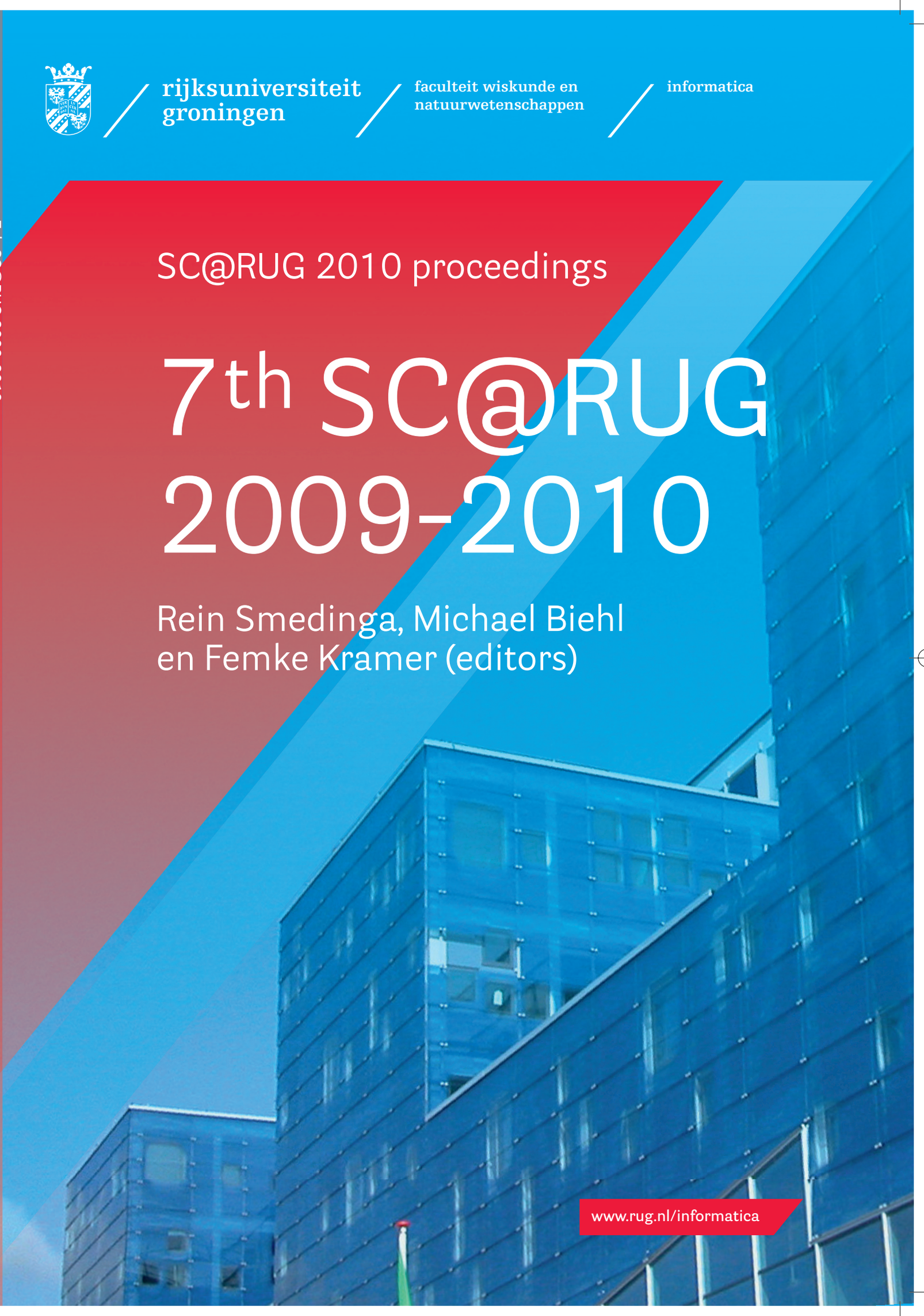
7th SC@RUG 2010 proceedings

Smedinga, Rein; Biehl, Michael; Kramer, Femke


rijksuniversiteit
groningen
faculteit wiskunde en
natuurwetenschappen
informatica
SC@RUG 2010 proceedings
7th SC@RUG
2009-2010
Rein Smedinga, Michael Biehl
en Femke Kramer (editors)
www.rug.nl/informatica

SC@RUG 2010 proceedings

Rein Smedinga
Michael Biehl
Femke Kramer
editors

2010
Groningen

ISBN 978-90-367-4462-1
Publisher: Bibliotheek der R.U.
Title: Proceedings 6th Student Colloquium 2009-2010
Computing Science, University of Groningen
NUR-code: 980

Contents

About SC@RUG

Introduction SC@RUG (or student colloquium in full) is a course that master students in computing science follow in the first year of their master study at the University of Groningen.

In the academic year 2009-2010 SC@RUG was organized as a conference for the seventh time. Students wrote a paper, participated in the review process, gave a presentation and were session chair during the conference.

The organizers Rein Smedinga, Michael Biehl and Femke Kramer would like to thank all colleagues, who cooperated in this SC@RUG by collecting sets of papers to be used by the students and by being an expert reviewer during the review process. They also would like to thank Janneke Geertsema for her workshops on presentation techniques and speech therapy.

In these proceedings all accepted papers are published.

Organizational matters SC@RUG 2010 was organized as follows. Students were expected to work in teams of two. The student teams could choose between different sets of papers, that were made available through *Nestor*, the digital learning environment of the university. Each set of papers consisted of about three papers about the same subject (within Computing Science). Some sets of papers contained conflicting opinions. Students were instructed to write a survey paper about this subject including the different approaches in the given papers. The paper should compare the theory in each of the papers in the set and include own conclusions about the subject.
Two teams proposed their own subject.

