioned approaches to modify the content and/or structure.

Since it is primarily dynamic content that make a document dynamic, we can argue that we can only find a correct definition for dynamic documents if and only if we are able to define the term “*dynamic content*”.

Furthermore, we would like to remark that the idea of dividing a document in a set of subcomponents was often used by the researchers. We believe that this technique can be very helpful in facilitating the reuse of content and the support for dynamic content. By composing a document as a set of subcomponents, it is possible to manipulate the individual elements of the document. Therefore, we state that this approach is the most preferable to provide support for dynamic content.

As a final remark in our analysis, we would like to emphasise that none of the examined solutions takes the creation process of dynamic documents into consideration. All of the discussed solutions seem very promising. However, if their

solution is not suited for regular end-users, they cannot be widely adopted. The industry has also noticed the need for documents that are empowered with dynamic content. Microsoft for example has recently announced that their Office solution will provide support for dynamic content by enabling the integration of live-data²⁵. Since Office is targeted towards regular end users, they will also be forced to take the creation process of dynamic content into account.

From our analysis we can summarise the list of problems that prevent the progression of dynamic content in digital documents:

- There exists no general accepted definition for the term “*dynamic content*”;
- There exists no conceptual model of dynamic content;
- The creation process of dynamic content has received little or no attention in previous research.

We believe that by solving these problems, we are able to stimulate the integration of dynamic content and the creation of dynamic documents.

3.2 Review of Document Formats

In this section, we will review a set of existing document formats with regard to their support for dynamic content. The review is based on the existing work of Ahmed A. O. Tayeh [64]. Since he already analysed the formats, it is not our intention to repeat his research. We perform an additional analysis with regard to the support for dynamic content.

Note that we have excluded the following document formats from the original review in [64]: Scribe, TNT, SGML, ODA and DIA Formats. The reason for this choice is that they do not support any form of dynamic content.

3.2.1 L^AT_EX

L^AT_EX is a generalised set of macros built on top of T_EX. T_EX is both a program that performs typesetting and a format that consists of a set of macros. The content and the layout of L^AT_EX documents are specified using the L^AT_EX’s markup language.

²⁵<http://www.pcworld.com/article/2043856/how-microsoft-is-using-tolerance9999-emergencystretch3em-hfuzz.5-p@-vfuzz-hfuzzlive-data-to-redefine-the-office-document.html> last accessed on 09/05/2014

Afterwards, it relies on $\text{T}_{\text{E}}\text{X}$ to format the output. This is handled by a $\text{T}_{\text{E}}\text{X}$ engine.

$\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ is built around the principle of separating content from the visual representation. This allows users to focus on both content and presentation in isolation. Because the content of a $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ -document can only contain plain text, it has to rely on packages to provide additional functionalities. The following two packages are examples that add a dynamic behaviour to the document's output:

- **Ocgtools**²⁶: This package seeks to create a dynamic behaviour by allowing multiple layers of content. These layers enable users to mimic dynamic behaviour by toggling the visibility of each layer. By doing so, it creates the feeling that the document is able to change autonomously.
- **Beamer**²⁷: This package also makes use of overlays to mimic a dynamic behaviour. By using the Beamer's overlay command, we are able to specify how content much change over time. In reality, beamer only merges two consecutive slides together in order to achieve the impression of changing content.

Other examples can be found on the Internet²⁸. These examples are created by using $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, ConTeXt²⁹ and other programs like MetaPost³⁰. However, we note that the dynamic behaviour of the documents can only be viewed using Adobe Reader.

Critical Analysis We have learned that $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ documents rely on external packages to support extra functionalities such as dynamic content. A $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ package is a file or a collection of files that contain a number of $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ commands and programming in order to add styling features to the documents. As a result, we can conclude that the creation of a package and subsequently creating dynamic content is not tailored towards the skills of most users.

From this review we can state that the document format provides support for dynamic content by means of external packages. However, the creation of such packages is a complex task that is not suited for regular users.

²⁶<http://www.ctan.org/pkg/ocgtools> last accessed on 04/02/2014

²⁷<https://bitbucket.org/rivanvx/beamer/wiki/Home> last accessed on 04/02/2014

²⁸<http://www.tug.org/texshowcase/dynamics> last accessed on 04/02/2014

²⁹ConTeXt: software for typesetting high-quality documents, written using a macro language <http://wiki.contextgarden.net> last accessed on 04/02/2014

³⁰MetaPost: a picture drawing language that outputs PostScripts files. It is used to produce figures for documents that can be embedded in $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ <https://www.tug.org/metapost.html> last accessed on 04/02/2014

3.2.2 HyperText Markup Language

HyperText Markup Language (hereafter HTML) is a markup language for publishing hypertext on the World Wide Web. It is a format based on SGML³¹. HTML documents rely on specific tags to structure text into heading, paragraphs, lists, etc. A browser can process an HTML document to present its content. We present a list of document formats that rely on the HTML format.

XHTML

The Extensible HyperText Markup Language (hereafter XHTML) is a document type that reproduces, subsets and extends HTML. Rather than relying on SGML, XHTML is based on XML (a strict subset of SGML). As a result the syntax of XHTML documents are more restricted than regular HTML documents. Additionally, this makes them easier to use by XML-based user agents.

XHTML has two methods for specifying dynamic content. The first method consists of adding specific tags. This was researched by Serge Abiteboul, Omar Benjelloun and Tova Milo [1]. These tags will instruct the interpreter to include/replace sections from other XML documents in the current XHTML document. The second method is to rely on JavaScript to add a dynamic behaviour. By using JavaScript, we are able to modify the DOM-tree of the document in the browser. Note that manipulations to the DOM-tree will only be visible in the browser. Afterwards, the document will remain unchanged.

HTML4

HTML4 has extended HTML with functionalities such as: multimedia options, scripting languages, style sheets, better printing facilities and better access for disabled people.

Developers can make use of client-side scripting languages such as JavaScript and server-side scripting languages such as: PHP, ASP.Net, Java, Coldfusion, Perl, Ruby, Python, etc. to add dynamic behaviours to the presentation of their document.

Additionally to scripting languages, users are also able to embed objects such as Adobe Flash SWF files in their document. This is done by using the OBJECT tag. These objects can be used to present dynamic content in the document. However,

³¹<http://www.w3.org/MarkUp/SGML> last accessed on 09/05/2014

these objects can bring along some disadvantages. For example, Adobe Flash Objects rely on a plugin mechanism. As a result, the browser must rely on the Adobe Flash plugin to present Adobe Flash Objects.

HTML5

The goal of HTML5 is to become an application platform for the web by making use of HTML, CSS and JavaScript. In contrast with HTML4, HTML5 is designed to avoid external plugins for integrating content.

HTML5 also comes along with a set of new tag elements such as `<canvas>` (used to draw graphics using scripting languages such as JavaScript), `<video>`, `<audio>`, etc. Before HTML5, users had to rely on either Flash or JavaScript hacks to integrate complex forms of content.

HTML5 is already a big leap forward when compared to HTML4. However, it is still not possible to specify a range of dynamic content by using the HTML5 specifications. Most of the time, developers still rely on JavaScript to add a dynamic behaviour.

Critical Analysis We have seen a set of markup languages for specifying content for the World Wide Web. The specifications of these languages allow us to create documents with a range of content.

However, dynamic behaviour is mainly achieved by using scripting languages such as JavaScript. We believe that the combination of HTML, JavaScript and CSS can become helpful in isolating the different phases of document editing such as determining the content and specifying the dynamic behaviour of the content and its structure. Nevertheless, by relying on scripting languages, the level of difficulty to specify dynamic content is highly increased.

3.2.3 Portable Document Format (PDF)

Portable Document Format (hereafter PDF) is a file format created by Adobe Systems with the intention to be easily exchangeable. Since 2008, the format was released as an open standard³². PDF documents encapsulate the description of a *fixed layout* document. PDF documents can contain one or more of the following content types: text, fonts, images and 2D vector graphics.

³²http://www.adobe.com/devnet/pdf/pdf_reference.html last accessed on 25/04/2014

Notwithstanding the open standard, Adobe has also added support for some other technologies such as Adobe XML Forms Architecture³³ and Adobe JavaScript³⁴. These tools enable PDF documents that contain JavaScript to add dynamic behaviour. However, by introducing JavaScript as a requirement for developing these documents, they become no longer suited for regular users.

The tool suite³⁵ for developing these documents consists of: a JavaScript editor, a console and a debugger. The JavaScript extension of Adobe has support for multimedia, web services, improved printing control, controlling layers, 3D support and more³⁶.

We would like to point out that the presentation of PDF documents highly depends on the PDF-viewer in which they are viewed. Adobe Reader³⁷ is one of the PDF-viewers that has support for both 3D objects as well as the ability to execute JavaScript. Preview³⁸ on the other hand, does not provide support for such content.

Adobe also created a software package named Adobe LiveCycle Enterprise³⁹. This application is an enterprise document and form platform with the following key capabilities⁴⁰:

- Fill in, sign, and save PDF Forms;
- Analyse form data with 2D barcodes that are updated dynamically as data is entered;
- Allow digital signatures as an authentication tool for PDF documents and forms;

³³https://partners.adobe.com/public/developer/en/xml/xfa_spec_3_3.pdf last accessed on 25/04/2014

³⁴<http://www.adobe.com/devnet/acrobat/javascript.html> last accessed on 25/04/2014

³⁵http://livedocs.adobe.com/acrobat_sdk/10/Acrobat10_HTMLHelp/wwhelp/wwhimpl/common/html/wwhelp.htm?context=Acrobat10_SDK_HTMLHelp&file=JS_Dev_Overview.71.1.html last accessed on 25/04/2014

³⁶http://livedocs.adobe.com/acrobat_sdk/10/Acrobat10_HTMLHelp/wwhelp/wwhimpl/common/html/wwhelp.htm?context=Acrobat10_SDK_HTMLHelp&file=JS_Dev_Overview.71.1.html last accessed on 25/04/2014

³⁷<https://get.adobe.com/nl/reader> last accessed on 25/04/2015

³⁸<http://www.apple.com/osx/whats-new/features.html> last accessed on 25/04/2014

³⁹http://www.adobe.com/be_en/products/livecycle.html last accessed on 25/04/2014

⁴⁰http://www.adobe.com/go/lc_readerextensions last accessed on 25/04/2014

- Enable offline interaction and submission of forms on reconnection;
- Exchange data with back-end systems by making use of web services.

Despite the functionalities of the aforementioned software, the lightweight user interface of Adobe LiveCycle does not enable regular users to create PDF documents with dynamic content in a user friendly manner.

Critical Analysis From the original specifications of the PDF format, we can derive that there is no support for integrating dynamic content. However, Adobe has empowered their PDF documents with the possibility to include XML Forms and JavaScript.

We can compare this approach with the one of HTML. Both formats rely on an external scripting language to adapt their content. However, by imposing the requirement to write scripts in JavaScript, we introduce a new form of complexity. Despite the efforts made by Adobe with the LiveCycle application, we found it too complex for regular users to use.

3.2.4 Extensive Markup Language

Extensive Markup Language (hereafter XML) is a markup language. XML documents are created by a set of rules which are developed to be easily readable for both human beings and machines. Since they are simple, general and easily usable, they are used throughout many domains.

Unfortunately, XML on its own has no native mechanism to integrate dynamic content. However, as discussed in Section 3.2.2, attempts have been made for creating Active XML documents. Active XML documents are XML documents that contain embedded calls to web services. The result of these web service calls are then used to replace content in the document. A web service call embedded in an XML document is presented in Figure 3.8.

```

...
<exhibits>
  <exhibit>
    <name>Naive Painting in Ancient Greece</name>
    <location>Le Louvre</location>
    <from>25-Apr-2003</from>
    <to>25-May-2003</to>
  </exhibit>
  <axml:call service="getExhibits@Yahoo.com">
    <city>Paris</city>
  </axml:call>
</exhibits>
...

```

Figure 3.8: The result of invoking `getEvents@TimeOut.com` [1]

Critical Analysis We have seen that XML documents on their own do not have default support for integrating dynamic content. However, researchers have provided a solution where it is possible to integrate web service calls into the document. This approach enables the document to change when it is processed by a particular XML agent.

We believe that the integration of web service calls is a step in specifying other types of dynamic content. Nevertheless, we would have liked to have seen some additional features such as adaptation of content with regard to the context.

3.2.5 OpenDocument

OpenDocument Format (hereafter ODF) is an XML-based file format. The format was created with the intention to be an open XML-based file format for specifying office documents such as: spreadsheets, charts, presentations, word processing.

Since the OpenDocument format relies on XML, it does not have a native mechanism to support dynamic content. However, there exist document editors such as OpenOffice that provide support for adding extra functionalities. An example of such an approach is the one taken by Carles Pina Estany⁴¹. He has extended an ODF document by using the Python programming language to create real time slides. His current implementation provides support for data coming from MySQL and Python code. The data embedded in the presentation is automatically updated every X seconds. The following figure presents how these different components work together.

⁴¹<http://goo.gl/wYlzcw> last accessed on 09/05/2014

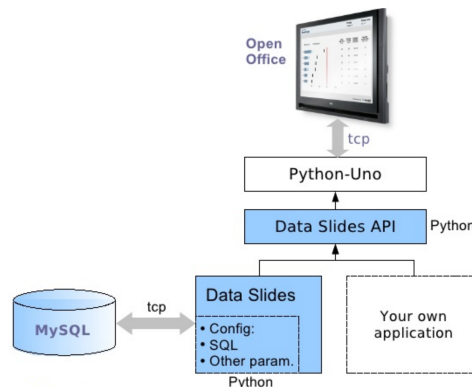


Figure 3.9: Architecture: OpenOffice combined with Python-Uno and MySQL to create real-time slides

Critical Analysis As with the XML format, we can remark that ODF also does not support the specification of dynamic content. We can also conclude that, if someone wants to extend the format with extra functionalities, they must create an extension for the editor. Even more, we have seen that the integration of extra functionalities requires expertise in a programming language. As a result, this approach excludes regular people that do not own such a skill.

3.2.6 DocBook

DocBook is a general purpose XML schema intended to create technical documentation. The specifications of DocBook allow us to independently create documents in a variety of forms in a presentation-neutral way. This means that one DocBook document can be converted to multiple output formats such as HTML, XHTML, EPUB, PDF, etc.

Since the DocBook schema is based on the XML language, it does not obtain a native mechanism to specify dynamic content. However, there exists a possibility to change the content of a DocBook at the server-side. By making use of the `<xi:include>` element, it is possible to include a piece of content at a certain location in the document. Common practice is to do this at the server-side with a specific script.

Critical Analysis Unlike the other formats, this document format is able to adapt itself towards multiple output formats. We can see this feature as a dynamic behaviour of the document format.

Despite the possibility of using the `include` tag, it is not possible to change the content of the document on the fly. Since the document format uses a schema based on XML, it is still not possible to specify dynamic content.

3.2.7 Office Open XML

Office Open XML (hereinafter OOXML) is an XML-based file format capable of representing word-processing documents, presentations and spreadsheets. For each type of document there is a specific markup language such as: WordprocessingML, PresentationML and SpreadsheetML⁴².

Since OOXML is based on XML, it also lacks the capability of supporting dynamic content. The only possibility to create dynamic content is to develop a C# script. With a script we are able to fetch information from a remote resource and update the document on the fly.

Critical Analysis Again, we notice that this format relies on XML. Therefore, the document format has no default support for specifying dynamic content. However, we have seen that it is possible to make the document dynamic by making use of a programming language. Again, we can conclude that this approach is not applicable to regular users with no programming experience.

3.2.8 Electronic Publication (EPUB)

EPUB is the abbreviation for **E**lectronic **P**ublication. EPUB is an open e-book format that was developed by the International Digital Publishing Forum. The main goal of the EPUB format was to support reflowable content. Reflowable content is content that adapts itself to the device on which it is read. However, most of the reading devices do not use this advantage or at least not to its full potential. The majority of e-readers only adapt the font size of the letters.

EPUB documents are composed of different files. They are a collection of: XML, HTML, CSS and SVG files. As a result, EPUB documents have the same expressive power to specify dynamic content as the files from which they are composed of.

The latest version of EPUB is EPUB3. One of the most advanced features of

⁴²http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf last accessed on 09/05/2014

EPUB3 is its ability to provide interactivity and scripting. These functionalities are achieved by using the capabilities of HTML5. Another primary design consideration of the document format was to support a dynamic adaptive layout. Currently the layout adaptation is achieved by using EPUB style sheets, which are a combination of CSS2.1 and some features of CSS3. More information on the EPUB3 specifications can be found on the website of the International Digital Publishing Form⁴³.

Despite the promising characteristics of the EPUB standard, the format is not yet widely adopted. As a result, few systems support this document format. A format that currently provides the best support towards this standard is the iBook format developed by Apple, see Section 3.2.9.

Critical Analysis We have been introduced to a new standard for e-books. This standard makes use of HTML5, XML, CSS and JavaScript to specify the content. As a result, this standard is also limited to the restrictions of HTML5, XML, CSS and JavaScript.

Again, we can conclude that the document format does not provide default support for dynamic content. We will still require JavaScript in combination with the style sheets to add dynamic behaviour to the document.

3.2.9 iBook

The iBook format is a format that is exclusively used by Apple in their iBook application. Actually, the iBook format is mostly EPUB because it uses the EPUB file format. Therefore it shares the same characteristic concerning the support of dynamic content. Additionally, the iBook format, and especially the new iBook 2.0 format, adds CSS extensions that do not conform to the EPUB3 standard. Consequently, it is possible that these documents will not be supported by other EPUB reading systems than iBooks. The iBooks Author application⁴⁴ is currently the only application that allows users to create iBook 2.0 documents. The application makes use of widgets to enable authors to create dynamic content, see Figure 3.10. The available widgets of iBooks Author are the following: an image gallery, a media element, an interactive review object, a Keynote presentation, an interactive image, a 3D object, a scrolling sidebar, a pop-over or an HTML Widget. These interactive widgets are provided by the combined power of HTML5 and JavaScript which allow us to add powerful functionalities to the document.

⁴³<http://idpf.org/epub> last accessed on 05/02/2014

⁴⁴<http://www.apple.com/ibooks-author/> last accessed on 05/02/2014

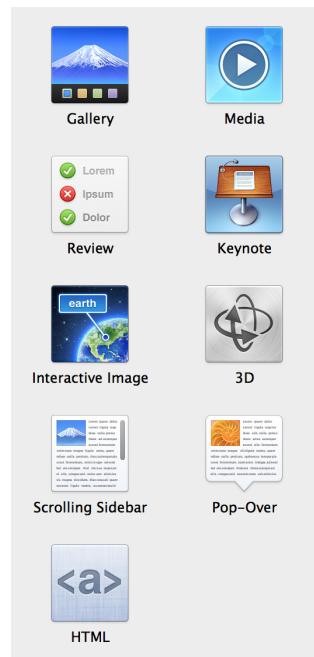


Figure 3.10: iBooks Author: Available Widgets

Critical Analysis The iBook format is built on top of the EPUB standard. Therefore, it is possible to use the functionalities provided by HTML5, CSS, XML and JavaScript. It is interesting to observe that iBooks Author, the editor for iBook documents, has explicit support for creating dynamic content. By proposing a set of available widgets, the user is able to create a range of dynamic content. We believe that this feature is a major advantage compared to the other solutions. Document editors such as Open Office have to rely on some kind of hack for integrating dynamic content in a document, making it a very difficult task for regular users to create dynamic content.

We concur with the approach of iBooks Author. It may be that the document format relies on other technologies such as JavaScript to enable dynamic content. However, since these techniques are not suited for regular end users, the document editor can offer assistance in the process of approaching these technologies.

3.2.10 Conclusion

In the previous section we have analysed a set of document formats, Table 3.1 summarises our findings. Note that the third column indicates the need for other technologies to enable dynamic content. From our analysis we can conclude that there exists no document format that provides full support (denoted by a ✓) for

the integration of dynamic content. In fact, the majority of the document formats do not have a way to specify dynamic content (denoted by a **X**). Indeed, we have noticed that most of these document formats rely on other technologies (mainly a programming language), to add support for dynamic content. As a result, the difference between a document author and a programmer has become blurred.