After submission their papers, each student was assigned one paper to review using a standard review form. The staff member who had provided the set of papers was also asked to fill in such a form. Thus, each paper was reviewed three times (twice by peer reviewers and once by the expert reviewer). Each review form was made available to the authors of the paper through *Nestor*.

All papers could be rewritten and resubmitted, independent of the conclusions from the review. After resubmission each reviewer was asked to re-review the same paper and to conclude whether the paper had improved. Rereviewers could accept or reject a paper. All accepted papers can be found in these proceedings.

All students were asked to present their paper at the conference and act as a chair and discussion leader during one of the other presentations. Half of the participants were asked to organize the of the conference day (i.e., to make the time tables, invite people etc.) The audience graded both the presentation and the chairing and leading the discussion.

Femke Kramer gave an introductory lecture about what is a scientific conference, and about general aspects of presentation techniques to help the students with their presentation and one on reviewing scientific papers. Michael Biehl taught a workshop on writing a scientific paper and Janneke Geertsema gave workshops on presentation techniques and speech therapy that was very well appreciated by the participants.

Students were graded both on all three aspects: the writing process, the review process and the presentation. Writing and rewriting counted for 50% (here we used the grades given by the reviewers and the re-reviewers), the review process itself for 15% and the presentation for 35% (including 5% for the grading of being a chair or discussion leader during the conference). For the grading of the presentations we used the judgements from the audience and calculated the average of these.

In this edition of SC@RUG students were videotaped during their presentation. The recordings were published on Nestor for self reflection.

On 23 April 2010, the actual conference took place. Each paper was presented by both authors. That day, we had eight presentations, each consisting of a total of 20 minutes for the presentation and 10 minutes for discussion. As mentioned before, each presenter also had to act as a chair and discussion leader for another presentation during that day. The audience was asked to fill in a questionnaire and grade the presentations, the chairing and leading the discussion. Participants not selected as chair were asked to organize the day.

This time, the conference was sponsored by Shell and Dr. R.F. Meiburg from Shell was the keynote speaker. He gave a talk with the title "Pushing the drill bit through petabytes, searching for Oil and Gas in the digital era."

All but one submitted papers were accepted for this proceedings.

Thanks We could not have achieved the ambitious goal of this course without the invaluable help of the following expert reviewers: Marco Aeillo, Ahmed Kamal, Tobias Isenberg, Michael Biehl, Nikolai Petkov, Heerko Groefsema, Alexander Lazovik and Paris Avgeriou.

Also, the organizers would like to thank the *School of Computing and Cognition* for making it possible to publish these proceedings and Shell for sponsoring the conference.

Rein Smedinga
Michael Biehl
Femke Kramer

Smart House: perspectives of XXI century information technologies

Elena Lazovik Josip Marić

Abstract—In this paper we present a new point of view on very innovative and modern idea of Smart House. We provide an overview and classification of the technologies used in realizing this idea: user interfaces, sensor technology, device interfaces and middleware. Finally, we illustrate the examples for every classification issue and Smart House in general.

Index Terms—Smart House, web services, user interfaces, Wireless networks, modern technologies.

1 Introduction

Although the idea of smart buildings is commonly presented by the hobbyists in the early 1960's, the term of the Smart House together with Smart Office is formally introduced by the researchers of PARC laboratories in the '80s[15]. The meaning of the term Smart House "is not in how well it is built, nor how efficiently it uses the space; nor because it is environmentally friendly, using solar power and recycling waste water;"[12] but in the way it is involved in daily activities involving the people living there. Besides of that general idea, we use the term of Smart House in a form of presenting ubiquitous (pervasive) computer technologies. The focus of our research lays in providing a high-level classification of the technologies currently in use, their historical background and their future perspectives.

The remainder of the article is organized as follows. Firstly, in Section 2 we propose the classification of the main classes of problems that should be solved to achieve the final goal of constructing a Smart Home. After that, in Sections 3, 4, 5 and 6 we provide a description of technologies available on the market that could be apllied to solve the existing Smart House problems and discuss the perspectives of available technologies. Finally, in Section 7 a discussion on open research issues is proposed, and Section 8 is dedicated to the main results of conducted research.

2 High-level classification

Weiser from PARC laboratories in his interview[15] identified main directions of further development of technologies concerning smart buildings. After that the attention was paid to the particularities of concrete technologies. Nobody has provided a clear classification for the main problems to solve in this area. In this article we propose such high-level classification and explain why every of the proposed high-level problems are important in area of Smart Houses.

The classification of the technologies that we propose consists of four main groups:

user interfaces. The problem is that all full-featured user-directed technologies are adapted to a personal computer or to a laptop, in the best case - to a mobile phone. The main issue for the user interface is its adaptation to the continuously changing environment of the house. The technologies capable to solve this issue are discussed further in Section 3;

Sensor technologies are being used for localization of the users and the very context in the system environment. That refers on defining the position of the devices and appliances. Besides that, sensors technologies are used to construct a map of the environment based on the data collected about the user(s) and context. Based on that localization system is able to define algorithms and actions needed to be taken. The available technologies in this area are discussed in Section 4;

devices interfaces. Every device has its own interface for interaction with environment. The problem is that the whole environment is heterogeneous and devices do not have a standard interface and sometimes the interaction is not possible. Possible ways of presenting the functionality of devices are described in Section 5;

integrating middleware. In order to establish communication and integration of heterogeneous devices in the general system the special middleware is needed. Such middleware should be responsible for integration of devices, permanent control of the house state and interaction of devices with people who live in this house. The examples of the middleware under development are presented in Section 6.