Document Format	Default Support	Other Technology
LaTeX	X	✓
XHTML	X	✓
HTML 4	X	✓
HTML5	(✓)	✓
PDF	X	✓
XML	(✓)	✓
OpenDocument	X	✓
DocBook	X	✓
OOXML	X	✓
EPUB	(✓)	✓
iBook	(✓)	✓

Table 3.1: Summary of analysed documents formats

We can conclude that the creation of documents with dynamic content implies a certain form of complexity that is not suited towards regular users. Fortunately, we are not alone in noticing this problem. iBooks Author, the editor for the iBooks format, facilitates the process of creating dynamic content by means of widgets. Widgets are off the shelf components that provide a certain dynamic behaviour. We believe that their main goal is to facilitate the transition to complex technologies such as JavaScript, style sheets, etc. for regular end-users.

From this analysis we have learned the following things:

- The majority of document formats do not support dynamic content by default;
- Adding dynamic behaviour to documents is mainly accomplished by using other technologies such as scripting languages;
- The line between document authors and programmers is becoming blurred;
- Document editors must evolve in order to leverage the complexity introduced by the integration of dynamic content.

We believe that it will take some time before document formats will adopt the integration of dynamic content. However, due to our analysis we came to the conclusion that document editors will play an important role for integrating dynamic content in digital documents. We think that the majority of document editors will have to reconsider their approach in order to comply to the needs for integrating dynamic content. As a result, this thesis will focus on the creation of a document editor that should be tailored to regular end-users for including dynamic content in digital documents. The following two chapters will discuss the creation our conceptual model of dynamic content and de development of our document editor.

Chapter 4

Dynamic Document Scenarios

We start this chapter by presenting several scenarios where digital documents are empowered with dynamic content. Afterwards, we categorise these scenarios with respect to their dynamic behaviour. This chapter will also discuss the development of three scenarios along with an introduction to the Resource-Selector-Link metamodel.

4.1 Scenarios

In order to compose a definition that captures the meaning, the use, the function and the essence of the term “*dynamic content*”, we judge it necessary to think about possible scenarios where digital documents can profit from the notion of dynamic content. The following list presents ten scenarios, each with their own insight on dynamic content:

Active Storytelling

Before writing, storytelling was performed orally as means of entertainment, education, etc. These storytellings could change on the fly depending on the speaker. Since books are primarily static, they lose some of these advantages. Therefore, we believe books can benefit from dynamic content. Consider a book where the storyline changes according to the current context (temperature, user, environment, etc.). Different story lines could be present in one book.

The details of this scenario are described in the Appendix, see Section [A.1](#).

Dynamic Restaurant Menu

Most prices of ingredients change according to the season. Additionally, when the weather is nice outside, more guests will want to eat outdoors.

We believe that restaurant owners can benefit from menus that dynamically change. Consider a menu where the prices change according to the current season. This way, restaurant owners will never lose money when ingredients get more expensive. On the other hand, guests will pay a fair price for their product. Additionally, the price could also adapt according to the current climate and whether the guests are eating outside or indoors.

The details of this scenario are described in the Appendix, see Section [A.2](#).

Adaptive Tour Guide

Tour guides have the property to change frequently. Travel agencies often change their prices according to the current season, amount of bookings, weather, etc. However, these changes are not visible in traditional tour guides. We believe that by empowering tour guides with dynamic content these documents could become more useful. Tour guides could for example be autonomously linked to an external source in order to update prices, recommendation of destinations, etc.

The details of this scenario are described in the Appendix, see Section [A.3](#).

Smart Content

Researchers often distribute documents to many people. However, not all parts of the document are relevant to all of the readers. We believe that documents could benefit from knowing who is reading them and where. We think that dynamic content could be used to intelligently adapt parts of the document in order to meet the requirements of the reader and the author.

The details of this scenario are described in the Appendix, see Section [A.4](#).

Finger Reader

When a child starts reading they use their fingers to “*follow*” the sentences. In most cases when a reading device with touch input is touched, an action is performed. Consider that we have a joystick at hand. This joystick could take over the task of the finger and show changes in the document as the joystick “*moves*” over the content of the document.

The details of this scenario are described in the Appendix, see Section [A.5](#).

Fluid Reading

Documents are read on different locations. In some cases it may be preferable that the content of the document is audio instead of text. Consider the case that you are in a car and you were reading a document. Dynamic parts

of the document could then adapt themselves to create a better cognitive impact by providing an alternative presentation of the content such as audio.

The details of this scenario are described in the Appendix, see Section [A.6](#).

Fluid Font

When we are working with documents there is always a certain distance between us and the document. In some cases a document can be far away making it harder to read it. On the other hand, we can look very close to a document in order to see more detail. We believe that a reader could benefit from a document that adapts itself to the distance between the user and the document. Consider a document where the font of the document increases as the reader steps back or a document that autonomously locks itself when the reader leaves the room.

The details of this scenario are described in the Appendix, see Section [A.7](#).

Interactive Vocabulary

Learning a new language can become interactive when using dynamic content. Instead of having a static document, text could change according to user input. Consider a vocabulary where we have two lists, a list with words in a foreign language and a list of words in our mother tongue. By playing with the light intensity or the movements of the reading device, we could change the content of the document.

The details of this scenario are described in the Appendix, see Section [A.8](#).

Live Catalogue

Most of the catalogues present a list of items. However, changes to these physical items are not reflected in the catalogues. As a result, catalogues have to be manually changed and reprinted. This can be a tedious process. We believe that catalogues can benefit from the notion of dynamic content. Instead of replacing the full catalogue, the document could use dynamic content in order to change itself to the latest updates.

The details of this scenario are described in the Appendix, see Section [A.9](#).

Live Travel Guide

Paper travel guides contain a lot of information about a specific area. Most of the time, people have to browse through these documents in order to find the appropriate information. We believe that travel guides can benefit from dynamic content by showing the appropriate information at the right time.

Consider a walk through the city, the content that is displayed could depend on the current location. Another example could be the adaption of the point of interests by using the current time, date, temperature, etc.

The details of this scenario are described in the Appendix, see Section [A.10](#).

Since these scenarios are quite diverse, we have chosen to develop a categorisation that classifies the scenarios according to their means of adaption.

4.1.1 Categorisation of Scenarios

Due to the diversity of our scenarios, we deem it helpful to have a classification system that allows researchers to group and compare their developed scenarios or prototypes according to a number of relevant factors.

In essence, dynamic content adapts the document in some kind of way. Therefore, our classification system is orientated towards content adaptation. Since there already exists a general classification system for content adaptation we were able to reuse parts of that.

Mohd Farhan, Jemal Abawajy, et al. propose in [22] a classification for content adaptation system. Their classification consists of the following six components:

- Locality: *Where* to perform the content adaptation (centralised or distributed).
- Strategy: *Who* should perform the content adaptation (underlying system, application, system in combination with the application).
- Mechanism: *What* should be adapted.
- Purpose: *Why* should it be adapted (general purpose or content-type-specific).
- Context: *To what* should it be adapting.
- Method: *How* to adapt (transcoding, content layout-rearrangement, distillation, etc.).

As a basis for our categorisation, we reused the following elements: *Locality*, *Mechanism*, *Context* and *Method*. The reasons for the aforementioned components are explained in the next paragraphs.

For the first component, locality, the reason is quite obvious. For context-aware

documents, adaptation is mainly performed on the client side. The client is responsible for adapting the view of the document. Additionally for some context-aware documents the adaptation may already be performed at the server-side. In this case, the server that delivers the document will be responsible for the adaptation. When we look at the second component, strategy, we see that it emphasises on the role of performing the actual adaptation. Because all of the purposed prototypes are specific applications that are system-independent, we did not take this factor into account. To our understanding, the majority of applications that perform content adaptation will only rely on the underlying system as platform to operate.

Another important factor besides locality is the mechanism of adaptation. Every context-aware document must define which parts of the document should adapt and how this is realised. This component becomes of a great interest during the development phase of the document. It allows developers to anticipate on the dynamic structure of the document. Furthermore, it contains information on how adaptive parts shall influence the internals of the document. When we review the purpose-component of the aforementioned categorisation, we have two choices. Either the adaptation is a general purpose adaptation or it is content-type-specific. Because dynamic content can have many purposes, we could deduce that there exists no general purpose to perform content adaptation. In fact, one could state that it does not exist because there is no general standard for adapting documents. The potential adaptation that could be performed on the content of a document is in a sense so broad, that it is nearly impossible to grasp it with a categorisation component.

The most important classification component is without saying the context itself. Rather than including the context component as a whole, it is further subdivided into four parts, namely: *Target for adaptation*, *Input*, *Environment Conditions* and *Monitoring Entity*. The idea behind subdividing the context into these four parts was inspired by two other papers. The first paper [20] proposes design guidelines for adaptive multimodal mobile input solutions and was written by Bruno Dumas, María Solórzano and Beat Signer. The second paper [66] written by Angela Sasse and Chris Johnson, introduces the notion of plasticity as a new property for interactive systems. From the first paper, we reused the part of their categorisation on context sensitive automotive input adaptation. The second paper served as an insight on the design space of content adaptation. By making this division, we were able to categorise our prototypes with a more specialised classification.

First of all, it is important to identify for whom or what the adaptation is designed for. This is called the target for adaptation. The target could either be

the adaptation towards a user(s), the environment or the physical characteristics of the system. Besides the target of adaptation, we can also classify a content adaptation system by the way the adaptation is initiated on a lower level. The input-component is responsible for describing these low level input channel(s). Of course, besides the low level input we need to have a higher level of context-description which represents the semantics gained through low level inputs. This is where the environmental conditions come in.

Environmental conditions are a description of the resulting fusion of the different low level input channels. In order to make the link between the environmental conditions and the low level input channels, there is a need for a third component, namely, the monitoring entity. The monitoring entity component acts as an adapter to connect the low level input channels to the corresponding environment conditions.

The last component that was re-used is the method component. This component describes the manner in which the adaptation is achieved. Again this component is subdivided into three other components. The three components in which the method component is divided are the following: *Means of Adaptation*, *Output Influence* and *Adaptation Policy*. The how-property of a content adaptation system is quite essential, therefore it deserves an extensive examination.

The means of adaptation allow us to identify the software components which are involved in the adaptation process. As described by the writers of [66], these software components are typically: the system task model, the way of rendering the view and the way the systems provides help during a certain task. In our scenarios the most frequently used software-component tends to be the system rendering. This software component visually changes the structure and/or content of the document. However, in some cases where the focus lies more on making the document satisfy the plasticity property [66], the software component may also be the system-task model.

When we look at the adaptation policy, we see that there are two inference mechanisms, namely, a *rule-based approach* and a *heuristic algorithm*. With a rule-based mechanism, we eliminate the need for user-feedback during the adaptation-process. According to us, this approach eases the development process which makes rapid prototyping possible. As a last subcomponent of the method component, we have the output-influence component. The output-influence component reflects the output of the performed adaptation.

Categorisation

Figure 4.1 presents the categorisation of our prototypes.

Prototype	Where? (locality)	Target for Adaption	Means of adaption	Input	Output Influence	Environment Conditions	Monitoring Entity	Adaption Policy	Adaption Mechanism
Dynamic Tour- guide	Distributed: - Client Side: view - Server Side: tour information	-Environment	System rendering	Sensors: - location - temperature - light - humidity	Provided information tour	-Location: GPS location -Physical Condition: weather (sunny, cloudy, rainy) -Time (spring, summer, autumn, fall, winter)	- GPS location - Weather - Time	Rule Based	- Changing visiting order - Changing places to visit, recommendations
Active Storytelling	Distributed: - Client Side: view - Server Side: story	-Environment -user	System rendering	Sensors: - location - temperature - light - humidity	Change in storyline	-Location: GPS location -Physical Condition: weather (sunny, cloudy, rainy) -Time (spring, summer, autumn, fall, winter)	- GPS location - Weather - Time	Rule Based	- Influence the story by: weather, location,....
Interactive Vocabulary	Centralised: Client side	-User	System rendering	Sensor: - light	Word translation tips	-Physical Condition: light level (bright, medium, dark)	Light level	Rule Based	- Translation or tips are shown according to light intensity
File Access & User Specific Content	Distributed: - Client Side: view - Server Side: User Content	-Environment -user	System rendering	Sensor: -RFID -Location	Change of content	-Location: indoors (offices), outdoors	Location	Rule Based	- Change the content according to the current user and location
Dynamic Restaurant Menu	Centralised: - Server Side: tour information	-Environment	System rendering	Sensors: - location - temperature - light - humidity	Prices recommendations	-Location: indoors (at the window, close to the air- conditioning,...), outdoors (terrace) -Physical condition: weather (sunny, cloudy, rainy) -Time (spring, summer, autumn, fall, winter)	- GPS location - Weather - Time	Rule Based	- Prices are depending on environmental factors - Food is recommended according to the environment and seasons
Fluid Font	Centralised: Client side	-Environment	System-task model	Sensor: - proximity	Font size	-Location: relative location (device-ear, device-face)	Proximity	Rule Based	- Distance between user and screen adjust's font-size
Finger Reader	Centralised: Client side	-User	System rendering	Sensors: - slider	Current emphasised word	(N/A)	Slider location	Rule Based	- Slider determines which word should be highlighted
Fluid Reading	Distributed: - Client Side: representations - Server Side: data	-Environment	System-task model	Sensor: - proximity sensor	Content representations	-Location: relative location (device-ear, device-face)	Proximity	Rule Based	- Depending on the current situation display the content or speak the content
Live Catalog	Centralised: Client side	(N/A)	(N/A)	Store Database	Prices amount of products in stock	(N/A)	Web service	Rule Based	- Update prices and amount based on live information stored in the DB of the store
Live Travel Guide	Centralised: Client side	(N/A)	(N/A)	Web Service Yahoo travel	Update top destinations	(N/A)	Web service	Rule Based	- Update to document so it always previews the top recommendations according to Yahoo Travel

Figure 4.1: Categorisation of dynamic document scenarios

4.2 Prototypes

4.2.1 RSL Metamodel

Preface

The term “*hypermedia*” was coined in 1965 by Ted Nelson [44] along with the term “*hyperlink*”. Still, the origins of hypermedia systems can be traced back even further to the visionary Vannevar Bush. In 1945, Vannevar Bush wrote the essay: “*As We May Think*” [12]. In this paper, Bush describes the problem of the information explosion with regard to scientific research. In his paper, he presents the concept of the “*Memex*”, an abbreviation for **Memory Extender**. The Memex is a hypothetical microfilm based system for personal information storage and retrieval. This machine allows users to store their data (books, records and communication) and provides a mechanism to consult this data with exceeding speed and flexibility. The interesting part of the Memex is the matter of selection. Instead of adopting the mechanisms used by libraries (indexing), the Memex allows selection by association. The reason behind this approach is that the human mind does not operate by indexing but rather by creating associations between data i.e. creating a web of associative *trails*. Of course in our human brain, trails that are not frequently followed are prone to fade away. This problem could be solved by making use of the Memex as a supplement to the user’s memory.

Throughout the years, various types of hypermedia systems (adaptive hypermedia systems [11], physical hypermedia systems [26], etc.) and models (adaptive hypermedia application model [18], Dexter hypertext reference model [27], etc.) have arisen. All these models and hypermedia systems were developed with Bush’s underlying model of information spaces as a collection of resources that are linked with each other. Even the WWW (World Wide Web), the world’s most famous hypermedia system, finds its origins back to the Memex.

However, as pointed out by the authors of “*As We May Link: A General Metamodel for Hypermedia System*” [58] there is a lack of a conceptual model that would encourage the development of a wide range of hypermedia systems. As quoted from their work:

“a study of the hypermedia literature reveals a lack of clear, conceptual models that are general and flexible enough to support the development of a wide range of hypermedia systems and applications.”
[58]

Fortunately, the authors of this paper propose a solution to this problem. Their goal was to produce a general platform, flexible enough for the development of

both future and current hypermedia systems. This platform was developed by the principles of metamodel-driven engineering and extensibility. The core of the proposed metamodel can be divided into three parts, namely: resources, selectors and links resulting in the name “*Resource-Selector-Link (RSL) metamodel*”. Essentially the RSL metamodel is based around the concept of linking resources as envisioned by Vannevar Bush and his concept of the Memex.

The RSL metamodel has also proven itself, as it has been used as basis for a myriad of projects. The iPaper framework [46] is such a project. The iPaper framework allowed the creation of PaperPoint [59] (a paper-based presentation and interactive paper prototyping tool). Furthermore, the RSL metamodel enabled the creation of a cross-media information platform, named, iServer [57]. The iServer platform is able to support different categories of hypermedia systems due to the generality and extensibility of the RSL metamodel.

Three of our scenarios were developed using the distributed iServer⁴⁵. Since the iServer is a cross-media information platform based on the RSL metamodel we will discuss the components of the RSL metamodel in the following sections. In order to describe each component, we relied on the work of Beat Signer and Moira C. Norrie performed in [58].

Link Functionality

In this section, the linking functionality of the RSL metamodel is explained. It is important to note that the metamodel is specified using the semantic of the object-oriented data model OM. The details of OM can be found in [45]. Essentially, OM provides constructs for both the representation of entities as their relationships. Furthermore, OM provides some distinguishing features such as *typing* and *classification*. On the one hand, typing deals with the representation of the entities as objects (i.e attributes and methods) and supporting inheritance. On the other hand, classification will present the semantic roles for the entities.

With OM we are also able to define collections for semantic grouping of entities. This allows us to define constraints among collections. This means that we have the possibility to constrain the collection in terms of the membership type. As the OM model defines a full operation model over objects, collection and relationships, it also provides constructs for their definition. These expressive features allow us to capture the semantics of the application domain by using a simple set of constructs. An important advantage of this model is the direct representation

⁴⁵This iServer implementation is not the original implementation presented in [57]. However, it has been developed with the same conceptual model underneath.

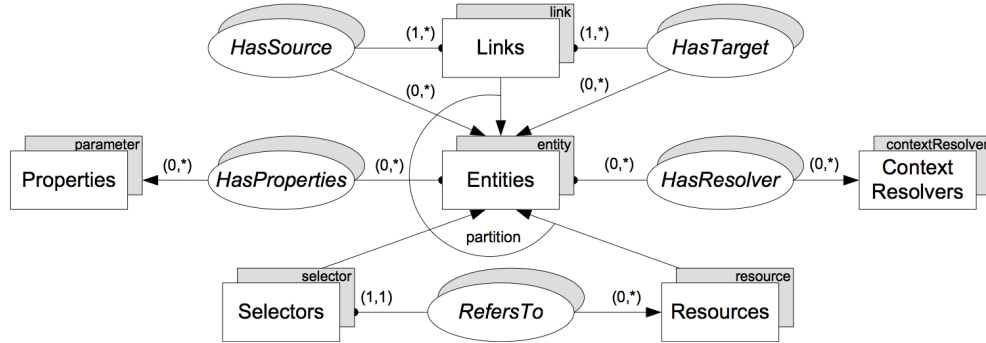


Figure 4.2: Core Link Metamodel [45]

and manipulation of associations which is ideal for link management in hypermedia systems.

The schema of the core link functionality is presented in Figure 4.2. Note that the shaded rectangular shapes represent the classifications i.e denote the collection of objects, whereas the unshaded part of the shape presents the type of the associated shaded part. Thus, resources in the model are represented by objects of the type `resource` which are grouped into the `Resources` collection. In order to present the associations between entities of two collections we make use of shaded oval shapes.

The most general concept of the RSL metamodel is the notion of an `entity`. Each entity can have properties. Properties are assigned to entities in the form of key-value pairs. All the elements used in the RSL metamodel inherit from this type. Furthermore we can distinguish three main types of entities, namely: `resource`, `selector` and `link`. The `resource` subtype is used to represent an entire information unit. Since the resources of an hypermedia system can vary (video, text, audio, etc.), a plugin mechanism is used to provide a specific extension of a resource. It may also occur that we want to create a link to a specific part of a resource. The RSL metamodel provides us with the `selector` subtype to address a particular part of a resource. When we look at the cardinality defined at the source point of the `RefersTo` association, we notice that a selector is only associated with one resource. On the other hand, it is perfectly possible that a resource can have multiple selectors. As with the `entity` type, support for selectors for different resource is handled by a plugin mechanism. Of course we want to create links between entities, this is done by the `link` type. When we look at the model, we notice that a link can have multiple sources and multiple targets and should at least have one source and one target (placeholders can be used

when the source or target are not available at link creation time). This restriction prevents the system of being in an inconsistent state due to dangling links. The division between sources and targets is required to determine the direction of the link. Nevertheless, sources and targets are both subtypes of the `entity` type. This results in a flexibility to create links of which targets or sources can be of any subtype of an entity, even other link instances.

Notice that an entity may have zero or more context resolvers. Context resolvers are used to determine the visibility of the entity. If a context resolver is executed, it will return a boolean value. When all the context resolvers of an entity return positive, then the entity is visible. A benefit of introducing the concept of context-dependent information at the core of our model is that we are able to specify the visibility of resources, selectors and links separately from each other. Especially adaptive hypermedia systems will benefit from this feature. Furthermore, all systems that implement this model will have to provide a context-resolver for handling access rights. This part is presented in the next section.

User Model

The personalisation of links and resources requires the notion of *data ownership*. Furthermore, when sharing entities with others, we need to define access rights. The RSL metamodel supports both personalisation and access rights at the entity level. This enables us to define permissions for both links, resources and selectors. A prerequisite for this support is that the metamodel has an explicit notion of a user. Figure 4.3 presents this user management component in the RSL metamodel.

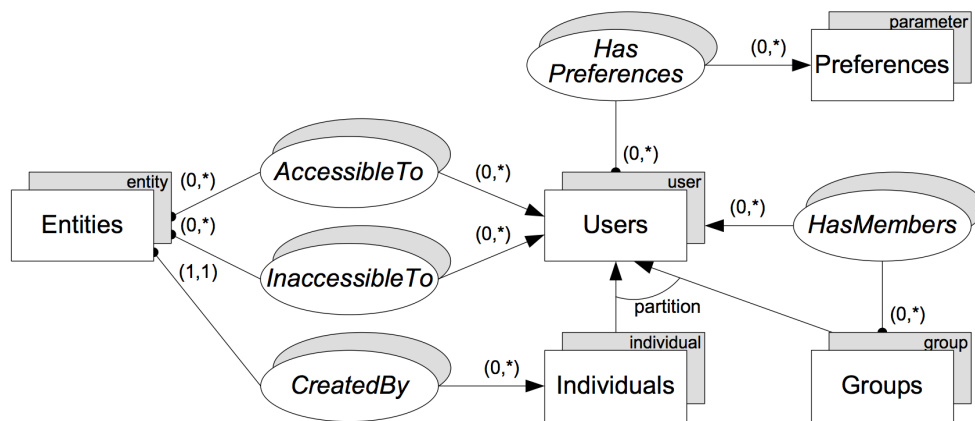


Figure 4.3: User management [45]

A `user` can either be an individual or a group and a group can exist out of users i.e individuals and groups. Each entity must have a creator specified which is always one individual. Furthermore, flexible access rights can be accomplished by two types of associations namely, `AccessibleTo` and `InaccessibleTo`. Access to the specific entities are granted to the individuals and groups which are in the subset of the individuals associated by `AccessibleTo` minus the individuals associated by `InaccessibleTo`. It is important to note that there is a constraint that prioritises individual access rights above the access rights defined for a group. Therefore, it is not by default that when a group retains access rights to an entity, every individual or other group within this group will retain access to this entity. Instead the individual access rights overwrite the group's access rights. In other words, these associations enable complex access rights on entities for both individuals and groups.

Layers

We have already seen that selectors allow us to address a particular part of a resource. But then we can ask ourselves the question “*How do we deal with a resource of which its selectors overlap with each other?*”. The RSL metamodel introduces the notion of “*layers*” to solve this problem. This concept is presented in Figure 4.4.

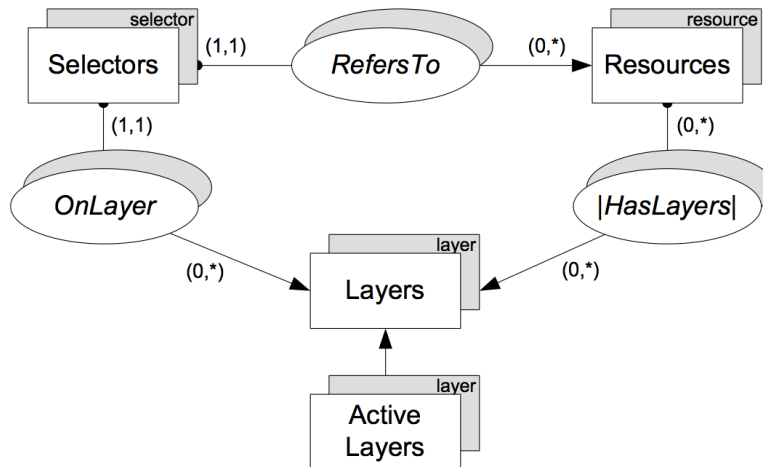


Figure 4.4: Layers [45]

A selector is associated with exactly one layer and overlapping selectors cannot have the same associated layer. When a selector returns multiple links by activating multiple overlapping selectors of the resource, then by definition, the upper-

most layer will be selected. In order to know which layer is the uppermost, the collection of layers must be ordered (notice the '|' at the `|HasLayers|` association). Note that we also have the possibility to activate or deactivate certain layers and that the resource determines to which possible layers his selectors can be associated with. This flexibility combined with the context resolvers allows us to create dynamic selectors and layers that are context-dependent.

Structure

We have already been introduced to the concept of links. In the RSL metamodel, links are treated as first class objects. Additionally, links also allow us to describe the navigational relationship and structural relationships between resources. This implies that both navigation and the structural components are on the same level as the resources. As presented in Figure 4.5, the collection of links is partitioned into navigational links and structural links. Because the structural link is a sub-collection of the regular links, we are able to define a structure on every entity-subtype (e.g. resources, links and selectors).

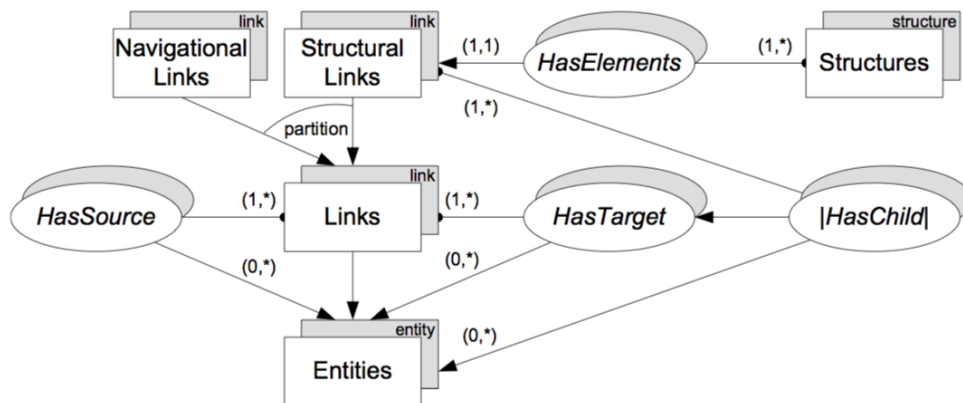


Figure 4.5: Navigational and structural links [45]

Because parts of structures may want to be reused by other structures, the RSL metamodel uses the **Structures** collection that is responsible for handling the structures. A structure in this collection is associated with its structural links through the **HasElement** association. As presented in Figure 4.5, structural links have an ordered **|HasChild|** association. This association is necessary when for example we want to model a book. A book may have different chapters which in turn may have different sections. The chapters and the sections combined together, determine the full structure of the book. Therefore, the **|HasChild|**

association together with the `Structures` collection provide the necessary information on what elements belong to a structure and what their structural relationship is.

By introducing these structural links we are able to specify the structural relationships of different resources i.e *structure over data*. As we have seen in our example about the book this is required to specify the relationships between chapters, sections, paragraphs, etc. The systems that use the RSL metamodel will be responsible for defining domain-specific structures. An example of such a system is the one presented in the masters thesis of Ahmed A. O. Tayeh [64]. In his thesis, he presents the *Fluid cross-Media Document Format* (FCMD) metamodel that is also based on the RSL metamodel. FCMD is a metamodel that presents how the RSL metamodel can be extended to form a general metamodel for all document models. Furthermore, it describes how it deals with the logical structure (i.e organisation of logical objects) and the physical structure (i.e physical representation) of documents.

We have also observed that Tayeh’s master thesis justifies the need for this thesis. Tayeh states that in order for ubiquitous computing to succeed, the default logical objects such as images, text blocks, video, etc. are not enough. Furthermore, the author wrote the following statement:

“The document metaphor has to be extended to apply to dynamic objects derived from the enclosing environment, for example, information obtained from a database or information from sensors (e.g. temperature or accelerometer data)” [64]

Again the contribution of this thesis is noticeable. Our extension over the document model will enable such dynamic objects. We refer to [64] for more information on the FCMD metamodel and its implementation.

Besides putting structures on resources, we can also have *structures over structures*. This allows us to superimpose any structure on top of existing structural components. This is possible because each structural link in a structure defines a substructure that contains the structural links of its source elements and its children (this happens recursively). Eventually, we are also able to put *structures over links*. These structures would for example enable us to create structural links on navigational links. This feature allows us to modify the structure of navigational links, and therefore changing the trail in which resources are visited.

4.2.2 Objectives of the Prototypes

A first and main objective is to understand how dynamic content can be seen conceptually in order to provide a technical implementation. Additionally, we believe that by developing some of our scenarios, we are able to make a better judgement of what the definition of dynamic content should entail.

A second objective is to investigate the current state of the RSL metamodel. Since the developed prototypes rely on the RSL metamodel, we can conclude that it stands up to the requirements of dynamic content. Afterwards, we can study how we must extend the RSL metamodel in order to provide support for a wide range of dynamic content categories.

4.2.3 Infrastructure

The developed applications are targeted for the Android operating system. We have chosen to develop a mobile application for each scenario since most mobile devices are equipped with a range of sensors that can easily be managed by the provided Android SDK. Table 4.1 presents an overview of the prototype's apparatus.

Table 4.1: Tools and Platform

Integrated Development Environment	Android studio v0.4.0
Programming Language	Java
OS Developing Machine	OS X version 10.9
Testing Device	HTC V One
Minimum SDK Version	7
Targeted SDK Version	19

4.2.4 Prototype 1 - Interactive Vocabulary

This prototype consists of three educational parts. The first educational part covers the training phase of a new vocabulary. During this phase children have the possibility to interactively learn new words by hiding and showing the translations according to the child's personal needs.

The vocabulary can be divided into two columns. The first column contains a list of words in the mother tongue of the child. The second column is a list that contains the corresponding translations. This is presented in Figure 4.6. Based on the flux value sensed through the light sensor, more or less information will

be presented. This enables children to learn the new words independently and at their own pace.

English => Dutch	
DOG	HOND
SNAKE	SLANG
RAVEN	RAAF
MOUSE	MUIS
SQUIRREL	EEKHOORN
CAT	KAT
COW	KOE
HEDGEHOG	EGEL
HORSE	PAARD
TIGER	TIJGER

Figure 4.6: Prototype 1: Learning table - *Fully exposed to light*

The child can cover the light sensor to decide how much information should become visible. The idea is based on a technique often used by children to learn a new vocabulary, namely hiding and peeking the translation of a vocabulary. Figure 4.7 and Figure 4.8 shows how the document adapts to the content according the incoming light flux value.

English => Dutch	
RAVEN	_____
SNAKE	_____
HEDGEHOG	_____
MOUSE	_____
TIGER	_____
DOG	_____
COW	_____
HORSE	_____
SQUIRREL	_____
CAT	_____

Figure 4.7: Prototype 1: Learning table - *Fully covered from light*

English => Dutch	
RAVEN	R__F
SNAKE	_L_A_
HEDGEHOG	_G_E_
MOUSE	MU__
TIGER	T_J__
DOG	HO__
COW	K_E
HORSE	_A_D
SQUIRREL	_K_O__
CAT	KA__

Figure 4.8: Prototype 1: Learning table - *Partially covered from light*

The second part is a quiz to test the knowledge of the child. As with the previous part, the light sensor is used to trigger the content adaptation. Only this time, a reverse process is used for showing and hiding the content. The prototype allows a child to fill in the correct translation. When entering the answer, the hands of the child are placed on the sides of the device. Therefore, it becomes impossible to use the same hide and show mechanism as the one described above. In order to solve this problem, the adaptation will be triggered in the reverse way. When the light-sensor is fully exposed to light, the answer stays hidden. When the light sensor is covered up, the answer becomes visible. This is presented in Figure 4.10.

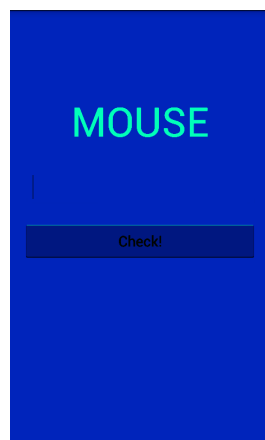


Figure 4.9: Proto-
type 1: Quiz - *Fully
exposed to light*

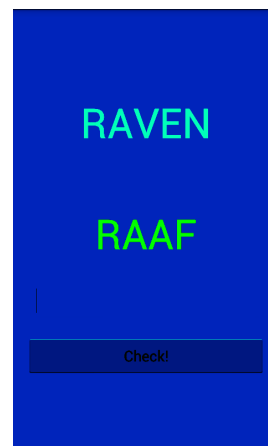


Figure 4.10: Pro-
totype 1: Quiz - *Covered from light*

The final part of the application relies on the movement of the reading device. Nowadays, most smartphones and tablets contain an accelerometer. This component is capable of detecting movements of the device. We thought of a way where we could use the accelerometer to adapt the content of the document.

When launching this exercise, the child is presented with two words. The first word is in the mother tongue of the child and the second word is the translation of the word. However, the letters of the translation are not well ordered. Since the accelerometer of the device is capable to detect motion, we can detect when the device is being shaken. This exercise, see Figure Figure 4.11, will shuffle the order of the letters according to the movements of the device.

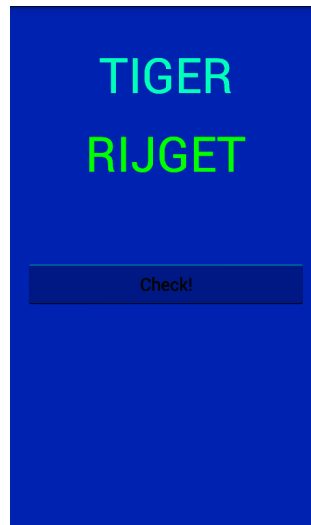


Figure 4.11: Prototype 1: Shake it - *Device Shuffled*

We also added support for tilting the device. When the device is tilted to the left, the letters of the translation are sorted into alphabetical order, see Figure 4.12. On the other hand, when the device is tilted to the right, it sorts the translation into a reversed alphabetical order, see Figure 4.13.

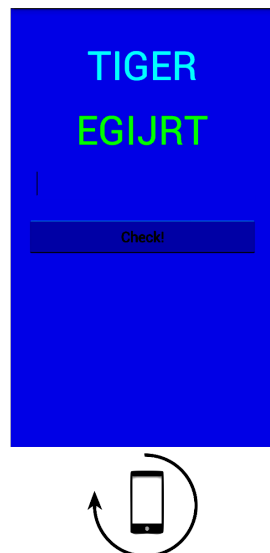


Figure 4.12: Prototype 1: Rotate left

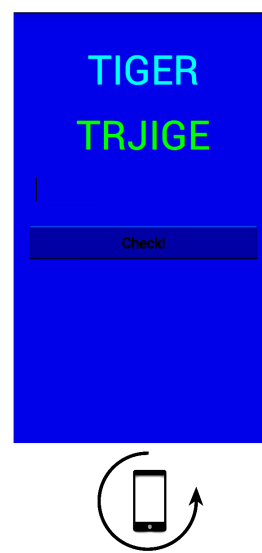


Figure 4.13: Prototype 1: Rotate right

iServer Extension

Our application acquires its content by querying an iServer instance. In order to cope with the requirements of our prototypes, we had to extend the RSL metamodel. This is presented in Figure 4.14. We have introduced the notion of a *Complex Resource* to present a resource that contains a list of other resources. In order to specify the translation of a word, we made use of structural links. Structural links are connected to the corresponding translation of a word.

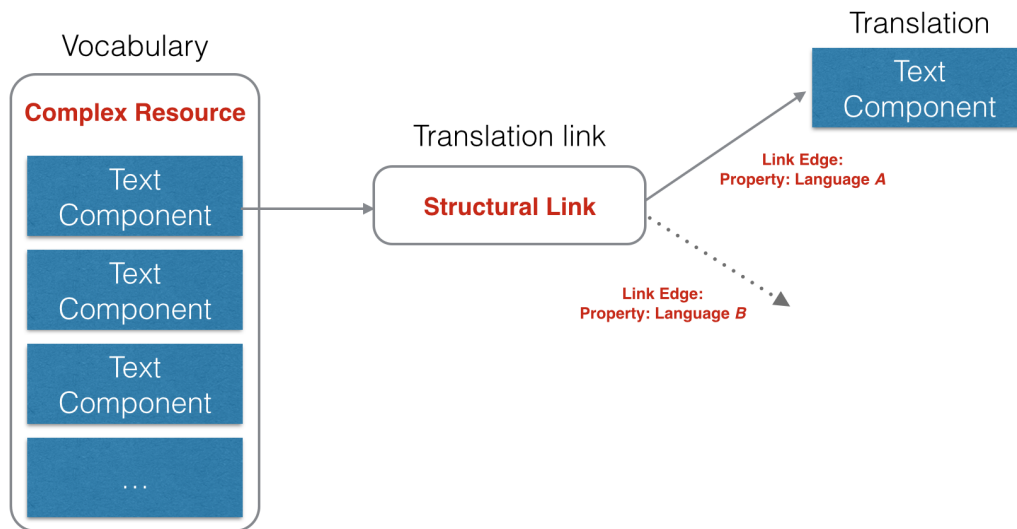


Figure 4.14: Prototype 1: RSL - Content Presentation

4.2.5 Prototype 2 - Adaptive Travel Guide

This prototype is a simplified version of a travel guide catalogue. The catalogue represents historical information about a city combined with up to date weather conditions for the current location.

When starting the application, we present the first page of our catalogue. The first page presents the latest weather conditions of Sydney. This is the dynamic part of the catalogue. Furthermore, the page also presents some static content such as historic information over Sydney. The second page presents similar information but then for the city of Brussels. Figure 4.15 and Figure 4.16 represent the aforementioned pages in respective order. The weather information is retrieved by invoking a web service at a predefined interval.



Figure 4.15: Prototype 2: Informational page over Sydney



Figure 4.16: Prototype 2: Informational page over Brussels

iServer Extension

In order to support content that changes according to an external event, we have introduced the `ObservableResource`-class. An `ObservableResource` implements the `Subject` interface of the `Observer Pattern` [23]. The implementation of the `ObservableResource` is presented in Listing 4.1.

Listing 4.1: Observable resource

```
// a resource capable of being observed and notify the
// observers when changes are made to the resource
public class ObservableResource extends Resource
    implements Subject{
    private Vector<Observer> observers;
    private Object result;
    public Object getResult() {
        return result;
    }
    public void setResult(Object result) {
        this.result = result;
    }
    public ObservableResource() {
        this.observers = new Vector<Observer>();
    }
    @Override
```



```

public void notifyObservers() {
    for (Observer observer : observers)
    {
        observer.update(result);
    }
}
@Override
public void registerObserver(Observer o) {
    observers.add(o);
}
@Override
public void removeObserver(Observer o) {
    observers.remove(o);
}
}

```

By creating such a class we have extended the RSL metamodel with the notion of an ObservableResource. This subtype of resource is able to be observed by others. Another requirement is that the resource is able to update itself. We have implemented this mechanism by making use of a scheduler. A scheduler allows a resource to asynchronously execute a task at a certain interval. In case of a web service, this task can be a web service request.

The following diagram describes how we have extended the RSL metamodel to support the integration of a Web Service. We have introduced the Observable Resource entity. This entity is able to notify another entity when the content must change. Changes can be triggered by a RealTimeInternetComp. These components will change the resource according to updates coming from a web service.