The criteria of dividing the proposed problems into four classes are that every class addresses a wide range of technologies and solutions concerning the same area of research which does not intersect with other classes. Together, these classes cover all technical issues that should be solved in order to achieve the goal of living in a new, really smart environment in the houses.

3 User interfaces

User interfaces in computer technology have been present from very beginning (Batch interface - 1948, command line user interface - 1961, graphical user interface GUI - 1981)[16].

They offer functionality of the system in the form of service. Such service pervasivity is particularly evident in examples where systems needs to continuously interact with human users. Service of the equipment that is embedded in systems like Smart House is the question we focus in this chapter. Interaction between human and systems offering kind of services is no more being static, but have to be ready to answer to the challenges like:

adaptability;

acceptability;

usability;

consistency;

cost.

User interfaces must go beyond 'friendly' and be attractive to users. People have to want to use the system. In this chapter we present user interfaces (UI) currently available and suitable for Smart Houses. We give short overview of all full-featured user technologies that we can find adapted to different kind of informations technologies (IT) devices. Particularly the ones used and easily found on personal computers (PCs), laptops or even mobile phones.

Adaptive user interface (also known as AUI) is a UI which adapts, that is changes, its layout and elements to the needs of the user or context and is, as the word adaptive implies, alterable to the same degree

Elena Lazovik is a computing science student on Master course at the University of Groningen, E-mail: E.Lazovik@student.rug.nl.
Josip Marić is a computing science student on Master course at the University of Groningen, E-mail: J.Maric@student.rug.nl.

by the user themselves[9]. These mutually reciprocal qualities of both adapting and being adaptable are, in a true AUI, natural (innate) elements that comprise the interface's components. Hence, these portions of the interface might adapt to and affect also other portions of the interface.

A **context sensitive user interface** is one which can automatically choose from a multiplicity of options based on the current or previous state(s) of the program operation. Context sensitivity is almost ubiquitous in current graphical user interfaces, and should, when operating correctly, be practically transparent to the user. The primary reason for introducing context sensitivity is to simplify the user interface. Advantages include:

> Reduced number of commands required to be known to the user for a given level of productivity.
>
> Reduced number of clicks or keystrokes required to carry out a given operation.
>
> Allows consistent behavior to be pre-programmed or altered by the user.
>
> Reduces the number of options to be on screen at one time (i.e. "clutter").

Disadvantage of context sensitive actions may be leaving the operator at a loss as to what to do when the computer decides to perform an unwanted action.

Remote control interfaces (RCI) are quite common in nowadays IT devices. Controlling the TV, pool, volume of the stereo, ACs and many others, have shown emerging possibilities of RCI. Remote controls for these devices are usually small wireless handheld objects with an array of buttons for adjusting various settings such as television channel, track number, and volume. In fact, for the majority of modern devices with this kind of control, the remote contains all the function controls while the controlled device itself only has a handful of essential primary controls.

Beside these mentioned, we present the new type of user interface, direct neural interfaces - **brain computer interface** (BCI). Its main idea lies in the direct communication between a brain and external device[17]. First researchs in this field have been done in 1970s on University of California Los Angeles (UCLA) and have been rapidly expanding since then.

BCIs were first imagined in a way of using artificial devices to replace the function of impaired nervous systems or sensory organs[17]. That means using prosthetics to replace damaged parts of human body (sight, hearing, movement, ability to communicate, and even cognitive function) and controlling them in direct connection between brain (nervous system) with computer system. "BCI" usually designates a narrower class of systems which interface with the central nervous system. Also, lot of research has been done in developing BCIs and algorithms that decode neuron signals. That allowed development of BCIs showing biggest impediment of this kind of technology at present - the lack of a sensor modality that provides safe, accurate, and robust access to brain signals. It is conceivable or even likely that such a sensor will be developed within the next twenty years. The use of such a sensor should greatly expand the range of communication functions that can be provided using a BCI. This is the area that can be focused for improving the Smart Houses by implementing BCIs sensors that would easily allow users to adopt to continuously changing environment of the house. The test case is shown at the Figure 1.

Of course, development and implementation of a BrainComputer Interface (BCI) system is complex and time consuming, but that does not stop us thinking about the emerging new possibilities of 21st century technology.

To briefly summarize, user interfaces have advantages that are malleable to variant user interface paradigms, logical orderings user preferences and their surroundings allowing them to be tailored almost perfectly to "the task at hand".

But flexible interfaces require additional facilities for the servicing of applications as buttons and menu items are not only moved about a plane on the screen yet are also moved through, that is up and down, logical action orderings and hence may not be where they were when the interface was shipped[10].

Though for some this indicates additional opportunities in employment and further innovation others consider it a cost.

4 SENSOR TECHNOLOGIES

Sensor technologies are present and can be found almost everywhere around us. Their presence in everyday life compared before 20 years has grown and will grow even more. Their perspectives are numerous. We are using sensor technologies for detecting movements of the people in areas of special importance (security cameras). Also, using sensors of smoke for detecting fire is useful in protecting house and its appliances not to get damaged. Then, of course, sensors for temperature used in regulating temperature and heating of the objects. Not to mention different kind of specialized sensors for pressure, force, navigation systems, radars, ionizing radiation, subatomic particles, etc. Everyday technology changes and improvements are made to the IT gadgets, like so sensors as well. That reflects on new possibilities of sensor technology within computer science. We focus on presenting the news concerning this field of IT, essential and highly needed for Smart Houses. What is the location of the user? What is the actual location of the device in the house? How to define whether there is more than one person in the house? How to determine who is person A and who is person B? These are the issues in which sensors are especially handy. With combining them in the system we can use the sensors for localization of the person placement inside the house and to register the position of devices in the house, constructing a map of the house, and regulating the house conditions essential for human life.

WiFi triangulation is used for detecting persons location in the house - geolocation. Geolocation is the identification of the real-world geographic location of an Internet connected computer, mobile device, website visitor or other. IP address geolocation data can include information such as country, region, latitude, longitude and timezone. Geolocation can be performed by associating a geographic location with the Internet Protocol (IP) address, MAC address, RFID, hardware embedded article/production number, embedded software number (such as UUID, Exif/IPTC/XMP or modern steganography), invoice, Wi-Fi connection location, or device GPS coordinates, or other, perhaps self-disclosed information.

Infrared (IR) sensor technology is quite common technology used a lot with combination with different kind of IT devices (gadgets like mobile phones). Infrared is especially useful for tracking, also known as infrared homing. It refers to a passive missile guidance system which uses the emission from a target of electromagnetic radiation in the infrared part of the spectrum to track it. Missiles which use infrared seeking are often referred to as "heat-seekers", since infrared (IR) is just below the visible spectrum of light in frequency and is radiated strongly by hot bodies. Many objects such as people, vehicle engines and aircraft generate and retain heat, and as such, are especially visible in the infra-red wavelengths of light compared to objects in the background.

IR data transmission is also employed in short-range communication among computer peripherals and personal digital assistants. These devices usually conform to standards published by IrDA, the Infrared Data Association. Remote controls and IrDA devices use infrared light-emitting diodes (LEDs) to emit infrared radiation which is focused by a plastic lens into a narrow beam. The beam is modulated to encode the data. The receiver uses a silicon photodiode to convert the infrared radiation to an electric current. It responds only to the rapidly pulsing signal created by the transmitter, and filters out slowly changing infrared radiation from ambient light. IR communications are useful for indoor use in areas of high population density. IR does not penetrate walls and so does not interfere with other devices in adjoining rooms. Infrared is the most common way for remote controls to command appliances.

Camera used as a sensor is new technology that we can find in, for example, digital photo cameras that use sensors to detect persons smile by actually scanning the area of face detection and then to denote the

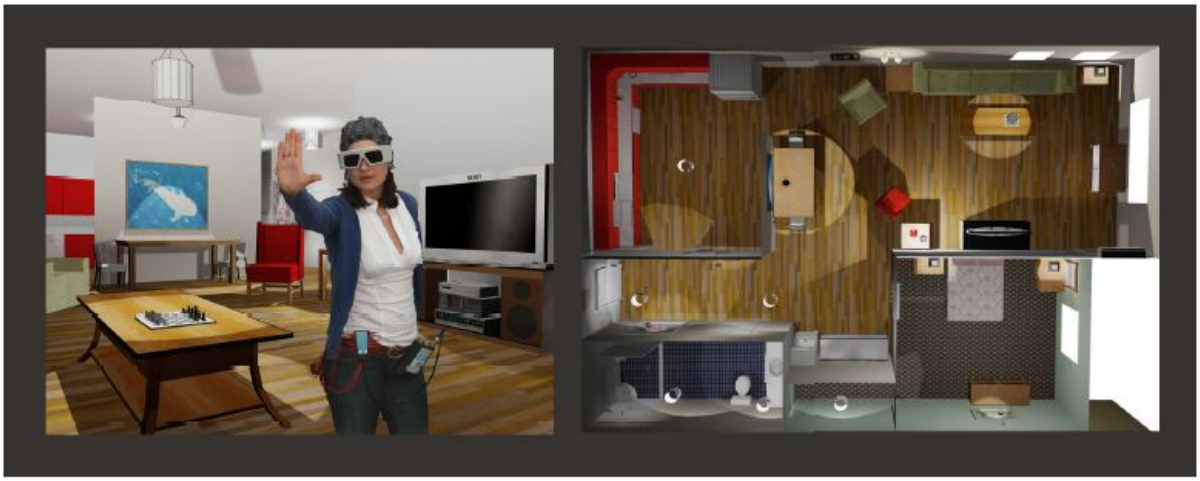


Fig. 1. Brain-computer interface in Smart House environment.[9]

movements of the persons face gestures. This is principle copied from the robotics where cameras are used to do the same but with the idea of copying the mode of the person to the object, as robots in this case.

A **video sensor** (also video-sensor or videosensors) describes a technique of digital image analysis. A video sensor is application software, which interprets images . Video sensors use programmable algorithms running on a computer. Video sensors are used to evaluate scenes recorded by a video camera. Objects and their characteristics (size and speed for example) are verified and compared to the pre-set examples or templates. When there is a match between object and model, then the frame and the objects are marked digitally. The operator can recall the digital marked images for further use. Video sensors are mostly deployed with video surveillance (CCTV) systems. The commercial use of video sensors is increasing. Two main applications are electronic security and market research.[18]

Lasers are new with the idea of usage in SH and are found to be very pervasive and innovative. The main idea stand from the fact that lasers are possible with their wide beams to act as radars and use that in SH for checking all of the possible things found in the example of the house to apply to work with the system. They perform by checking the coordinates of an object that they detect in 3 different ranges (beam spectrum) of coverage. Point to Point Laser Technology (PPLT) refers to a technology that enables a user or 'surveyor' to survey or capture a building's geometry in real time or while on site by translating laser range finder data directly into a [CAD] or [BIM] work station.