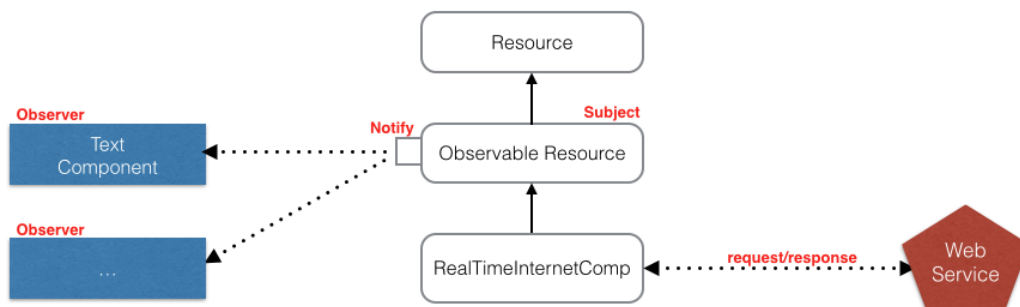


Figure 4.17: Prototype 2: RSL - web service and observable resource

4.2.6 Prototype 3 - Live Travel Guide

This prototype will present a live travel guide. Since the content of the travel guide should change according to the current location of the user device, we need a way of specifying this relation.

In order to accomplish a relation between the text and the location we have to specify two links. One link that specifies which text should become visible when entering the location and another link that specifies which text should show when leaving the location. The results is presented in Figure 4.18 and Figure 4.19. These applications show the same page but the content is changed according to the current location. By following the the correct link, the application knows which text should be presented when entering or exiting a particular location.

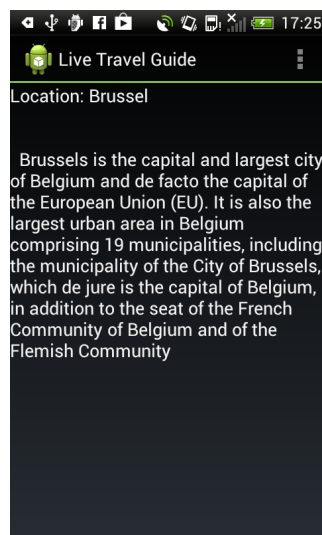


Figure 4.18: Prototype 3: Travel guide information of Brussels

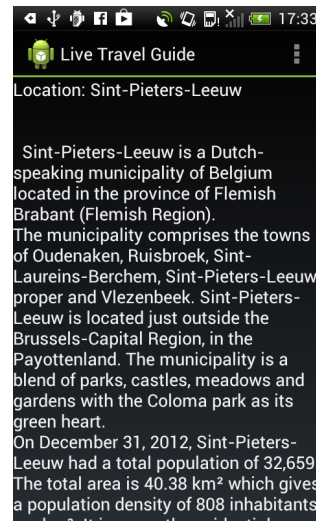


Figure 4.19: Prototype 3: Travel guide information of Sint-Pieters-Leeuw

The application detects the transition of entering and exiting a certain location by monitoring a *geofence*. A geofence is characterised by a longitude, a latitude and a radius. It is a virtual perimeter for a real-world geographic area. Figure 4.20 presents an example of a geofence with a certain location with a particular radius.



Figure 4.20: Geofence with latitude 50.784, longitude: 4.250 and radius 15 meter

The geofences in our prototype are implemented by using the “*Geofencing API’s*”⁴⁶. Note that small geofences only work properly with a radius of 15-20 meters or higher.

iServer Extension

In order to create the links between the geographical area and the content we had to extend the RSL metamodel. We started extending the model by introducing the `AbsoluteLocation` entity. The `AbsoluteLocation` class contains two properties, namely a latitude and a longitude. Afterwards we were able to use structural links to specify how text should change according to absolute locations. How the structural links are connected to the content is visualised in Figure 4.21. Each absolute location is connected to two text components. One text component is displayed when entering this location and another text components is displayed when leaving this location.

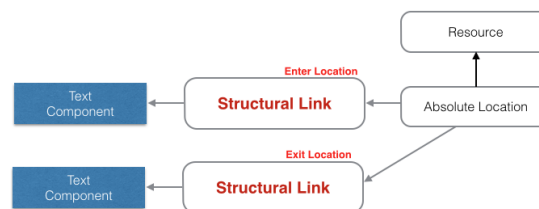


Figure 4.21: Prototype 1: RSL - Content Presentation

⁴⁶Geofencing API's: Allows an Android application to specify geographical boundaries around certain locations with the ability to retrieve notifications when entering and leaving this location. <http://developer.Android.com/google/play-services/location.html>

4.3 Conclusion

In this chapter we have discussed several scenarios. These different scenarios provide food for thought for the opportunities of digital documents with dynamic content.

One of the objectives of developing these prototypes was to investigate whether the RSL metamodel can easily be extended to support the integration of dynamic content. Since all of our prototypes were developed with the RSL metamodel, we can conclude that this model works sufficiently well enough when extended with the appropriate components. Even more by working on top of this thoughtful basis, we were able to easily provide a set of extra features such as an authorisation feature, the ability to specify relations amongst resources, etc.

A second and prime objective of developing these prototypes was to determine the key requirements that enable dynamic content on a conceptual level. We noticed that in order to support dynamic content, we had to perform several steps (these steps were accomplished by extending the RSL metamodel). For example, in the case where we change the text depending on the location: as a first step, we specified a link between a location and a text element; as a second step, we monitored the current location of the device; as a third and final step, we used the observer pattern to notify if the text should change according to the current location.

What we actually did was dividing the process of updating data into a set of sub-tasks. We created a chain of resources and we imposed rules over them. When a resource changes, it notifies the other resources (in some cases data is passed between these resources). This concatenation of elements allows the prototypes to change their data on the fly.

By dividing the support for dynamic behaviour in sets of subcomponents, we embrace a branch of software engineering known as component-based development. In this branch, a reuse-based approach is adopted by implementing loosely coupled components that together provide a certain functionality.

Since the approach of dividing and connecting components seems to be successful, we can abstract it and use it as an extension on the RSL metamodel for supporting dynamic content. This will be further discussed in the next chapter.

Chapter 5

Conceptual Model of Dynamic Content

In this chapter we present a definition for the term “*dynamic content*” and how we came to this definition. Next, we introduce our conceptual model of dynamic content along with some examples that explain the different components of our model.

5.1 A Formal Definition of Dynamic Content

Analysing previous research (see Section 3.1) has revealed that researchers do not have a common ground on what dynamic documents should be. Since dynamic content enables dynamic documents, we believe that this problem is due to a missing definition of the term “*dynamic content*”. A precise definition for the term will avoid further misconceptions and will be indispensable to ensure the progress of further research.

Both Chapter 3 and Chapter 4 present a wide range of scenarios where dynamic documents are used. These scenarios are very diverse. Still a good definition must be independent and invariant. Figure 5.1 shows our current perspective on the term “*dynamic content*”. This presentation shows content surrounded by a shell. When we think of dynamic content, we imagine a extra layer that is responsible to react to the external influences on the content. The type of influence can be very diverse (we have displayed the most relevant ones with regard to our scenarios). The shell surrounding the content determines how the content should react to the impact. Therefore, dynamic content is able to cope with the requirements needed to present data that changes according to these influences.

Static content, on the other hand, has the property to be immune to external influences. This will ensure that the content will remain unchanged no matter the time, situation or any other external factor. Therefore, static content is useful to present information that never changes such as facts, historical events, etc.

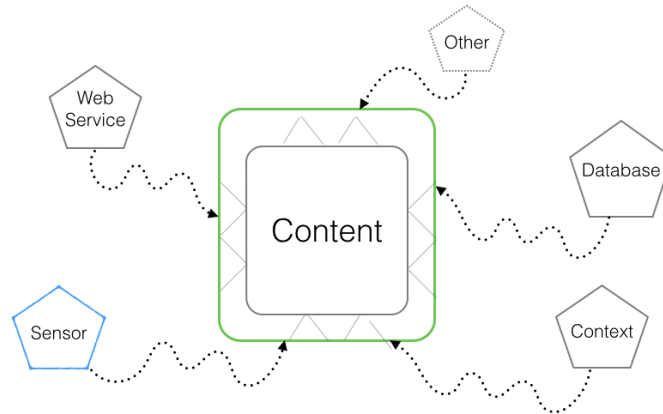


Figure 5.1: Dynamic content shell

Dynamic content and static content present data and information with a different perspective on change. Hence, we can state that they are each other’s counterpart. This gives rise to the following definition for the term “*dynamic content*”:

Definition 1. *Content that belongs to a document and has the capacity to change both itself and the structure of the document as a result of external factors that are linked to the content such as: a contextual element, an external resource, etc.*

The proposed definition is constructed on our findings in: the analysis of previous research, see Section 3.1; the analysis of document formats, see Section 3.2 and the development of several prototypes, see Section 4.2. We believe that this definition reveals the meaning of the term “*dynamic content*”, therefore it will be used throughout the thesis. Based on this proposed definition, we can define a “*dynamic document*” as follows:

Definition 2. *A digital document that contains dynamic content which allows the document to present variable information and to change its content and/or structure without the need for a manual editing effort.*

5.2 Conceptual Model of Dynamic Content

In addition to the lack of a formal definition, we also concluded that none of the solutions discussed in Section 3.1 relied on a clear conceptual model. The lack

of a conceptual model encourages different interpretations that could easily cause confusion amongst researchers and developers. Furthermore, a good conceptual model forms a stable basis for subsequent development of applications.

A main objective of this thesis is to create a conceptual model of dynamic content. Our model is developed as an extension of the RSL metamodel. The RSL metamodel is discussed in detail in Section 4.2.1. Since the RSL metamodel has support for linking, user rights, content adaptation and distribution, our model will inherit these features as well.

The next sections will discuss the different elements of our conceptualisation. Afterwards, we present the relations amongst these elements in the general overview of the model.

5.2.1 Elements of the Conceptual Model

Component

In Chapter 4, we have concluded that adding dynamic behaviour to a document can be a complex process. In order to cope with this complexity we have proposed to use a component-based approach. With this approach we divide the functionality over a set of subcomponents in order to achieve a specific behavior.

Consider a document that presents the point of interests (hereafter POI) of a city that changes according to the current location. For example, show nearby POI's. In order to achieve this, we can divide this behaviour in the following components:

- **POI:** A list of point of interests;
- **Location Filter:** A filter that filters on location;
- **Location:** A certain location;
- **Current Location:** The current location;
- **Link:** A link to specify the relation.

The **POI** which is a static resource⁴⁷ is **linked** to a specific **Location**. The **Filter** compares the **Current Location** with the specified **Location**. When the Current Location is within the span of the defined location, the appropriate **POI** will be presented. We can conclude that these components are sufficient enough to specify the dynamic behaviour of the content.

⁴⁷ Note that a static resource can also be a part of dynamic content

Since subcomponents are the building blocks to present dynamic content, they are an indispensable entity of our conceptual model on dynamic content. We believe that the components are best seen as a part of the resources of the documents. Therefore the `Component` entity will inherit from the `Resource` entity. This is presented in Figure 5.2.

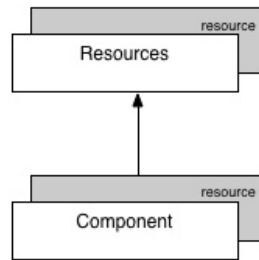


Figure 5.2: Conceptual model: Component

When developing the prototypes, we have observed that we can divide the concept of a component into two parts, namely `Components` and `Crosslets`. `Components` are more comparable with static content since they do not have the ability to change over time. Still, they play an important role in the updating process of the dynamic content object of which they are part. `Crosslets` on the other hand are more advanced components. The responsibility of a crosslets can be very diverse. The main concern of a crosslet is to perform a certain action that is needed for the content to become dynamic. Since a crosslet must perform a certain action, it may also be that it has to rely on other components, even other crosslets, to achieve its goal. How the `Crosslet` and `Component` entity relate to each other is presented in Figure 5.3.

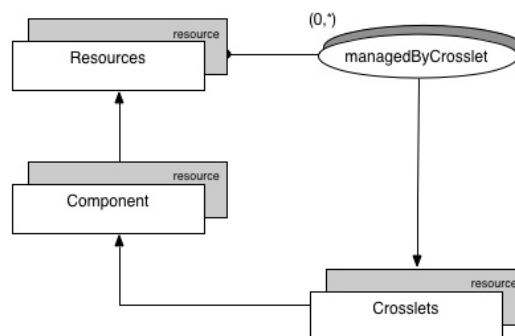


Figure 5.3: Conceptual model: Components with crosslets

Example: The following example will explain the idea and justify the need for the `Components` entity and the `Crosslet` entity.

Consider the example discussed above, namely, a digital document that has to present a particular text at a certain location. First of all, the text that must be displayed should be specified. This can be handled by a `Text` component which is a regular resource. Along with a `Text` component, we also need to specify a `Location` component. This component specifies at which location the `Text` component should become visible. Besides this location, we also need the current location of the device on which the document is read. Finally, we must use a filter that determines whether or not the current location is within a certain geographical area of the defined location. When the filter detects that the current location is within a certain range of the predefined location (i.e the `Location` component), then the filter will ensure that the defined text in the `Text` component will be displayed in the document.

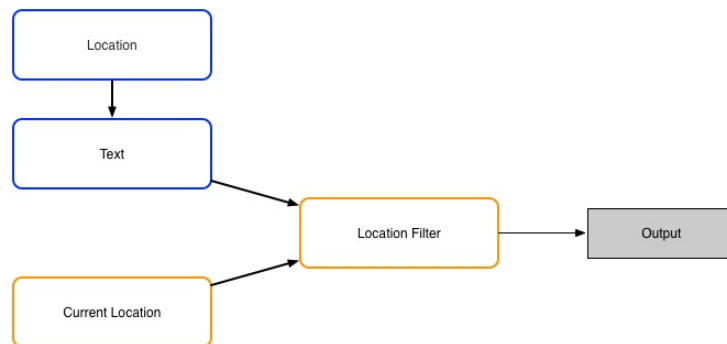


Figure 5.4: Conceptual model: Components of content example

A visualisation of the aforementioned components is presented in Figure 5.4. The regular components are the ones with a blue border. These components do not perform a certain operation, they are static resources that help constructing the dynamic content object. The crosslets are specified with an orange border. These components perform a particular task, namely, obtaining the current location and performing the filtering process. Note that the *output* box represents the actual result of the dynamic content.

Tube

Components play a crucial role in enabling dynamic content. However, without the ability to specify how they work together, they lose their value. So in order to achieve the dynamic behaviour, we must specify how these different subcomponents work together.

In order to specify how these different subcomponents are interlinked, we introduce a new entity in our conceptual model, namely, the *Tube* element. A *Tube* is used to specify the relation and cohesion between two subcomponents. This approach results in a structure where components are loosely coupled.

In Section 4.2.1, we introduced the *Link* entity as one of the main components of the RSL metamodel. The *Link* entity allows us to define an explicit relation between different entities. Even more, we have seen a subclass of the *Link* entity, namely the *StructuralLink*. This specific type of link is used to specify the structural relationship between resources.

Because a *StructuralLink* has the ability to specify the structure, they become the perfect tool to specify the composition structure of the dynamic content. However a property of *StructuralLink* (i.e they can have any number of input and output entities) can lead to an ambiguous composition of the dynamic component objects. As a result, we propose a subclass of the *StructuralLink*, namely the *Tube*. A *Tube*, as presented in Figure 5.5, is characterised by having only one input and one output entity. Additionally, these input and output entities must be instances of the *Component* class.

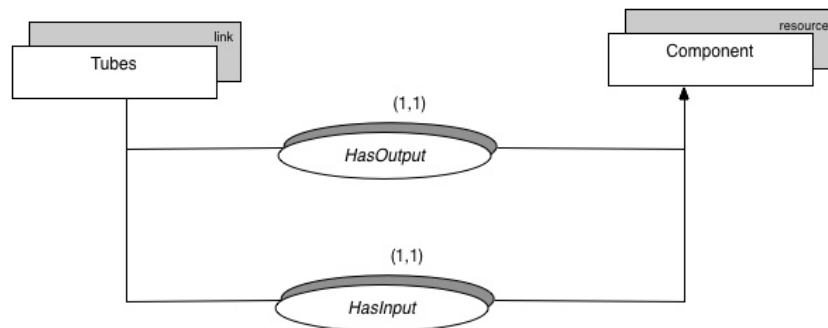


Figure 5.5: Conceptual model: Tubes

Example: In Section 5.2.1, we have presented an example of how a composition of components can be used to specify the dynamic behaviour of the content. While the concept of components was explained, we assumed that there was a certain structure that specified how these different components work together in order to achieve the goal of the dynamic content.

However, at that time we were unable to explicitly specify how these components were connected to each other. Fortunately, due to the introduction of the

Tube entity, we are now able to explicitly define the connections between the different components. A tube must be specified between each connected component. This results in a chain of connected components that together present the dynamic content object. A visualisation of the tubes of our previous example is presented in Figure 5.6, by means of purple arrows.

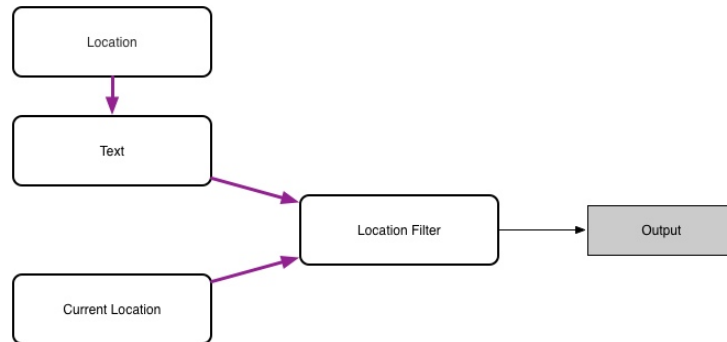


Figure 5.6: Conceptual model: Example of digital content - tubes

5.2.2 Tubes as First Class Objects

By defining the relationships amongst components by means of tubes, we obtain certain features that come along with the `Link` entity. Since a `Link` is a subclass of the `Entity` entity, our tubes will inherit all of its features such as linking, user rights, content adaptation and distribution. This makes our conceptual model even more expressive.

This advantage allows us, for example, to specify certain properties on the tubes themselves instead of integrating them in the components. This enables us to create a better encapsulation of the components, while offering a more advanced form of reusability. The following example explains how tubes can encourage the reusability of components. Meanwhile, we elaborate on the role they entail when creating dynamic content.

Assume that we want to present some informational text to the user. But instead of showing the same text to every user, we want to display a specific text depending on the current user's access rights. This could become useful when some parts of the document are meant to be private. Figure 5.7 presents such a scenario.

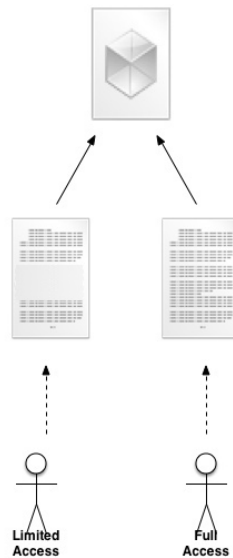


Figure 5.7: Conceptual model: Text specific access rights

Consider that we specify the access rules on the text components themselves. This approach is visualised in Figure 5.8 by means of green rectangles. If we then want to reuse these text components (using transclusion), then these access rules will also be present, intentionally or not. This approach may lower the form of component reusability.

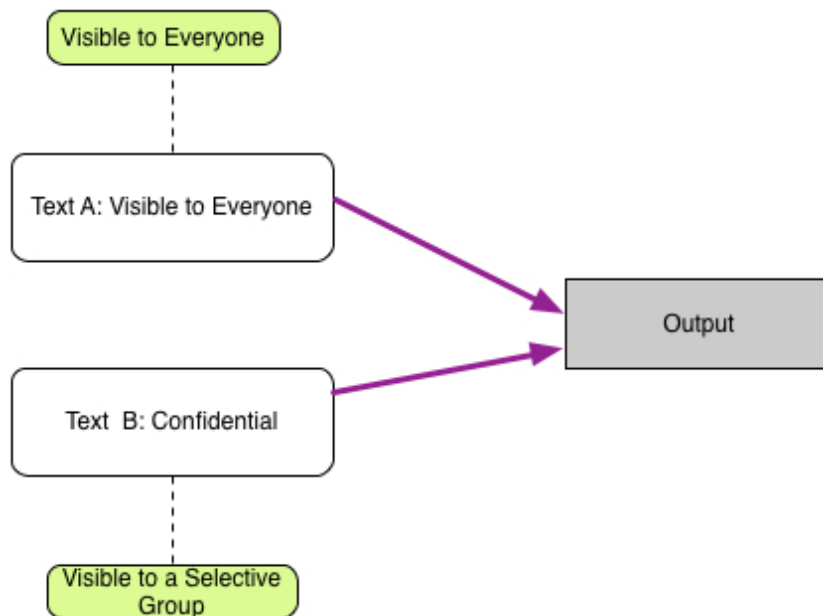


Figure 5.8: Approach 1: Access rights on the text

Fortunately, with the introduction of our tubes, we are able to delegate the user access rights to tubes instead. This second approach is visualised in Figure 5.9. Notice that this time the access rules are defined on the tubes instead of the text components.