5 Device interfaces

Every device has its own interface for interaction with environment. The problem is that the whole environment is heterogeneous in many cases and devices do not have a standard unique interface and sometimes the interaction is not possible using different ways of communication. This problem appeared from the beginning of developing computer application. Variety of different programming languages, platforms, and even hardware has lead to increasing the difficulty of interoperation between different applications and devices.

One solution that was used initially is to develop a special intermediary between different devices that can get the messages from different devices, process them and transmit messages to other devices in format that is suitable for recipients.

Another solution that was also in use is to develop a special interface layer in devices that is responsible for converting the messages to the format required by the recipient applications and than sending the messages to the recipients.

The problem with such solutions is that it is possible to develop intermediary or an extra-level for the application within device only knowing which devices it will contact in future. Without the knowledge of the details of interfaces of other devices at home it is not possible to elaborate such networking protocol. In this case the interaction with, for example, newly arrived device and the network of devices built before using implementation of special intermediary for home or an extra layer in devices are not able to connect with each other.

Smart home needs an interaction between the devices in conditions of ever-changing environment where the devices could be easily added and removed on-the-fly.

These interactions are not possible to realize unless standardized protocols are developed. Protocols are often presented as stacks of related protocols taking care of different aspects, in the spirit of the famous ISO/OSI Networking stack reference model. To make an interoperation possible for heterogeneous application in Internet the most popular solution is to use Web and its protocols based on HTTP.

The emerging success of service-oriented computing in Web brings back services from Internet to real life. Everyday devices, such as mobile phone and media players, and even fridges and TV become smarter and smarter, and often provide their functionalities in form of embedded services. These services can be accessed through standardized API, e.g., web services.

Nowadays, the web services architecture is widely used as an architecture for creating different distributed computer systems.

There is a number of protocol stacks that eliminate the problem of interoperation between heterogeneous devices. The most famous ones are REST [11] and web services [13, 7]. These protocol stacks are usually formed by several layers, starting from low-level transport protocols, e.g., SOAP [1], up to complex composition [8] and transaction [19] languages.

The most popular scheme for the implementation and presenting web services on Internet is using SOAP and WSDL standards together.

SOAP(Simple Object Access Protocol), is a protocol specification for exchanging structured information by messages between web services in distributed computer networks. It relies on Extensible Markup Language(XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission[6]. SOAP provides a basic messaging framework upon which web services can be built.

WSDL (Web Services Description Language) is an XML-based language describing web services[2]. WSDL allows to separate the description of the functionality of web service from its concrete implementation, used middleware and underlying operating system. WSDL describes web service at two levels: abstract and concrete. At abstract level it describes the service in concepts of messages that service sends

and receives. At concrete level WSDL specifies binding of description with implementation of service in terms of transport and wire format.

The pair SOAP/WSDL becomes a popular format nowadays for use inside smart environments.

Term REST(REpresentation State Transfer) has been proposed by Roy Fielding and recently RESTful services gain more and more popularity within web community. The REST Web is the subset of the WWW (based on HTTP) in which agents provide uniform interface semantics – essentially create, retrieve, update and delete – rather than arbitrary or application-specific interfaces, and manipulate resources only by the exchange of representations[3].

The perspectives of both RESTful and "Big", i.e. SOAP/WSDL, web services are very good as they provide not only the division between business logic and representation of data, but also the transparent standard communication protocol. That solves the problem of interoperation between devices within Smart House changing environment.

6 Integrating middleware

When the devices are connected into the home network, every device can have an association to another device. That means that device can invoke the functions or require the necessary information from other devices. The solution is to allow devices to send requests for obtaining necessary functionality from others. However, often it is not possible because of the heterogeneity of devices. Only by solving the problem of integration of the devices within the network it is possible to obtain an environment, free from heterogeneity problems. The devices inside such network can exchange an information, but for controlling the whole house and to interact with persons living in that house a special middleware is needed. Such middleware is responsible for receiving the data from devices, parsing it and deciding which actions are needed to be performed on-the-fly. with middleware the interaction of user with the environment of Smart House is facilitated and security and privacy are increased. The developing of such kind of middleware begins to be possible only at present time. All necessary enabling technologies were needed for permitting at least a thought about the special middleware. Therefore, there is no much history for the middleware for Smart Houses, but there are some research projects that consider the constructing of the special middleware for intercommunication between devices, for user communication with the system and for the permanent control of the house environment. We provide here an overview of two research projects aimed on constructing a middleware for Smart Homes: SM4ALL[14] and Hydra[4]. These projects are not unique in considering middleware as a solution for changing environment of Smart Home. The choice is based on some characteristics that the projects have. SM4ALL project is chosen because of the using web services, its dynamic composition on-the-fly and possibility of performance of the complex scenarios. Hydra project is chosen because it proved the possibility of custom configurations and remote management of networked sensors as a basis for Smart Home changing environment. Moreover, both projects adress wide range of users from the disable people to children and adults.

6.1 SM4ALL project

One of the projects taking into account a middleware as the system controlling the house is SM4ALL european project[14]. The novelty of the SM4ALL project is that it is aimed on providing a dynamic web service composition and use of the non-traditional user interfaces, like brain-computer interface, together with the traditional ones.

Goal of the SM4ALL architecture is to seamlessly integrate devices in order to simplify access to services provided by these devices and dynamically compose these services to offer to the end users more complex functionalities and a richer experience of the domotic environment. The general architecture of the SM4ALL middleware is presented in the Figure 2.

Due to the different technologies employed by the devices that are expected to interact within SM4ALL, the architecture relies on an abstracting communication layer represented by the UPnP standard.