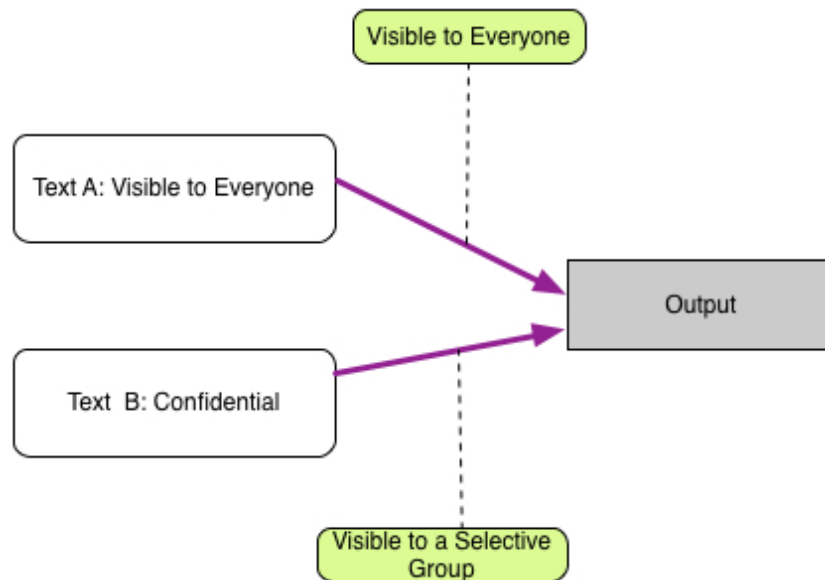


Figure 5.9: Approach 2: Access rights on the tubes

Our example focusses on the user access rights of the components. Still, we want to emphasise that this is only one of the many features that are possible for manipulating the tubes. It may also be the case that the tubes behaviour may depend on other contextual factors such as: temperature, light intensity, personal preference, etc.

We like to note that by introducing this second approach, we provide an alternative way of constructing dynamic content. It does not exclude the first approach by any means. The second approach only encourages the reusability of components.

5.2.3 Conceptual Model

We have observed that dynamic content is best described in smaller, more manageable, elements. As a result, we have introduced the notion of `Components` in our model. Components have their own responsibility to achieve the overall dynamic behaviour. Some of these components can be static and some components can perform an action. These last components are part of the `Crosslet` class.

Additionally, in order to specify the connection between the different components, we introduced the notion of Tubes. Tubes have the responsibility to specify the structure of the different components. This structure determines how the different components should work as a whole to satisfy the needs of the dynamic content object.

Besides the tube's ability to specify the composition of elements, they also make the model more powerful in form of expressiveness. Since tubes are a subclass of the `Link` entity, they also inherit all of its features as described in Section 4.2.1.

By combining the components and the links, we are able to specify a range of dynamic content. The overview of the full conceptual model is presented in Figure 5.10.

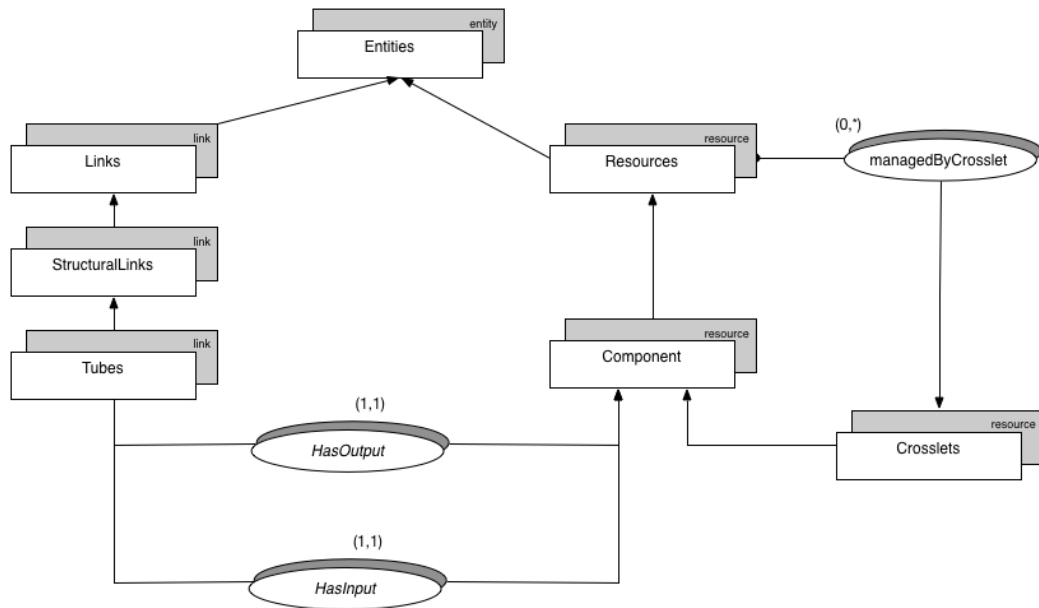


Figure 5.10: Conceptual model of dynamic content

This conceptual model has the ability to describe the essence of dynamic content. Furthermore, we believe that document formats, when tightly mapped to this metamodel, are able to specify dynamic content. In order to proof the efficiency of our model, we propose an online document editor that uses this model as basis to create dynamic documents, this is further discussed in Chapter 6.

5.3 Conclusion

In this chapter, we have presented our own definitions for the terms “*dynamic content*” and “*dynamic documents*”. In order to come to these definitions we have: analysed previous research, analysed a set of document formats and developed several prototypes. We believe that by formalising the meaning of the terms, we create a common ground for further research.

Additionally, we have developed a conceptual model for dynamic content. This model is inspired by the development of our prototypes. After analysing our prototypes, we concluded that by dividing the dynamic content in subcomponents, we are able to specify a range of dynamic content in a manageable manner.

However, in order to proof the contribution of our conceptual model, we need to test it. This will be further discussed in the next chapter.

Chapter 6

Proof of Concept: Document Editor

In this chapter, we discuss our document editor for dynamic documents. We start by presenting the objectives of this editor. Next, we present some related systems that inspired the development of our editor.

Afterwards, we discuss the technique used by our editor and we explain the architectural design of the implementation. Finally, we present a conclusion of our approach along with a comparison to the related systems.

6.1 Objectives of the Proof of Concept

The review performed in Section 3.2 has revealed that document editors must evolve in order to support the creation of dynamic documents. Therefore, the first and main objective is to develop a document editor that focusses on the creation of dynamic content. With this editor we seek to obtain a fine balance between *expressiveness* and *usability*.

In Chapter 5, we proposed a conceptual model for dynamic content. However, in order to justify its correctness we judge it necessary to test it. Therefore, as a second objective, we will use our conceptual model as basis to develop our document editor. This enables us to conclude whether we made a proper conceptualisation or not.

6.2 Related Systems

To inspire the design of our editor, we relied on three related systems with regard to their design approach. In this section, we will present the different approaches. Note that this section only serves as an introduction to each design approach. Due

to the scope of this thesis, it is not our intention to give a full detailed explanation for each system.

6.2.1 Squidy

Werner A. König, Roman Rädle and Harald Reiterer are the creators of Squidy [36]. Squidy is a library that unifies frameworks and toolkits in order to ease the design of natural user interfaces (hereafter NUI's). The library was developed since most interaction designers of NUI's are confronted with the following challenges:

- Practical knowledge must be available at every level (drivers, protocol, signal processing, etc.).
- They are concerned with both monologic tools as programming languages in a development environment.
- They experience slow prototyping due to the complexity. Additionally, it is hard to compare different techniques and solutions.

Their proposed design is based on high-level visual data flow programming combined with zoomable user interface concepts, see Figure 6.1.

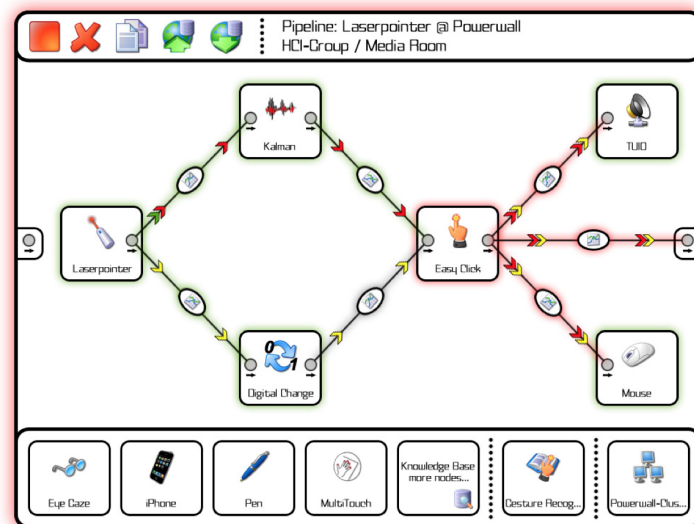


Figure 6.1: Squidy Design Environment [36]

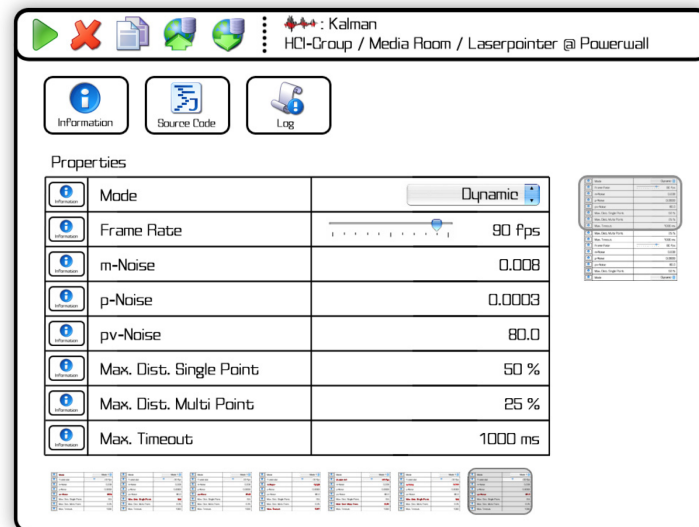


Figure 6.2: Squidy Design Environment: Zoomed in Kalman filter [36]

Since their interface is based on semantic zooming, we can zoom into a certain component. The zoomed in view of the Kalman filter is presented in Figure 6.2. This approach enables us to quickly navigate between different components of the NGU's. More importantly, users of this interface can quickly switch between different levels of detail.

6.2.2 Yahoo Pipes

Yahoo Pipes⁴⁸ is a composition tool based on pipes and filters that enables regular users to manipulate and mash up content from around the web. It relies on simple commands and components to create a specific output of data.

In order to create a mashup, Yahoo Pipes provide a rang of modules that are connected with one another by means of pipes, see Figure 6.3⁴⁹. The list of available modules is divided into the following categories:

Source: Used to retrieve one or multiple sources from the Internet such as RSS feeds, Flickr, etc.

Output Input: Allows the user to specify the parameters that serve as input for the pipes to another module.

⁴⁸<http://pipes.yahoo.com/pipes> last accessed on 14/05/2014

⁴⁹http://nick.typepad.com/blog/2007/02/youtunes_an_exa.html last accessed on 14/05/2014

Operators: Perform a filtering operation for the data that streams through the pipes.

URL: Provides the ability to manipulate URLs.

String: Helps in the process of manipulating and combining text strings.

Date: Allows the user to define and format a date.

Location: Converts text strings into geographical locations.

Number: Helps in performing basic arithmetic operations.

Deprecated: A specific type of module that will continue to work however it is encouraged to use the newly introduced module with improved functionality.

More details on the different modules can be found in their documentation⁵⁰.

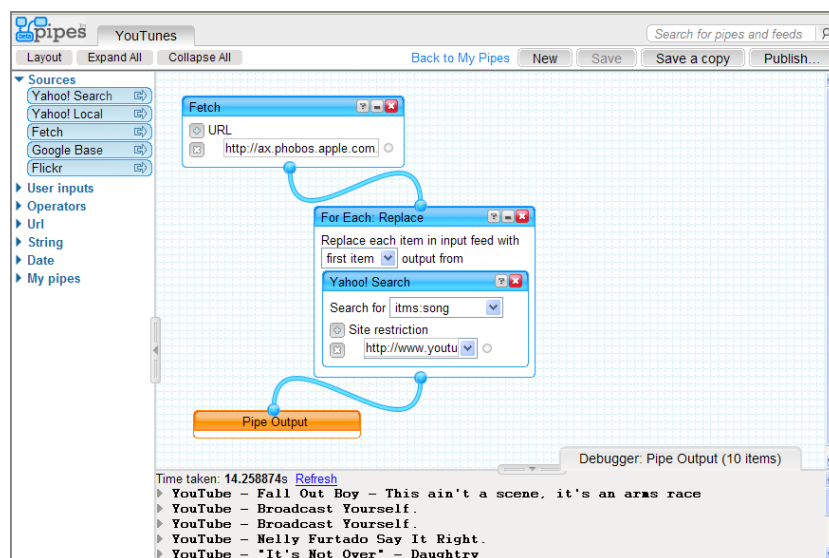


Figure 6.3: Yahoo Pipes: YouTube links for the top 10 song on iTunes

The composition of components by means of pipes provides a user-friendly manner to manage information coming from different sources. The result of the pipe system can later on be used to create applications or to embed them into a website. This last option is facilitated by introducing the concept of Pipe Badges. Pipe Badges are currently available in the following three types: map, image and list. Pipe Badges are integrated in the webpage by using HTML. As a result, they are

⁵⁰<http://pipes.yahoo.com/pipes/docs?doc=modules> (last accessed on 13/04/2014)

harder to use by regular people. However, the overall intention of Yahoo Pipes is to facilitate the creation of mashups by providing an interface that focusses on simplicity.

6.2.3 iBooks Author

The iBooks Author application is targeted towards a wide audience, ranging from publishing houses to individual writers. The editor allows for the creation and publication of iBooks documents. A main feature of iBooks Author is its ability to enable users to create dynamic multitouch elements, which they call *Widgets*. Widgets are integrated into the document in a WYSIWYG approach. With this approach, the document is presented with a close resemblance to its final presentation to the end user.

Widgets are the way of iBooks Author to allow users to present dynamic content. The current set of available widgets is described as follows:

Keynote presentations Users are able to embed a keynote presentation in their iBook document. A keynote presentation may contain custom animations.

Interactive Images Besides static images, iBooks Author also allows the creation of interactive images. These interactive images have support for callouts and pan-zoom features.

Interactive Galleries Instead of presenting one image, iBooks author also allows the user to define a set of images. This set of images is presented to the users as an interactive photo collection.

Scrolling Sidebars The ability to add relevant information to the current content.

Pop-Over Traditional static images can be enriched by the power of a pop-over that allows the user to present additional information such as images, text or other related data.

Media By introducing video and audio elements to the digital documents, the content of the document becomes more alive and interactive than when only static text and images are used.

Chapter Reviews iBooks documents allow users to test their knowledge by incorporating interactive questions. Questions can be of the following types: multiple choice, choose the correct image, label the image or a mix of all three.

3D image The iBooks Author application allows users to include 3D images. The editor itself does not provide the functionality to create a 3D object, it only allows the embedding of one.

HTML Modules Users are able to create their own widgets by making use of standard web technologies such as HTML5, CSS, JavaScript, etc. This enables more advanced users, with knowledge of these technologies, to create complex dynamic content.

Each widget has its own level of complexity. Since the HTML widget requires knowledge about current web technologies, it is likely the most complex one to use by regular users that have no experience with web technologies.

As a result, the usability factor of the HTML widget is rather low when compared to the other widgets. On the other hand, the HTML widget is the most expressive widget of them all. It is able to subsume all other widgets and it has the ability to provide additional functionalities. More information on the widgets and how to use them can be found on the official website⁵¹ of iBooks Author.

Additionally, the reusability of a widget is rather low. The only form of reuse is provided by a copy/paste operation of the widget or its internals (HTML files, CSS files, etc.). The developed widgets do not support reusability by transclusion.

Another important remark is that the configuration for each widget must be done in a separate window, named, the inspector window as presented in Figure 6.4.

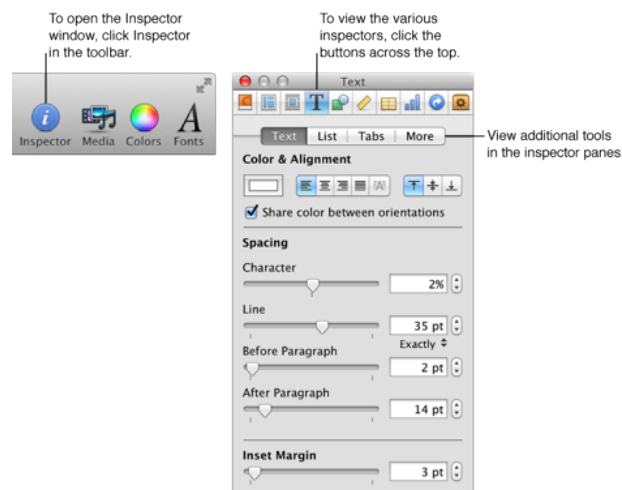


Figure 6.4: iBooks Author: Inspector window

⁵¹<http://www.apple.com/ibooks-author/gallery.html> (last accessed on 16/04/2014)

Since the configuration of a widget is performed in the inspector window, it is likely that it becomes easily overwhelming and too complex.

6.3 Document Editor

Content is one of the most important elements in the editing process of digital documents. Therefore, managing (creating, editing and deleting) both static and dynamic content should be greatly assisted by the editor. We believe that a WYSIWYG approach falls short when dynamic content becomes involved.

Section 6.2.3 revealed that the configuration of dynamic content with iBooks Author only receives a small amount of attention by means of a configuration screen, see Figure 6.4. As a result, the details of the dynamic behaviour are difficult to configure. Additionally, this approach does not provide an overview of how the dynamic content influences other parts of the document. The aforementioned limitations lead to a cumbersome process for managing dynamic behaviour in a digital document.

From this conclusion, we judged it necessary to investigate other approaches. A proper approach should facilitate the configuration of dynamic content and encourage the transparency of how it relates to other parts of the document.

In Section 6.2.1 and Section 6.2.2, we have discussed Squidy and Yahoo Pipes, respectively. We found that these techniques could be used to facilitate the management of dynamic content. The reason for this statement is that the so-called pipes and filters design pattern [23] aligns with our conceptual model of dynamic content. We believe that this close alignment will increase the usability of our editor.

By dividing the content in components and linking them to a whole, we believe that we are able to cope with the complexity of dynamic content. Additionally, we think that this approach encourages the individual configuration of each component while retaining an overall overview of the document. By doing so, this approach counters the limitations of the traditional WYSIWYG approach, wherefore we hope to create a fine balance between usability and expressiveness. An overview of our editor is described in the following section.

6.3.1 Approach of the Document Editor

User Authentication

In Section 4.1, we discussed a set of scenarios. From these scenarios we can derive that a document benefits from knowing who is viewing the document. Therefore, we have chosen to make the editor aware of the user by a simple authentication system. In order to support this feature, we enabled the users to create an account in the editor. After creating an account, users are able to authenticate themselves in the document editor. The logged in view of a user is presented in Figure 6.6. The current logged in user is “*Jochen*”.

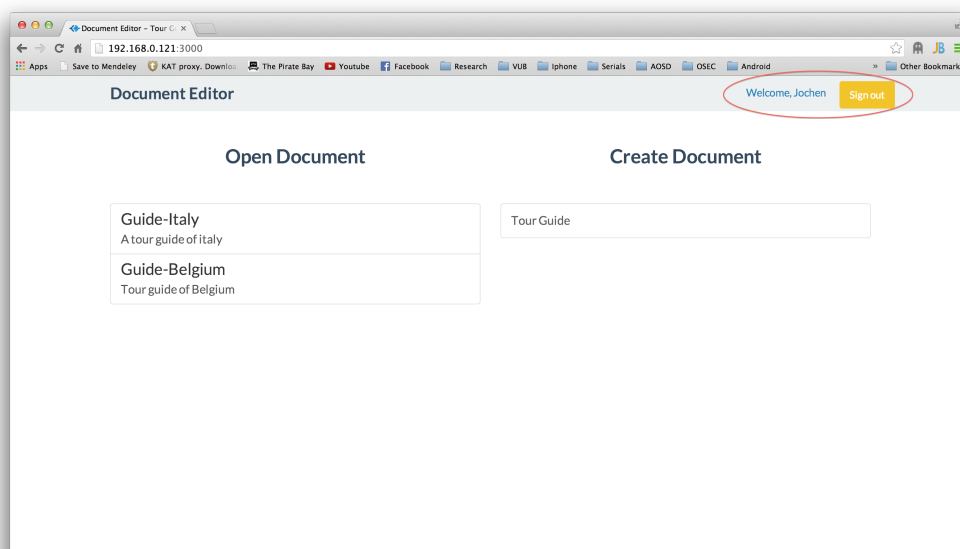


Figure 6.5: Document editor: Start view

Overview

Existing documents are presented on the left side of the window. These documents can be selected for further editing. In order to create a document, users have to select a template. The available templates are presented in the rightmost list of the page. The current implementation supports one template, namely the *Tour Guide* template. We made this choice since the primary concern of the editor is the editing phase. In principle, other templates should work accordingly.

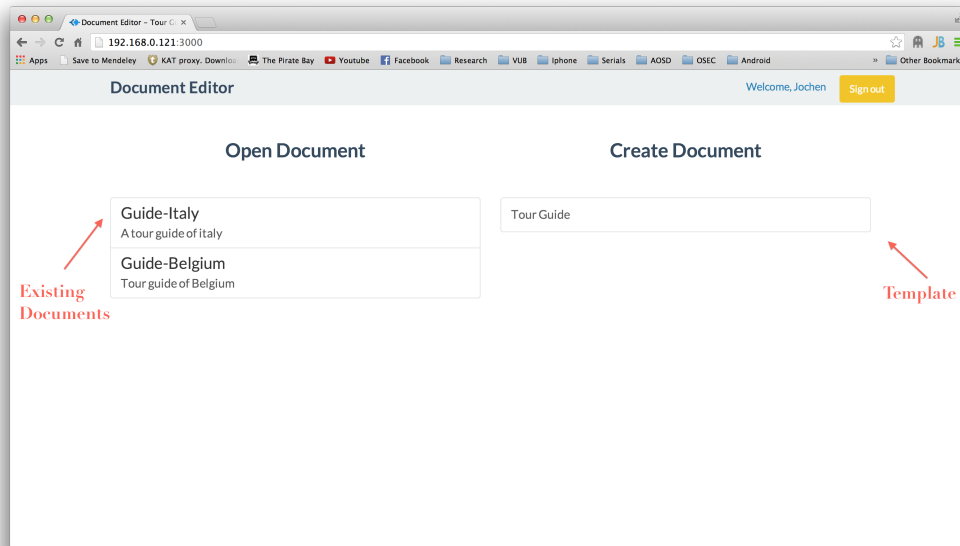


Figure 6.6: Document editor: Start view

The tour guide template presents a basic document with placeholders for the final content of the document. This idea is inspired by the virtual documents, see 3.1.7, of the multimodal documents. These placeholders are dynamic in the sense that they are able to update themselves according to the specification of the document defined by the author of the document. This means that, instead of having multiple pages in the document, this document will only have one page that adapts itself to the current requirements.

Document Creation

When creating a new document from a template, we are presented with the following window, see Figure 6.7. After specifying the details of the document, we submit the form. After the form is submitted, our *node.js* server, see Section 6.3.2, performs the necessary operations to create a document template in the iServer. Besides persisting the document template in the database, the *node.js* server also instructs the iServer to add a specific amount of placeholders to the document template. The amount of placeholders depends on the selected document template.

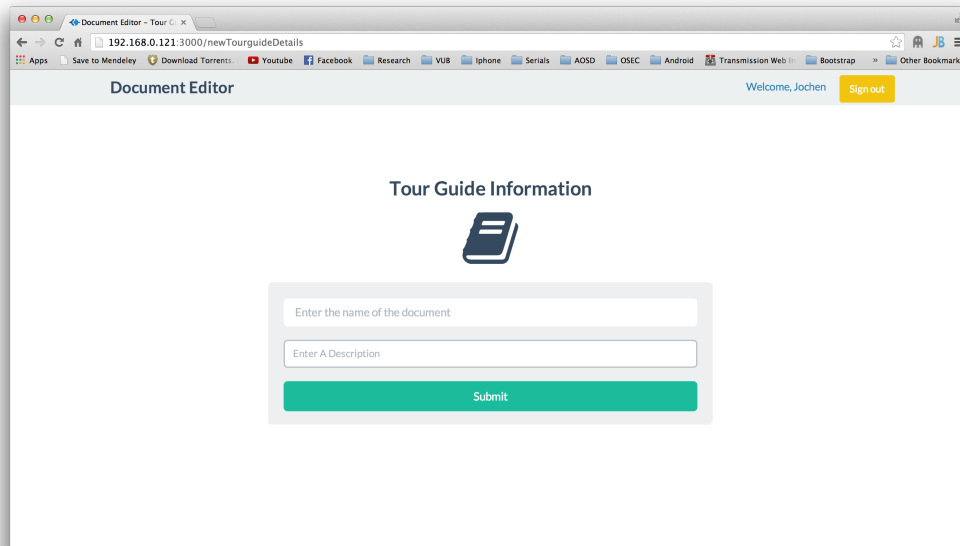


Figure 6.7: Document editor: Details of a document

A placeholder is a component of our document that has a predefined location in the document template. A placeholder can be compared to a “*Lorem Ipsum*” in web design, as it demonstrates where the content will be displayed when it is set. The user is responsible for using the editor to specify what the actual content should become. In the editing phase, the user will be able to view the placeholders of the document and specify their value.

Overview of a Template

When the document form is successfully submitted, the user is presented with the document template, as shown in Figure 6.8. We notice that this template contains seven placeholders. These seven placeholders will also be visible in the editing phase. From the moment that a placeholder is configured in the editing phase, it will become dynamic in the template. This means that the document will already show its dynamic behaviour. This is an important design choice since the author of the document can easily switch between an editing modus and a view modus. In order to proceed to the editing phase, the user has to click on the “*Edit Document*” button at the bottom of the page.

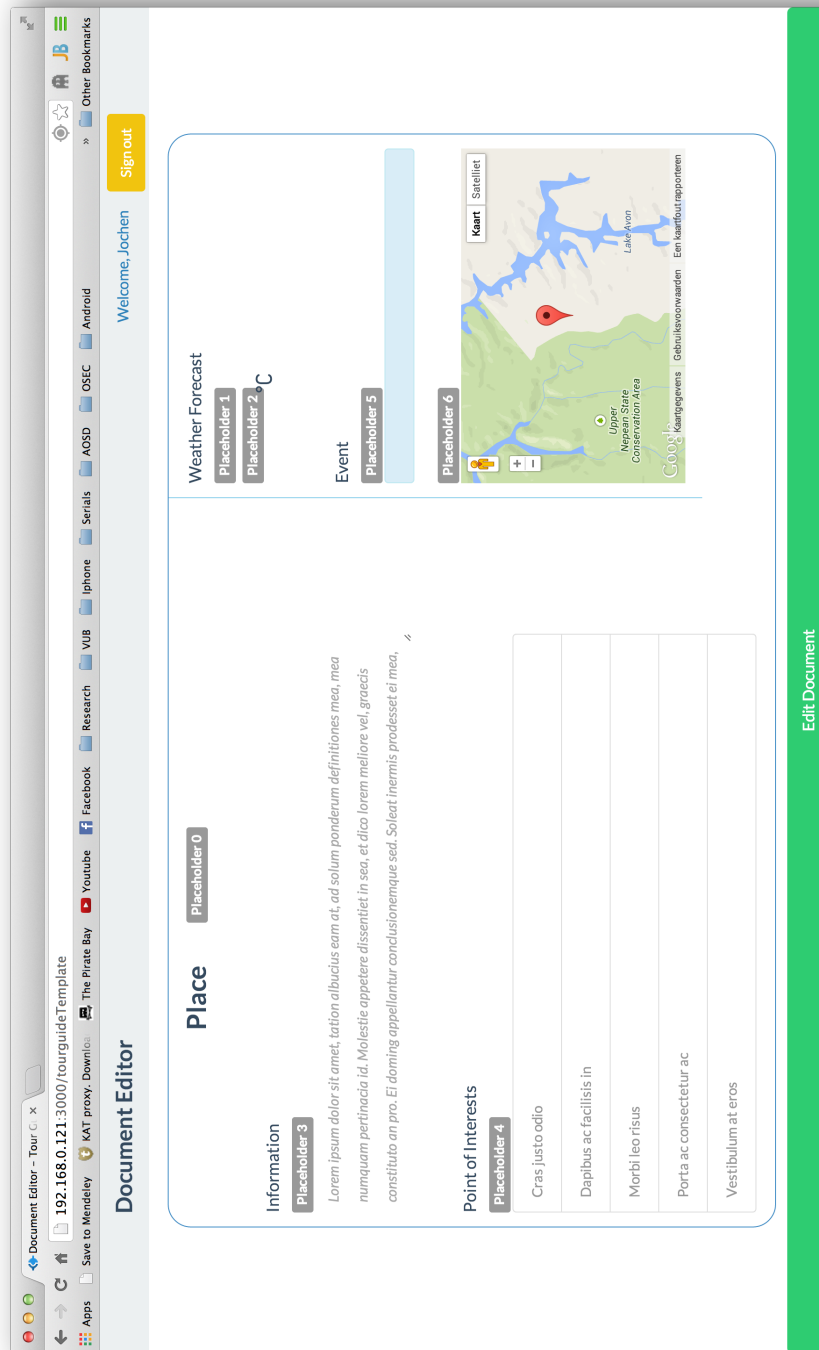


Figure 6.8: Document editor: Template view

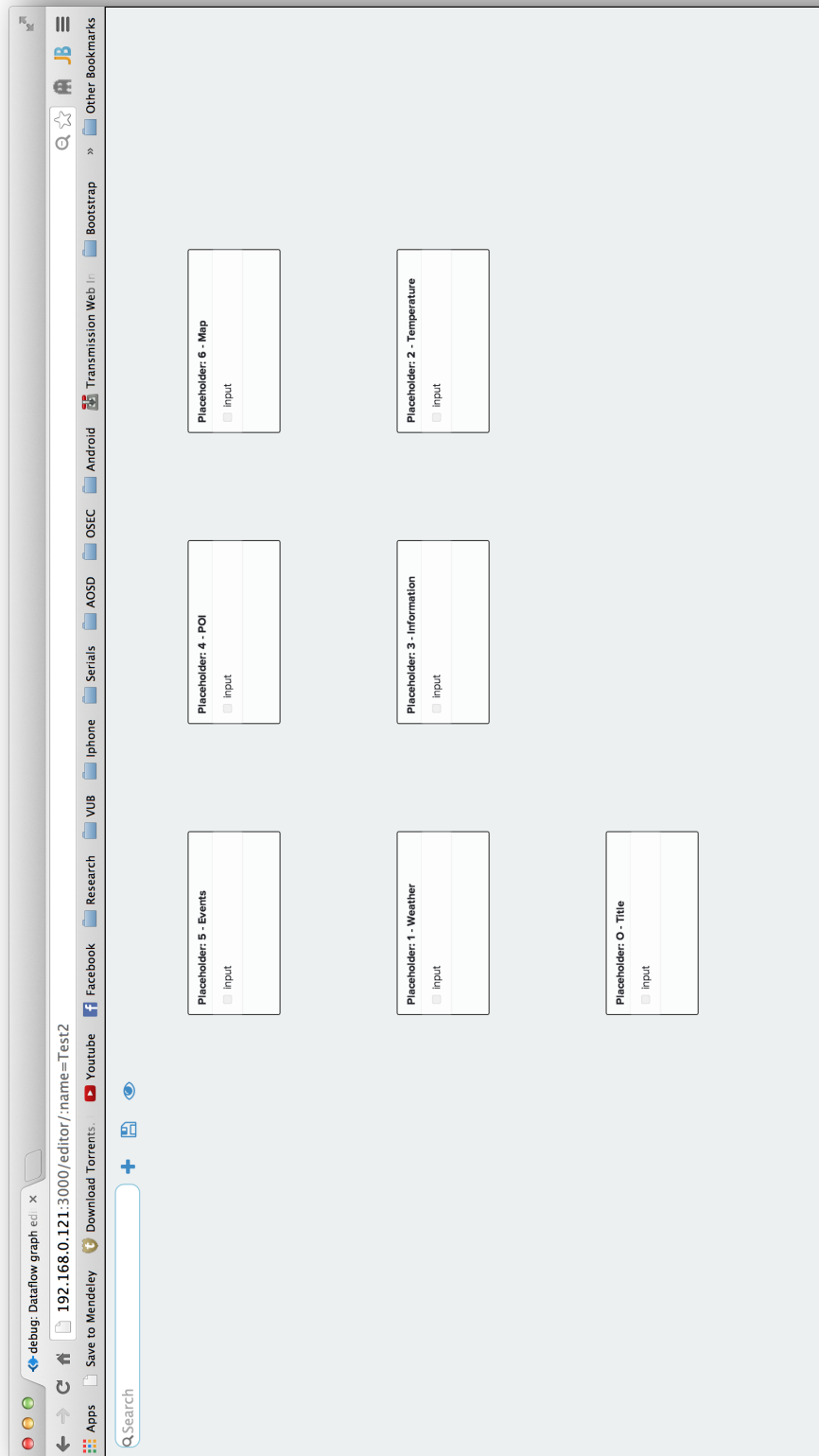


Figure 6.9: Document editor: Editing phase

Editing a document

The editing phase is presented in Figure 6.9. The editor presents the user with the available placeholders of the template. As already mentioned in our previous section, Section 6.3, we rely on the pipes and filters design pattern to edit a document.

The document template (see Figure 6.8) reveals that “Placeholder 4” presents the point of interests for a certain location. It could be useful to make the content of this placeholder dynamic. The content could for example rely on the current location of the device in combination with a web service, such as Yelp⁵², to provide real-time information of the current interesting places adjusted to the reader’s location.

In order to support dynamic content, our editor provides a range of components. The current implementation of our editor provides the following components:

Audio An audio component allows the user to specify a text that, when evaluated by the document editor, should be converted to speech.

Date A date component allows the user to specify a certain date with optional time arguments.

Light A light component allows the user to specify a flux value, which is of the type *double*.

Location A location component allows the user to specify an absolute geographical location. This is done by a specific pairing of latitude and longitude.

Filter A filter performs a certain filtering operation. It may have multiple inputs but always returns one output. In order to specify the filter, the user must zoom into the detail of it. This is inspired by the semantic zoom feature of Squidy [36].

Range Filter A range filter performs a filtering operation by using two values. If the value is higher then the minimal specified value and lower than the maximal specified value, then they are passed through the filter.

Current Location The current location components uses the device’s ability to obtain the current location from which the document is read.

Current Date The current date component is able to return the current date and time of the system on which the document is read.

⁵²<http://www.yelp.com/developers/documentation> (last accessed on 12/052014)

Current Light The current light component uses the device's ability to obtain the current ambient light value.

Webservice The web service component allows us to specify the connection between the client and a RESTful API. Besides the information needed to connect to the webserver, such as credential information, it also allows the user to add arguments that are used to specify the web service requests.

Text Component The text component allows the user to specify textual information.

This current set of components is already sufficient enough to satisfy a range of dynamic content. Furthermore, it would take little effort to specify new components.

In order to customise *Placeholder 4* to show up to date point of interests, we would require two components, namely: a *Current Location* component that provides the current location of the reading device and a *Webservice* component to retrieve the latest points of interests from a web service. These components are then linked to the appropriate placeholder. The linking of components results in a dataflow as shown in Figure 6.10.

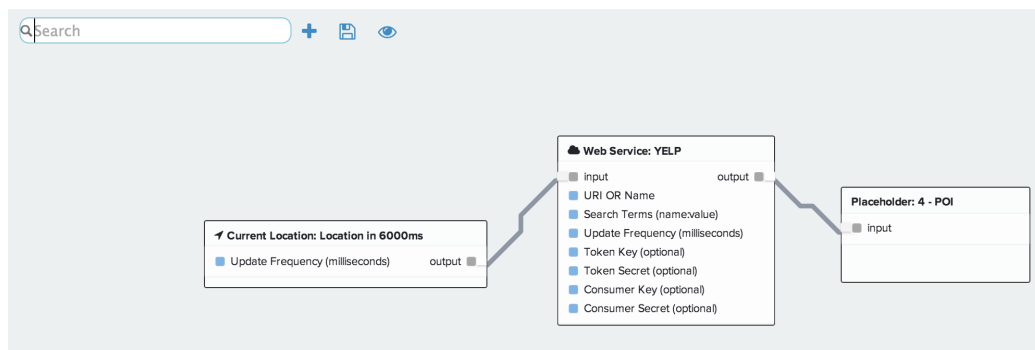


Figure 6.10: Document editor: Placeholder 4 composition

YELP

Type: Web Service

URI OR Name

Search Terms
(name:value)

Update
Frequency
(milliseconds)

Token Key
(optional)

Token Secret
(optional)

Consumer Key
(optional)

Consumer
Secret (optional)

Web Service: YELP

☐ input ☐ output

☒ URI OR Name

☒ Search Terms (name:value)

☒ Update Frequency (milliseconds)

☒ Token Key (optional)

☒ Token Secret (optional)

☒ Consumer Key (optional)

☒ Consumer Secret (optional)

Figure 6.11: Document editor: Web service configuration

Figure 6.11 presents how the user can configure a web service according to its needs. Note that the user is still responsible for looking up the details of the web service connection. However, we have tried to minimise the configuration process to a minimum.

Another interesting aspect of our document editor is that the tubes of our editor are able to use all the functionalities of the tubes as provided by our extension of the RSL metamodel. We have chosen to support a “*Visible To*” feature for the tubes. This feature makes use of the access rights feature of the RSL metamodel as described in Section 4.2.1. Since tubes are in essence also entities, we are able

to configure the “*AccessibleTo*” attribute of a tube. As a result, we can specify which users have access to a tube and which ones do not. In our developed editor, specifying the access rights can be done for each tube individually by specifying the user name in the “*Visible To*” property of the tube. The configuration window of an edge is visualised in Figure 6.12.

This feature can be handfult when for example the text has to vary according to the current user. Then the “*Visible To*” property of the tubes, as shown in 6.12, could be used to specify this.

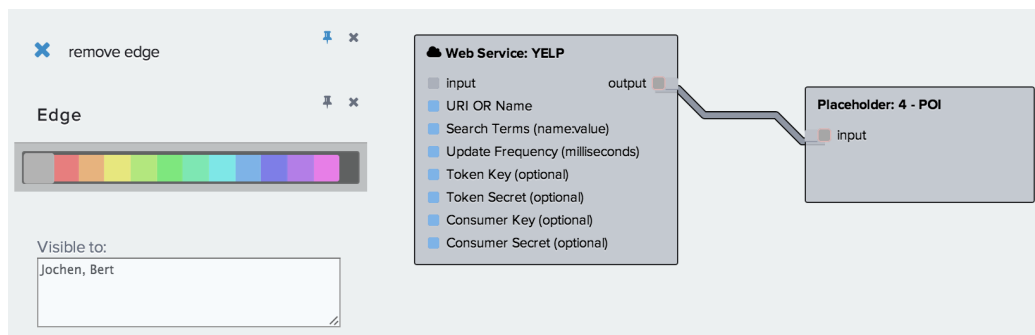


Figure 6.12: Document editor: Tube’s “*Visible To*” property

Figure 6.12 presents the properties of an edge. In this example, we have chosen to make the edge only visible to *Jochen* and *Bert*. Note that the properties of the edges can easily be extended with other properties of the RSL metamodel.