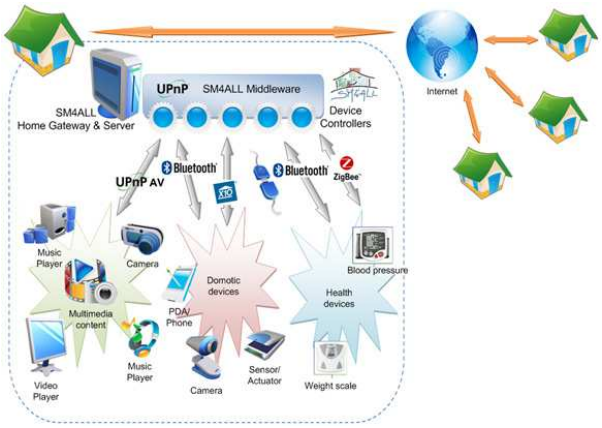


Fig. 2. SM4ALL middleware architecture.[14]

The SM4ALL system is constituted by a set of logical components arranged in three distinct layers:

User layer - This layer is devoted to the interaction with final users and administrators. This interaction will be realized through various UIs.

Composition layer - This layer has the main goal of receiving high level commands issued by users through the interface layer and fulfilling the corresponding complex goals by controlling the execution of lower level services offered by devices deployed within the SM4ALL architecture.

Pervasive Layer - It represents the physical layer of the SM4ALL architecture and the software components needed to abstract it.

The modules of the SM4ALL architecture belonging to different layeers and the interactions between the modules are presented in the Figure 3.

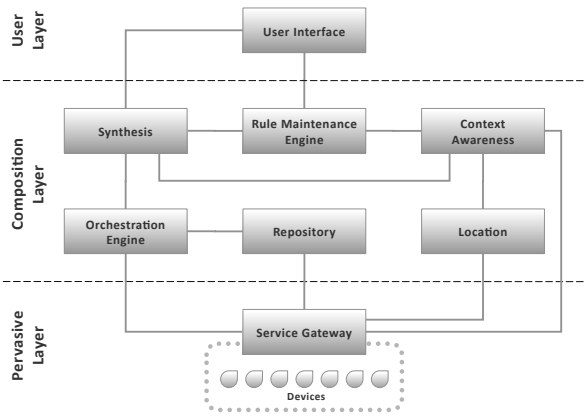


Fig. 3. SM4ALL middleware architecture.[14]

User layer is the interaction layer. It is dedicated to the interaction of the system with different types of users. Due to the different communication technologies it is realized through different User Interfaces. From the point of global view one can see this layer as one logical component: User Interface. It is the component built to get user input which will become the service invocation at the level of lower layers. Such interfaces will be used by users to establish new goal to achieve, to set their individual preferences or to put preferable

rules for the automatic monitoring and control of the home on the permanent basis.

The main goal of the **composition layer** of the middleware is to receive user input on the high level language from user interface, to fulfill goals to achieve and to control the execution of the services provided by smart home devices deployed within the middleware platform.

Following logical elements constitute the composition layer:

Repository. It is a general repository for descriptors of services, different ontologies and other types of data;

Synthesis. This logical component receives user input from the upper User layer or from the Rule Maintenance Engine and composes concrete Plans. Synthesis is purposed to translate a high-level complex goal into the sequence of more simple actions that can be assigned to different devices having corresponding web service functionalities. The translation of the high-level goal is conducted according to the information from the Context Awareness logical component;

Orchestration Engine. This engine receives the Plan from the Synthesis logical component and constructs the set of the web services available from devices deployed within the middleware platform. Thus, Plan can be executed with actual services. The orchestration is executed while interacting with the Repository data containing web services descriptors.

Rule Maintenance Engine. This engine is constructed in order to maintain automatic actions of the system inside smart home environment. It activates some functionalities when special determined conditions are hold. It means that Rules are activated depending on the Conditions. The Conditions are inserted through User layer as Rule Preferences for smart home. The Plans constructing depends on the Context Awareness and Location logical components.

Context Awareness. This component collects data from devices, processes and stores the resulted values in order to provide up-to-date information about real environment and current status of the system. These results represent the context for all layers of the middleware. User preferences and Rule preferences are the parts of the context too.

Location. This component is constructed to contain an important information for the whole context. It contains locations of the objects and people inside the smart home environment where middleware is running. Specific logical component was elaborated for such kind of information because complex mechanisms are involved in calculating the locations of people and objects within the home.

Logical components Context Awareness and Location are permanently interacting with Pervasive layer in order to receive up-to-date data from physical devices with embedded web services systems.

To deal with complex scenarios within smart home a planning system is a necessary element of the Composition layer. It synthesizes plans on-the-fly based on goals given by home inhabitants.

The goal is specified through one of the possible interfaces, be it brain-computing interface, mobile device, voice-recognition, or, possibly, other software. Given a goal, the planner collects the information about the current state of the house, e.g., available services and their current states, through the context module, which may possibly pre-fetch the data for better performance.

Synthesized plan is then given to orchestration component which is responsible for a plan execution. It also includes simple reasoning capabilities for simple failure recovery and service instantiation. To execute a particular action, the orchestration component finds one of the possible services that implements the desired action. Some extra constraints may be associated in this case to reduce possible instantiations. For example, alarm action may instantiate corresponding implementation which is close to the user, e.g., by showing a message on TV screen if the user is watching it, or by invoking alarm in the alarm clock if the user is sleeping in his bed.

User himself is represented as one of the services at the pervasive layer. That is, whenever interaction with the user is needed according to a plan, orchestration component simply invokes one of the services that represent the user.

Pervasive layer of the middleware is a physical layer of the system which is represented by single logical component: Service Gateway.

Service Gateway is a component which lets physical devices to interact with the other layers components by presenting the descriptors of the embedded web services, executing service instances and working as a specific middleware between user communication devices and the platform, etc. The component can be viewed as an abstract wrapper for all devices which are presented in house.

The middleware is aimed to provide a comfortable and safe environment for different types of people: from young to old, from healthy to disabled.

6.2 Hydra project

The Hydra project[4] is a 4-year Integrated Project that constructs the middleware for Networked Embedded Systems. The Hydra project is co-funded by the European Commission.

The Hydra middleware allows developers to incorporate heterogeneous physical devices into their applications by offering easy-to-use web service interfaces for controlling any type of physical device irrespective of its network technology such as Bluetooth, RF, ZigBee, RFID, WiFi, etc. Hydra incorporates means for Device and Service Discovery, Semantic Model Driven Architecture, peer-to-peer communication, and Diagnostics. Hydra enabled devices and services can be secure and trustworthy through distributed security and social trust components of the middleware.[4]

One of the subparts of the Hydra project is MORE project[5]. MORE is a Specific Targeted Research Project (STREP) that implements a new technology to facilitate communication and distributed intelligence across groups of users using different wireless standards. The project addresses the problem of how the interaction between humans and embedded systems can be efficiently supported by developing a system that can be tailored to the specific needs of diverse organizations.[5] A project is focused to construct a middleware that hides the heterogeneity complexity of embedded systems through providing simplified interfaces and management mechanisms for the future operators of these systems.

Fig. 4. MORE middleware architecture.[5]

The design of middleware at this project considers a client-server architecture with the central control point in network, as it can be seen in Figure 4. The main functionality of the middleware are:

Alleviate heterogeneity of devices;

Support scalable group communication;

Allow for multi-media communication and resource sharing between humans (voice, pictures, video) and machines (in particular sensors);

Ensure security of communications, data exchanges and protection of sensitive data;

Provide Gateway services, allowing access to embedded networks and increasing range of accessibility (e.g. connecting small scale local Bluetooth networks to large scale mobile networks like UMTS).[5]

To deal with heterogeneity of devices, the MORE functionality is presented to the developer as services. This specification of the services provides to the developer a wide range of different choices of services or their combinations. Each service should consist of several parts. First part is a functionality of service itself. Second part is a special connector for communication with other services and/or devices and users. Third part consists of implementations of the connectors and the service functionality itself. In Figure 1 the connectors are represented by green boxes at each side of the service circle and the white ring at the border represents the implementation and the inner part the functionality, in this context called the Service Logic. The most important service for every device in network is the Core Management Service (CMS) which provides common functionality needed for operation and management of the MORE middleware.

Each service should have a capability to send its functionality description to the Core Management Service. In description of service an information about communication protocol, memory, energy and hardware should be included.

To be able to reach the services on different devices, each service sends and receives messages through queues that are linked to connectors: Notifiers and Listeners. An implementation of at least one combination of a Listener and Notifier is compulsory for every service that communicates with another.

The famous pair of protocols SOAP/WSDL[6],[2] is used for communication between different services.

The Core Management Service is a main service running in one so called node (one smart house environment). CMS could start, pause, reschedule all other services belonging to the same node. It also provides support for service updates and for retrieving the information about service status and tasks by users. Different CMS can connect to each other through secure channels.

Such construction of network allows people to check remotely what happens at home and provides a good facility for interoperation between devices.

6.3 Summary

The most important difference between the SM4ALL and MORE architecture is that the complex scenarios are not possible to be executed on-the-fly in case of MORE. They should be programmed before the execution. SM4ALL middleware allows a wide range of complex scenarios due to its internal architecture.

Such types of middleware can be applied to any house of the present moment, though the support of SOAP/WSDL protocols is requested from the devices. The inclusion of support of that protocols to the devices seems to be possible now. Therefore, the perspectives of middleware are very good.

7 Open research challenges

Currently, many solutions are proposed for the size of networking devices, their capabilities of computation and their increased power. These technologies together with the protocols of communication for heterogeneous applications are enablers for constructing a person-friendly environment which controls a house and interacts with user. However, some issues are not solved and they are still under discussion.

One of the problems still under discussion is the management of concurrent requests of different users living in the Smart Home. For example, if both users want to listen the music, but they have different music tastes, the Smart Home system should decide whose request to deal with firstly. This problem could be solved using different concurrent programming techniques, however, it is not yet addressed by scientists.

Another problem that should be addressed in future is the personal and social consequences of living within smart environment which are largely being overlooked. Smart Home environment changes a personal experience of people living in the house significantly. However, the consequences of living in smart environment are not yet discussed in scientific world.

8 Conclusion

We provided classification based on the insight of Weiser[15] and on current development of technologies in area of Smart Houses. Our classification covers all main issues regarding the Smart Houses with all the instances in the environment where people live. The methodology of interaction between the system and human users is a crucial point of our classification.

The history of development of technologies shows that the solutions nowadays tend to be more abstract and much more reusable than before. The needs of disable people are also in the center of the attention and with the technologies that are currently under development it is possible to create a safe and comfortable atmosphere also for such type of people.

Adding the special middleware to the enablers allows to say that the concept of Smart House is not a distant future, but it is under development right now. However, there are some problems that are not solved yet, like concurrent process of the requests from different users or personal and ssocial consequences of living in smart environment. That means, that more work should be done to achieve the true Smart Home environment.