Since our document editor authenticates the current user via a login form, we are able to identify each user. When the user views the document, the editor will determine whether the user has access to a tube or not. If the user has access to a tube, the tube will become active. This ensures that the data from the “*text component*” is sent to the placeholder. If the user does not have access to the tube, the tube will seem invisible and the data will not be passed to the placeholder.

6.3.2 Architecture of the Document Editor

Back End

The document editor has been implemented as a web application and has a four tier architecture. The architecture is presented in Figure 6.13.

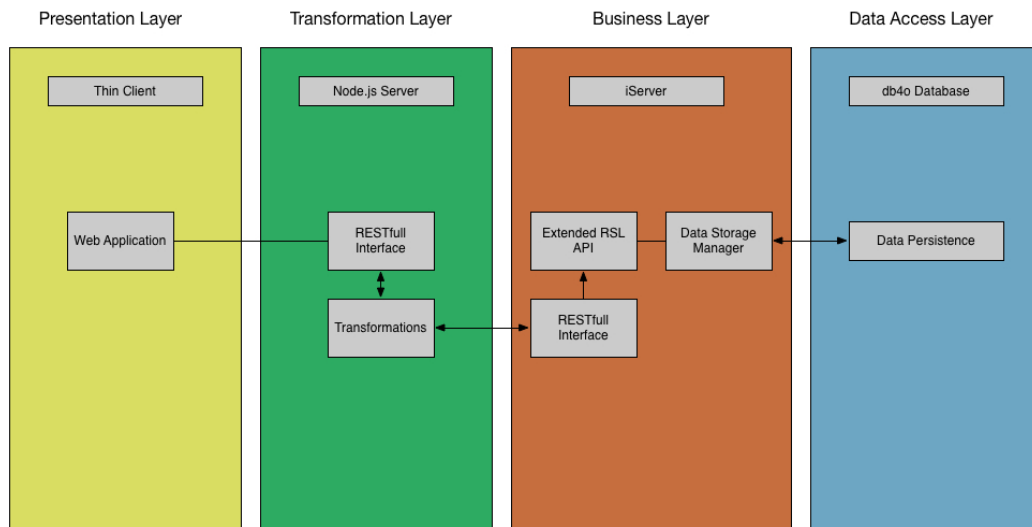


Figure 6.13: Architecture of the document editor

The client side of the document editor does not own a database store. Indeed, in order to obtain data it relies on a RESTful API provided by a *node.js* server⁵³. Besides listening to clients requests, this server also provides a transformation functionality between the data retrieved and sent between the client side and the iServer.

Since we did not want to alter the existing parts of the iServer implementation, the *node.js* server must perform a set of transformations. These transformation functionalities range from converting JSON data to performing web service requests to the iServer. Even more, by delegating the transformation functionality to a separate server, we lowered the coupling between our document editor and the iServer implementation.

By using the JSON format⁵⁴ to exchange data between the client, the *node.js* server and the iServer, we create a language neutral interface for the *node.js* server and the iServer implementation.

While implementing a *node.js* server, we made use of *Express*⁵⁵. *Express* is a web application framework for *node.js* that provides a robust set of features to build our multi-page web application. Another important library that was used during

⁵³<http://nodejs.org> (last accessed on 13/04/2014)

⁵⁴<http://www.json.org> (last accessed on 12/04/2014)

⁵⁵<http://expressjs.com> (last accessed on 12/04/2014)

the development of our *node.js* server is the *Socket.IO* library⁵⁶. This library assisted in the creation of a realtime application that works with every browser and on all mobile devices. It leverages the difficulties of setting up a socket connection between the client and the server by hiding the used transport mechanisms.

Our iServer implementation is built on top of the original implementation that is based on the RSL metamodel. We have extended this implementation with the entities of our developed conceptual model. The iServer is implemented using the Java programming language. The persistence layer is part of the iServer implementation and relies on the object oriented database named *db4o*⁵⁷. The general structure of the extended RSL metamodel is presented by a class diagram in Figure 6.14.

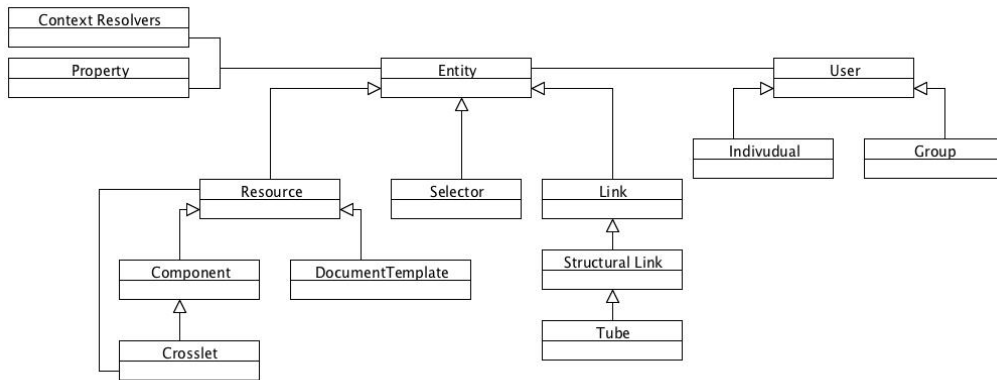


Figure 6.14: A general class diagram of the extended RSL metamodel implementation

Front End

The front end of the editor is developed by using the latest web technologies such as HTML5, JavaScript and CSS3⁵⁸. To create a good-looking user interface, we relied on a front end framework named Twitter Bootstrap⁵⁹ and more specifically, a Twitter Bootstrap Framework design and theme named Flat UI⁶⁰.

In order to facilitate the HTML webpage traversal, manipulation, event handling,

⁵⁶<http://socket.io> (last accessed on 12/04/2014)

⁵⁷<http://www.db4o.com> (last accessed on 16/04/2014)

⁵⁸<http://www.w3.org/TR/CSS> (last accessed on 12/04/2014)

⁵⁹<http://getbootstrap.com> (last accessed on 12/04/2014)

⁶⁰<http://designmodo.github.io/Flat-UI> (last accessed on 12/04/2014)

animation, and AJAX, we relied on jQuery⁶¹. jQuery is a lightweight API that is CSS3 compatible and is able to run in all popular browsers.

Essentially our document editor enables users to define the flow of data by means of filters and components. In order to facilitate the creation of such an editor, we made use of a framework called *meemoo*⁶². *Meemoo* is an open source framework that is still under heavy development. However, it serves as a good basis to start our editor with. We used *meemoo* to implement the editing phase of our document editor.

We extended the framework in such a way that it is able to handle custom components and custom edges. Some of our custom components are: *a date, a location, current location, current light intensity, a filter, a text component, etc.* We also extended the edges with the notion of accessibility. This allows users to specify which users have access to which edge.

As discussed in Section 6.3.1, these custom components and edges are useful to create a large range of dynamic components for digital documents. Furthermore, we added the functionality of loading and saving documents to the iServer. This functionality is delegated to the *node.js* server, as discussed in 6.3.2.

6.3.3 Comparison to Related Systems

This section will compare our approach to the related systems and we discuss how each system has influenced our design. Additionally, for the last two systems we explain how we solved some of their limitations.

Squidy

Squidy is used for facilitating the creation of NUI's while our editor focusses on the creation of dynamic content. This makes it difficult to compare the two systems. However, we can conclude that both Squidy and our editor rely on the pipes and filter design to facilitate a specific task. This facilitation is due to the fact that pipes allow us to divide the process in manageable entities to build a chain of processes in an intuitive manner. Since we observed in Section 5.2 that dynamic content is best divided in a set of subcomponents that are “*interchanged*”, this design perfectly fits for this task.

⁶¹<http://jquery.com> (last accessed on 12/04/2014)

⁶²<https://github.com/meemoo/dataflow> (last accessed on 12/04/2014)

Additionally, Squidy uses semantic zooming to present a detailed view for the components. We relied on the same technique to configure our filters. This allows us to create a filter at a certain level of detail while still having the option to view it as a black box. Instead of using a zooming action, we used an edit button on the filter itself to facilitate the selection process (see Figure 6.15).

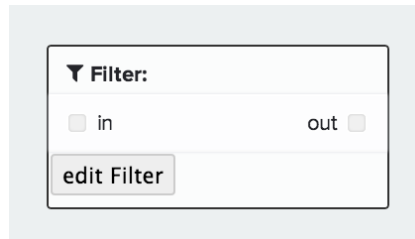


Figure 6.15: Filter Component

Yahoo Pipes

The purpose of Yahoo Pipes relates more to the purpose of our editor when we compare it to Squidy. Nevertheless, we believe that Yahoo Pipes is missing some crucial components to create rich forms of dynamic content. For example Yahoo Pipes does not provide the functionality to adapt content towards the context. This means that no matter the context, the result of a pipe will always be the same.

Since our previous research, see Section 3.1.3 has shown us that digital documents can benefit from context adaptation, we consider it a must to provide support for it. Our editor provides support for context adaptation by means of crosslets such as: *Current Location*, *Current Time* and *Current Light*. These components are capable of deriving the location, time and light intensity of the reading device to perform context adaptation.

Additionally, we have chosen to extend our pipes with properties. We believe that this is another shortcoming of Yahoo Pipes. We think that it is convenient to also have the ability to specify properties on top of the pipes, instead of the components alone. This way, the pipes themselves could become context-aware. Of course, we could argue that this feature could also be handled by introducing some additional components in combination with a filter. However, by specifying these details on a component level, we may lower their reusability. Hereby, we believe that in some cases it could be preferable to delegate some functionality to the tubes instead of the components.

Finally, another limitation of Yahoo Pipes is the following: *When a pipe become*

a part of a HTML document, the link between the document and the content becomes blurred. This is due to the fact that the creation of a pipe and the creation of an HTML document requires another editor. In our approach this is not the case. We have a clear link between the dynamic content and the resulting document by means of placeholder in template. As a result, the relation between the content and the document is always visible.

iBooks Author

iBooks Author and our editor share two characteristics, namely: 1) User oriented, suited to regular end users and 2) Provide support for dynamic content. iBooks Author relies on *Widgets* to enable non-developers to add dynamic behaviour to a document. In Section 3.2.9 we showed that each widget has its own level of complexity. The most complex and powerful widget is the HTML widget. Since this widget requires that users have some knowledge about web technologies, it is not targeted towards regular end users.

In order to obtain a fine balance between expressiveness and usability, we abandoned the traditional WYSIWYG approach. We relied on a design that matches with the essence of dynamic content and its complementary complexity, namely a pipes and filter dataflow design.

6.4 Conclusion

This chapter started by presenting some related systems that influenced the design of our editor. Afterwards, we presented our alternative editing approach based on a pipes and filter design. In order to justify this approach, we compared it to the previous discussed systems. Hereby, we emphasise their influences and limitations.

Since we are able to create complex forms of dynamic content while having a balance between usability and expressiveness, we believe that our approach outpaces the traditional WYSIWYG approach taken by iBooks Author. Additionally, Yahoo Pipes is missing some essential components to support more advanced forms of dynamic content. Nevertheless, a user evaluation should be performed to test this hypothesis.

Chapter 7

Conclusion and Future Work

In this chapter, the scope of our research is recapitulated and the main findings of our research with regard to the proposed research questions are summarised. Furthermore, the limitations of this thesis are considered along with some suggestions for future work.

7.1 Summary of the Research

This thesis has examined the progress of ubiquitous computing and its enabling technologies such as sensors and context-aware systems. In Chapter 2 we observed that the technical advances in ubiquitous environments will not become the main concern. On the other hand, the enormous amount of data acquired in these environments can become a greater problem. A concept that addresses this problem is ambient intelligence.

Ambient intelligence is a confluence of ubiquitous computing and artificial intelligence. Instead of having systems that compete for the user's attention, intelligent actions are performed to present the most relevant information to the user. The examination of ambient intelligence has led to the introduction of ambient documents. These documents are empowered with dynamic content to present information that changes depending on intelligent decisions. Unfortunately, these revolutionary documents were never built.

The literature related to this thesis is focussed around the developments of documents with dynamic content, the so-called dynamic documents. In Chapter 3, an analysis has revealed that the progress of documents with dynamic content is held back by the following problems: no clear definition for the term “*dynamic content*”, an absence of a clear conceptual model for dynamic content

and the lack of attention regarding the document creation process. Afterwards, a second analysis on document formats with regard to their support for dynamic content was performed. This second analysis showed that adding dynamic behaviour to documents is mainly done by using other technologies such as scripting languages. Therefore, the line between document authors and programmers has become blurred.

Additionally, this analysis has indirectly shown that document editors play an important role in the creation of dynamic documents. Unfortunately, current editors do not support the creation of dynamic content or their approach is not suited for regular end users. These two analyses have resulted in a set of research questions that must be addressed in order to ensure the progress of ambient documents and subsequently the progress of ambient intelligence and ubiquitous computing.

In order to investigate the purpose and possibilities of dynamic documents, a set of scenarios were devised. The development of three of these scenarios was discussed in Chapter 4. These developments assisted in defining the term “*dynamic content*” and subsequently “*dynamic documents*”, proposed in Chapter 5. Additionally, since these prototypes made use of a general conceptual model for hypermedia systems (i.e. Resource-Selector-Link metamodel), their insights helped with the creation of a conceptual model for dynamic content proposed Chapter 5.

After creating a conceptual model for dynamic content, it was used as a basis to develop a document editor for dynamic documents, see Chapter 6. Due to influences of other systems, the proposed editor relies on a pipe and filter design. In order to justify this design, we compared it to three other systems. Two of these systems also rely on this design approach, while the third system has a WYSIWYG design. The proposed editor tries to distinguish itself by focussing on the creation of dynamic content while keeping the balance between usability and expressiveness as good as possible.

The definition of the term “*dynamic documents*”, the creation of a conceptual model for “*dynamic documents*” and the creation of a document editor all contribute to the progress of ambient documents. The following section will emphasise the link between the contributions of this thesis and its proposed research questions.

7.1.1 Research Questions and Main Contributions

Research Question 1:

What does the term dynamic content mean? (See Chapter 5)

The analysis of Chapter 3 has shown that the term “*dynamic documents*” has received many diverse definitions over the years. However, a definition that is able to capture the meaning of dynamic content within these different schools of thoughts has never been proposed. In order to define the term, we analysed the state of the art and several relevant document formats. Afterwards, we used this knowledge as inspiration to propose a set of scenarios. Along with our scenarios, a categorisation was proposed for dynamic documents with their regard to adaptation. From this research, we were able to come up with a unified definition that is able to capture the meaning of dynamic content. This definition is presented in Chapter 5. Furthermore, this definition served as a basis to form a general definition for the term “*dynamic documents*”. This definition avoids further misunderstandings and serves as a common ground for further research. Therefore, it contributes to the field of research on dynamic documents.

Research Question 2:

Were there any attempts made in supporting dynamic content and why were they not successful? (See Chapter 3)

Chapter 3 has revealed that several attempts have been made to create the so-called dynamic documents. However, an analysis has shown that most of these solutions have a different point of view of what these documents should be. Additionally, none of these solutions were based on a clear conceptual model. As a result, comparing and reusing these solutions becomes a hard task. In order to rectify these problems, a conceptual model was proposed in Chapter 5. A conceptual model captures the essence of the concept as entities and relations. Therefore, this model contributes to the clarification of the concept of dynamic content and the creation of future dynamic documents.

Research Question 3:

To what extent do the current document formats enable the specification of dynamic content? (See Chapter 3)

Document formats have certain specifications on how the content must be specified. The second analysis of Chapter 3 has revealed that none of the analysed

formats have full support towards the specification of dynamic content. In fact, most document formats rely on other technologies such as JavaScript to enable dynamic content. Unfortunately, these technologies are too complex to be used by regular end users. However, one document format (namely iBooks) stood out because of its associated editor. This editor facilitated the creation of dynamic content by means of Widgets. Unluckily, in order to create advanced dynamic documents, the complexity of creating a custom widget does not meet the expected user friendliness for such a system. Therefore, an alternative document editor is proposed in Chapter 6. It relies on a pipe and filter design in order to create a balance between usability and expressiveness. As such, it contributes to the creation process of dynamic documents.

Research Question 4:

What are the struggles and challenges that come along when integrating dynamic content? (See Chapter 4)

A set of scenarios of dynamic documents were devised in Chapter 4, these scenarios led to the development of three prototypes. These prototypes helped to identify the struggles that come along when creating dynamic content. The main problem of creating dynamic content is the fact that a lot of different tasks must be performed to obtain a certain dynamic behaviour. Additionally, these different tasks closely rely on each other to work. Therefore, the prototypes were developed with a component based approach in order to cope with the complexity of dynamic content. By dividing the functionality of dynamic content in a set of sub-components, they became manageable units that are loosely coupled. As a result, they encourage reusability and a high cohesion. Furthermore, this approach also inspired the filter and pipe design of the editor proposed in Chapter 6. This contribution facilitates the creation of dynamic documents by providing a components based approach that relies on a clear conceptual model for hypermedia systems.

Research Question 5:

How can we attain a fine balance between expressiveness and usability with regard to the creation of dynamic content? (See Chapter 6)

In Chapter 3, the analysis of the iBook format has revealed that document editors play an essential role in creating dynamic documents. However, with iBooks Author, the complexity increased when more advanced functionalities were required. In order to cope with this complexity, an alternative design was proposed in Chapter 6. This editor seeks to find a balance between usability and expressive-

ness by means of a pipe and filter design. Since this design closely aligns with the conceptual model of dynamic content, we can expect that this approach is more intuitive to use even for complex use cases. Because the editor proposes an alternative editing process, it contributes to the creation process of dynamic documents.

7.1.2 Limitations

Concerning the Template-based Approach of the Document Editor

The current implementation of the document editor relies on a template to create dynamic documents. As a result, the design of the document is very restricted. This problem could be solved by providing a large set of diverse templates or by giving the users the possibility to create their own template. The first option will require us to think about which templates are necessary to cope with the user's demands. The second option, on the other hand, may complicate the authoring process. Since both solutions have their disadvantages, further research is needed to provide a more appropriate solution to this problem.

Concerning the Evaluation of the Document Editor

The design of the proposed editor has a pipe and filters interface. Therefore, users can create dynamic content as a chain of components that are linked to each other. This way of working closely aligns with our conceptual model of dynamic content, hence we can expect that this approach is more suitable to create dynamic documents than for example iBooks Author. However, in order to verify this hypothesis, a user evaluation must be performed. This evaluation could consist of a set of dynamic documents that must be created with a range of editors including our proposed editor. Depending on the resemblance to the provided document, we could compare each approach with a quantitative evaluation (quality of the document, creation time, overall expressiveness, etc.) and a qualitative evaluation (usability, preference, etc.)

7.2 Future Work

This thesis has made an effort in simplifying the process of creating dynamic documents by means of a pipes and filter design. The current implementation provides a set of components to create a range of dynamic documents. Still, in order to provide more advanced dynamic content, this set has to be extended. As discussed in Chapter 2, middlewares for sensor networks already exist. Therefore, our document editor could be extended to work with these frameworks by allowing users to specify details for direct sensor integration.

Since our document editor enables users to specify a web service it may be useful to investigate how we could improve this process. Working with a web service, requires users to study its documentation and write special purpose code to handle it. We could explore other options to simplify this process, for example use web servers GUIs [35].

Currently, it is the task of the user to configure the component of a document. This process is prone to errors and is time consuming. In order to facilitate this process, we could enable third party developers to develop predefined components. These components should then have to comply to the specifications of our technical implementations. An ontology could be used to enhance the collaboration between the different systems.

As discussed in the previous section, a limitation of the proposed editor is that it uses a template-based approach. However, in order to create more freedom with respect to document design, other approaches must be examined. Furthermore, in order to resolve the second limitation of our editor, we could perform a user evaluation in order to examine the effectiveness of our design approach.

Another aspect that would be interesting to investigate is how we could extend our editor to provide support for static mediums such as paper. Since paper, on its own, is not capable of displaying dynamic content, it would be interesting to investigate how we could solve this problem by providing alternative presentations.

Bibliography

- [1] S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML Project: an Overview. *The VLDB Journal*, 17(5):1019–1040, 2008.
- [2] K. Ashton. That Internet of Things Thing. *RFiD Journal*, 22:97–114, 2009.
- [3] P. M. Atkinson and A. R. L. Tatnall. Neural Networks in Remote Sensing. *International Journal of Remote Sensing*, 18(4):699–709, 1997.
- [4] J. C. Augusto. Ambient Intelligence: The Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence. In A. Schuster, editor, *Intelligent Computing Everywhere*, pages 213–234. Springer London, 2007.
- [5] J. C. Augusto. Ambient Intelligence: Basic Concepts and Applications. In J. Filipe, B. Shishkov, and M. Helfert, editors, *Software and Data Technologies*, volume 10 of *Communications in Computer and Information Science*, pages 16–26. Springer Berlin Heidelberg, 2008.
- [6] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. Fusing Mobile, Sensor, and Social Data to Fully Enable Context-Aware Computing. In *Proceedings of the Eleventh ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, HotMobile ’10, pages 60–65, Annapolis, Maryland, 2010.
- [7] H. Beadle, G. Q. Maguire Jr., and M. T. Smith. Using Location and Environment Awareness in Mobile Communications. In *Proceedings of the International Conference on Information, Communications and Signal Processing, ICICS, Part 3*, volume 3, pages 1781–1785, Singapore, September 1997. IEEE.
- [8] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [9] Y. Billibon. On Sensor Frameworks for Pervasive Systems. In *Proceedings of the International Conference on Software Engineering*, Limerick, Ireland, June 2000.

- [10] P. J. Brown. The Stick-e Document: A Framework for Creating Context-Aware Applications. In *Proceedings of the Electronic Publishing*, pages 259–272, Palo Alto, September 1996. IFIP.
- [11] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–1290, 1996.
- [12] V. Bush. As We May Think. *Atlantic Monthly*, 176(1):101–108, July 1945.
- [13] F. Cabitza and I. Gesso. Web of Active Documents: An Architecture for Flexible Electronic Patient Records. In A. Fred, J. Filipe, and H. Gamboa, editors, *Biomedical Engineering Systems and Technologies*, volume 127 of *Communications in Computer and Information Science*, pages 44–56. Springer Berlin Heidelberg, 2011.
- [14] J. C. Cardoso and R. José. A Framework for Context-Aware Adaptation in Public Displays. In *Proceedings of the Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems*, OTM ’09, pages 118–127. Springer-Verlag, 2009.
- [15] A. Celentano and O. Gaggi. Context-aware Design of Adaptable Multimodal Documents. *Multimedia Tools and Applications*, 29(1):7–28, 2006.
- [16] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient Intelligence: Technologies, Applications and Opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.
- [17] D. J. Cook and S. K. Das. How Smart Are Our Environments? An Updated Look at the State of the Art. *Pervasive Mobile Computing*, 3(2):53–73, 2007.
- [18] P. De Bra, G.-J. Houben, and H. Wu. AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia. In *Proceedings of the ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, February 1999.
- [19] A. K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, February 2001.
- [20] B. Dumas, M. Solórzano, and B. Signer. Design Guidelines for Adaptive Multimodal Mobile Input Solutions. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI ’13, pages 285–294, Munich, Germany, August 2013.

- [21] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive Design Optimization of Wireless Sensor Networks using Genetic Algorithms. *Computer Networks*, 51(4):1031–1051, 2007.
- [22] M. F. Fudzee and J. Abawajy. A Classification for Content Adaptation System. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 426–429, Linz, Austria, 2008.
- [23] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.
- [24] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications (MONET)*, 7(5):341–351, 2002.
- [25] S. Greenberg and C. Fitchett. Phidgets: Incorporating Physical Devices into the Interface. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 209–218, Santa Barbara, California, 2001.
- [26] K. Grønbaek, J. F. Kristensen, P. Orbaek, and M. A. Eriksen. Physical Hypermedia: Organising Collections of Mixed Physical and Digital Material. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 10–19, Nottingham, UK, 2003.
- [27] F. Halasz, M. Schwartz, K. Grønbaek, and R. H. Trigg. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, 1994.
- [28] K. Hallenborg. Contextual Interfacing: A Sensor and Actuator Framework. In L. T. Yang, M. Amamiya, Z. Liu, M. Guo, and F. J. Rammig, editors, *Embedded and Ubiquitous Computing, EUC 2005*, volume 3824 of *Lecture Notes in Computer Science*, pages 846–857. Springer Berlin Heidelberg, 2005.
- [29] V.-N. Hamed, Z. Kamran, and N. Nematbakhsh. Context-Aware Middleware Architecture for Smart Home Environment. *International Journal of Smart Home*, 7:77–86, 2013.
- [30] K. Henriksen and R. Robinson. A Survey of Middleware for Sensor Networks: State of the Art and Future Directions. In *Proceedings of the International Workshop on Middleware for Sensor Networks, MidSens '06*, pages 60–65, Melbourne, Australia, 2006.

- [31] P. Holroyd, P. Watten, and P. Newbury. Why Is My Home Not Smart? In Y. Lee, Z. Bien, M. Mokhtari, J. Kim, M. Park, J. Kim, H. Lee, and I. Khalil, editors, *Aging Friendly Technology for Health and Independence*, volume 6159 of *Lecture Notes in Computer Science*, pages 53–59. Springer Berlin Heidelberg, 2010.
- [32] S. S. Intille, J. Lester, J. F. Sallis, and G. Duncan. New Horizons in Sensor Development. *Medicine and Science in Sports and Exercise*, 44:24–31, January 2012.
- [33] D. Joshua and F. Susan. Active Documents: Changing How the Enterprise Works. *IDC: Internal Document*, 1:1–9, 2003.
- [34] G. Junzhong and C. Gong-Chao. Design of Physical and Logical Context Aware Middleware. *IJSIP*, 5:113–130, 2012.
- [35] M. Kassoff, D. Kato, and W. Mohsin. Creating GUIs for Web Services. *IEEE Internet Computing*, 7(4):66–73, 2003.
- [36] W. A. König, R. Rädle, and H. Reiterer. Squidy: A Zoomable Design Environment for Natural User Interfaces. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems CHI EA '09*, pages 4561–4566, Boston, MA, USA, 2009.
- [37] B. Kreller, A. S.-B. Park, J. Meggers, G. Forsgren, E. Kovacs, and M. Rosinus. UMTS: A Middleware Architecture and Mobile API Approach. *IEEE Personal Communications*, 5(2):32–38, April 1998.
- [38] B. Krishnamachari and S. Iyengar. Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. *IEEE Transactions on Computers*, 53(3):241–250, March 2004.
- [39] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A Survey of Mobile Phone Sensing. *Communication Magazine*, 48(9):140–150, September 2010.
- [40] Y. Lee, S. S. Iyengar, C. Min, Y. Ju, S. Kang, T. Park, J. Lee, Y. Rhee, and J. Song. MobiCon: A Mobile Context-Monitoring Platform. *Communications of the ACM*, 55(3):54–65, March 2012.
- [41] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury. StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 351–360. ACM, 2012.

- [42] N. Marquardt and S. Greenberg. Distributed Physical Interfaces with Shared Phidgets. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pages 13–20, Baton Rouge, Louisiana, 2007.
- [43] F. Mattern and C. Floerkemeier. From the Internet of Computers to the Internet of Things. In *Active Data Management to Event-Based Systems and More*, volume 6462 of LNCS, pages 242–259. Springer Berlin, Heidelberg, Germany, 2010.
- [44] T. Nelson. *Literary Machines*. Swarthmore, Pennsylvania, 3rd edition, 1981.
- [45] M. C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of ER '93, 12th International Conference on the Entity-Relationship Approach*, pages 390–401. Springer Berlin Heidelberg, Arlington, USA, December 1993.
- [46] M. C. Norrie, B. Signer, and N. Weibel. General Framework for the Rapid Development of Interactive Paper Applications. In *Proceedings of CoPADD 2006, 1st International Workshop on Collaborating over Paper and Digital Documents*, pages 9–12, Banff, Canada, November 2006.
- [47] G. M. P. O'Hare, M. J. O'Grady, C. Muldoon, and C. A. Byrne. Ambient Documents: Intelligent Prediction for Ubiquitous Content Access. In *Proceedings of the 4th International Conference on Universal Access in Human-computer Interaction: Ambient Interaction*, pages 971–979, Beijing, China, 2007. Springer-Verlag.
- [48] T. A. Phelps and R. Wilensky. Toward Active, Extensible, Networked Documents: Multivalent Architecture and Applications. In *Proceedings of the First ACM International Conference on Digital Libraries*, pages 100–108, Bethesda, Maryland, USA, 1996.
- [49] H. Raffler. Other Perspectives on Ambient Intelligence. *Password Magazine*, 2006.
- [50] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05*, pages 192–204, San Diego, California, USA, 2005.
- [51] M. Reitz and C. Stenzel. Minerva: A Component-Based Framework for Active Documents. *Electronic Notes in Theoretical Computer Science*, 114:3–23, 2004.

- [52] J. Rouillard. Contextual QR Codes. In : *ICCGI '08. The Third International Multi Conference on Computing in the Global Information Technology*, pages 50–55, Washington DC, USA, August 2008. IEEE Press.
- [53] F. Sadri. Ambient Intelligence: A Survey. *ACM Computing Surveys*, 43(4):36:1–36:66, October 2011.
- [54] A. M. J. Sarkar, K. Hasan, Y.-K. Lee, S. Lee, S. Zabir, and S. Muhammad. Distributed Activity Recognition Using Key Sensors. In *Proceedings of the 11th International Conference on Advanced Communication Technology*, volume 3 of *ICACT'09*, pages 2245–2250, Gangwon-Do, South Korea, February 2009. IEEE Press.
- [55] A. Schmidt, M. Beigl, and H. W. Gellersen. There is More to Context than Location. *The Interactive Applications of Mobile Computing (IMC '98)*, 23(6):893–901, November 1999.
- [56] A. Sheth, C. Henson, and S. S. Sahoo. Semantic Sensor Web. *IEEE Internet Computing*, 12(4):78–83, July 2008.
- [57] B. Signer. *Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces*. PhD thesis, ETH Zürich, Switzerland, 2005.
- [58] B. Signer and M. C. Norrie. As We May Link: A General Metamodel for Hypermedia Systems. In *Proceedings of ER 2007, 26th International Conference on Conceptual Modeling*, pages 359–374, Auckland, New Zealand, November 2007. Springer-Verlag.
- [59] B. Signer and M. C. Norrie. PaperPoint: A Paper-Based Presentation and Interactive Paper Prototyping Tool. In *Proceedings of TEI 2007, 1st International Conference on Tangible and Embedded Interaction*, pages 57–64, Baton Rouge, USA, February 2007. ACM.
- [60] C. F. Sørensen, M. Wu, T. Sivaharan, G. S. Blair, P. Okanda, A. Friday, and H. Duran-Limon. A Context-aware Middleware for Applications in Mobile Ad Hoc Environments. In *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing*, MPAC '04, pages 107–110, Toronto, Canada, October 2004. ACM.
- [61] R. Spinrad. Dynamic Documents. *Harvard University Information Technology Quarterly*, 7:15–18, 1988.

- [62] M. Srivastava, R. Muntz, and M. Potkonjak. Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-Solving Environments. In *Proceedings of the ACM SIGMOBILE 7th Annual International Conference on Mobile Computing and Networking*, pages 132–138, Rome, Italy, July 2001. ACM.
- [63] E. M. Tapia, S. S. Intille, and K. Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Proceedings of Pervasive 2004*, volume 3001 of *Lecture Notes in Computer Science*, pages 158–175. Berlin/Heidelberg, Germany, 2004.
- [64] A. Tayeh. A Metamodel and Prototype for Fluid Cross-Media Document Formats. Master’s thesis, Vrije Universiteit Brussel, 2011-2012.
- [65] D. B. Terry and D. G. Baker. Active Tioga Documents: An Exploration of Two Paradigms. *Electronic Publishing—Origination, Dissemination and Design*, 3(2):105–122, May 1990.
- [66] D. Thevenin and J. Coutaz. Plasticity of User Interfaces: Framework and Research Agenda. In *Proceedings of IFIP Conference on Human Computer Interaction Interact ’99*, volume 99, pages 110–117, Edinburgh, Scotland, August 1999. IOS Press Publication.
- [67] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A Taxonomy of Wireless Micro-Sensor Network Models. *Mobile Computing and Communications Review*, 6(2):28–36, April 2002.
- [68] V. Tsiatsis, A. Gluhak, T. Bauge, F. Montagut, J. Bernat, M. Bauer, C. Villalonga, P. Barnaghi, and S. Krco. *The SENSEI Real World Internet Architecture*, In *Towards the Future Internet - Emerging Trends from European Research*, pages 247–256. IOS Press, 2010.
- [69] G. M. Voelker and B. N. Bershad. Mobisaic: An Information System or a Mobile Wireless Computing Environment. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 375–395. Springer, Santa Cruz, USA, December 1996.
- [70] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison. Bridging Physical and Virtual Worlds with Electronic Tags. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 370–377, Pittsburgh, Pennsylvania, USA, 1999.

- [71] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.
- [72] R. Want and D. M. Russell. Ubiquitous Electronic Tagging. *IEEE Distributed Systems Online*, 1(2), September 2000.
- [73] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An Overview of the PARCTab Ubiquitous Computing Experiment. *IEEE Personal Communications*, 2:28–43, December 1995.
- [74] A. Weber, H. M. Kienle, and H. A. Müller. Live Documents with Contextual, Data-Driven Information Components. In *Proceedings of the 20th Annual International Conference on Computer Documentation, SIGDOC '02*, pages 236–247, Toronto, Canada, October 2002.
- [75] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, September 1991.
- [76] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, July 1993.
- [77] M. Weiser, R. Gold, and J. S. Brown. The Origins of Ubiquitous Computing Research at PARC in the Late 1980s. *IBM Systems Journal*, 38(4):693–696, 1999.
- [78] K. Whitehouse, F. Zhao, and J. Liu. Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data. In *Proceedings of the Third European conference on Wireless Sensor Networks, EWSN'06*, pages 5–20, Zurich, Switzerland, February 2006.

Appendix A

Scenarios

In this chapter, we elaborate on the scenarios proposed in Chapter 4. For each scenario, we discuss its purpose, its applicability and the external sources involved.

A.1 Active Storytelling

Purpose: Involve children in a story by adapting the storyline to the current environment. Make children aware of dangers and help them discover physical phenomenas such as the change of temperature.

How: Sensors are employed within and around the house to detect changes in the environment. Changes to the environment are reflected in the storyline. Examples of such changes could be the following:

- When it starts raining outside, the story being told could make the child aware that it is useful to bring an umbrella along;
- When the temperature is high and the sun is at its highest point the storyline could emphasis the dangers of a sunburn.

Involved sources: The sources for this scenario can be very varied and are free to choose. Examples are:

- Temperature sensor;
- Light sensor;
- Humidity sensor;
- GPS.

A.2 Dynamic Restaurant Menu

Purpose: Prices of a diner menu autonomously change towards the current environment. These varied prices could for example allow the restaurant to cope with extra costs.

How: Dynamically change the prices of the menu with respect to environmental factors such as temperature, light-intensity, humidity and location. Examples of such use could be the following:

- When you are eating outside on a terrace of a restaurant with a view on the lake and the sun is shining, the price will be higher than when you would dine inside;
- Food and beverages are recommended according to the current season.

Involved sources: The sources for this scenario can be very varied and are free to choose. Examples are:

- Location sensor: GPS-location, RFID;
- Light sensor;
- Humidity sensor;
- Temperature sensor;
- Web service.

A.3 Adaptive Tour Guide

Purpose: Deliver the right information at the right time during a visit of the area. This will allow people to discover the area at their own pace without the intervention of a human tour guide.

How: Determine which places of interest are advised to visit with regard to the current conditions. Recommendations may also depend on the current, past and future characteristics of the environment in combination with the preference of the user. Examples of such use could be the following:

- When the sun is shining and the temperature rises above 30, the adaptive tour guide could recommend the user to visit a water park;

- Emphasise or inform the user when there are special events planned at that moment.

Involved sources: The sensors for this scenario can be very varied and are free to choose. Examples are:

- Location sensor: GPS-location, RFID;
- Light sensor;
- Humidity sensor;
- Temperature sensor;
- Web service.

A.4 Smart Content

Purpose: Determine the access to a file based on user authentication. Additionally, we could filter the content towards the interests of a particular user.

How: Use an RFID-tag to identify the user. Filter the content of the document based on the user's social profile. Examples of such use could be the following:

- Close a file when the current user has no privilege;
- Show particular content of the document based on the user and add or remove specific content based on the user's social Facebook interests.

Involved sources: The sources for this scenario can be very varied and are free to choose. Examples are:

- GPS-location;
- RFID-reader in combination with RFID-tags.

Extra context: The following sources could assist the creation of the dynamic document by providing external information.

- Twitter;
- Facebook.

A.5 Finger Reader

Purpose: Help children read their first sentences. When children start reading, they use their fingers to *follow* a sentence. Since mobile phones and tablets are employed with touch-screens, it is no longer possible to merrily touch the screen without having a corresponding action performed. This means that, in most cases, a touch on a touch screen will result in an action. By making use of a slider or a joystick, we enable children to have a tactile interface to interact with the document and to "follow" the words in the sentences without triggering a certain action on the reading device.

How: By moving the slider we change the focus of the currently read word. Examples of such use could be the following:

- When the slider is moved to the left direction, it highlights the word on the left of the current position.

Involved sources: For now we can assume that the following two sensors/actuators are most suitable for the task.

- Slider;
- Joystick.

A.6 Fluid Reading

Purpose: In order to provide an interface that conveys information from any position in the room, we may want to adapt the presentation of the document.

How: By using the proximity sensor we are able to detect whenever somebody holds their phone up to his/her ear. This information can be used to determine when to switch between different output modalities. The modality that may result in the best user interface experience is chosen to present the information to the user. Examples of such use could be the following:

- When reading a document in your car it is advised that you keep your eyes on the road. Furthermore, as a corresponding action, the output modality could change from a visual presentation to an auditory presentation.

Involved sources: The sensors for this scenario can be very varied and are free to choose. Examples are:

- Proximity Sensor;
- GPS-location.

A.7 Fluid Font

Purpose: Maintain readability by automatically increasing/decreasing the font size of the document based on the proximity of the user.

How: Use a proximity sensor to detect the presence of people. Based on the proximity adjust the font size. Examples of such use could be the following:

- The further you are away from the screen, the bigger the font-size becomes;
- Close the document when no user is nearby.

Involved sources: The sensors for this scenario can be very varied and are free to choose. Examples are:

- Proximity Sensor;
- GPS-location;
- Light sensor.

A.8 Interactive Vocabulary

Purpose: Enable children to educate themselves by dynamically adapting the vocabulary to their needs.

How: Use a light-sensor to show the translation of a certain word. Depending on the amount of light that was sensed, provide a hint of the translated word. Examples of such use could be the following:

- Suppose a child wants to refine his knowledge of French. When the reading device is covered with the hand, the light intensity will be low. When there is a low light intensity, no hints are shown. Whenever the light intensity crosses a certain threshold, it will start showing some hints. Hints can for example be letters that appear according to the amount of incoming light;

- If the device is turned to the left or to the right, the letters of the translation may be shuffled to give a hint to the child.

Involved sources: The sensors for this scenario can be very varied and are free to choose. Examples are:

- Light sensor;
- Accelerometer.

A.9 Live catalogue

Purpose: Keep customers up to date by providing them with an accurate catalogue that always shows the current prices and the right amount of stock articles. Since the data is always up to date, customers are able to make a correct judgment of their purchases.

How: In order to keep an inventory of the store, storeowners could for example use RFID tags to keep everything organised and link this information with the database of the store. Afterwards, the catalogue could connect to the corresponding database in order to obtain real time information. Examples of such use could be the following:

- A magazine that will always present the right amount of stock articles.

Involved sources: The sensors for this scenario can be very varied and are free to choose. We think that the following sensor could be useful to enable the creation of such dynamic document:

- RFID sensor

A.10 Live Travel Guide

Purpose: Keep a travel guide up-to-date by providing users with the most popular destinations and weather information.

How: Retrieve the latest top 10 destinations by accessing web-services from travel and embed them in the travel guide. Furthermore the document could present real life weather information for a certain destination by making use of a web service and a sensor that provides the location.

Involved sources: The sensors for this scenario can be very varied and are free to choose, examples are:

- GPS-location;
- Web service.

External Source: The document could make use of external data sources to obtain useful information. Some of these external sources could be the following:

- Webservice;
- RSS feed.