References

[1] Soap v1.2: Simple object access protocol. http://www.w3.org/TR/soap12-part1.

[2] WSDL Version 2.0 Part 1: Core Language. http://www.w3.org/TR/2003/WD-wsdl20-20031110/, 2003.

[3] REST web services: Web services architecture. `http://www.w3.org/TR/ws-arch/#wsdossoa`, 2004.

[4] Hydra project. `http://www.hydramiddleware.eu/news.php`, 2005.

[5] More project. `http://www.ist-more.org/`, 2006.

[6] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). http://www.w3.org/TR/soap12-part1/, 2007.

[7] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services. Concepts, Architectures and Applications*. Springer, 2004.

[8] BPEL. Business process execution language for web services. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[9] C. Guger. http://www.perada-magazine.eu/pdf/1741/1741.pdf.

[10] T. Catarci, F. Cincotti, M. de Leoni, M. Mecella, and G. Santucci. Smart homes for all: Collaborating services in a for-all architecture for domotics. In *CollaborateCom*, pages 56–69, 2008.

[11] R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002.

[12] R. Harper. *Inside the Smart Home: Ideas, Possibilities and Methods*. Springer London, 2003.

[13] M. Papazoglou. *Web Services: Principles and Technology*. Prentice Hall, 2008.

[14] SM4ALL. EU STREP Project FP7-224332 Smart Homes for All. www.sm4all-project.eu, 2008.

[15] M. Weiser. *The Computer for the XXI century*. Scientific American International Edition, 1991.

[16] Wikipedia. http://en.wikipedia.org/wiki/User_interface.

[17] Wikipedia. http://en.wikipedia.org/wiki/Brain-computer_interface.

[18] Wikipedia. http://en.wikipedia.org/wiki/Video_sensor_technology.

[19] WS-Transaction. http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html.

Documenting Software Architecture Designs The "4+1" View Model vs. Siemens Four Views

Sara Mahdavi Hezavehi, University of Groningen

Abstract—This article considers two of the well-known methods, the "4+1" view model and Siemens four views, used for documenting software architecture designs. These methods consist of multiple views each of which addresses a set of system's requirements and stakeholders' concerns independently. First, we briefly describe these methods by explaining their views, and then we consider if these approaches affect different system's functional and non-functional requirements; Afterward we try to find those stakeholders influenced by each of these methods to be able to find a proper documenting method. In fact, in this paper we intangibly suggest an approach for documentig software architecture designs; the main idea is to identify stakeholders' concerns, and system requirements to be able to select a more appropriate documetnig method for our software architectures based on these two factors.

Index Terms—View, Concerns, Stakeholders, Requirements.

1 Introduction

Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects. But it is not easy to prepare a document which contains all the needed information in a comprehensive way. There are so many software architecture documents available, trying to provide architectural aspects of a system by offering diagrams and blueprints which may not be conceivable all the times; however, without following a structured method, it would be confusing to realize how much one should go into details of the system while preparing a blueprint, and which aspects should be considered in one diagram, that is why different models have been developed for documenting software architecture designs. Each of these methods consists of different views to address variety of architectural aspects of the system, requirements and stakeholders' concerns [1], [5].

2 The "4+1" view model

The "4+1" view model is suitable for documenting large and complex architectures, it offers five views for describing a software architecture (fig. 2.1), and each of the views addresses a specific set of concerns and system requirements. For instance, the process architecture takes into account some non-functional requirements, such as performance and availability [1]. The "4+1" view is briefly explained in the following section.

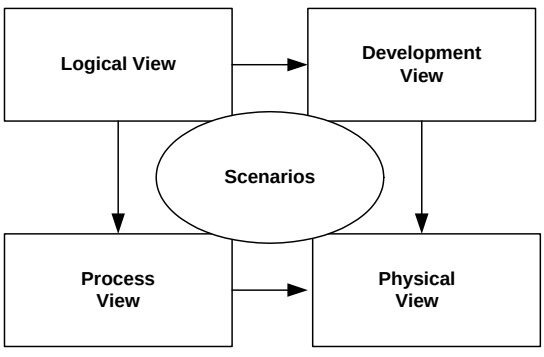


Fig. 2.1. The "4+1" view model [1].

2.1 The logical view

When an object-oriented design model is used, the logical view is the object model of the design and chiefly supports functional requirements. The system decomposed into a set of key abstractions, taken from the problem domain, in form of objects and object classes. They use the principles of abstraction, encapsulation and inheritance [1]. The purpose of this kind of decomposition is not only doing functional analysis, but also identifying common mechanisms and design components existing in different parts of the system. To present the view, class diagrams and class templates are being used.

2.2 The process view

As mentioned before, the process view is mainly concerned about non-functional requirements such as performance and availability, and also addresses issues of concurrency and distribution, of system's integrity, of fault-tolerance, and how the main abstractions from the logical view fit within the process architecture [1]. This view can be explained at several levels of abstraction, each of which addressing different concerns.

2.3 The physical view

The physical view describes the mapping(s) of the software onto the hardware and reflects its distributed aspect [1]. It is mainly concerned about non-functional requirements of a system such as availability, reliability, performance, and scalability.

2.4 The development view

The development architecture of a system describes the static organization of the software in its development environment, and is represented by module and subsystem diagrams. This view provides the basis needed for reasoning about software reuse, portability and security [1].

2.5 The "+1" view

The fifth view is actually putting all the views together by use of a set of important scenarios.

3 Siemens Four Views

The Siemens method uses four views to document the architecture design of a system (fig. 3.1). The first task for each view is global analysis. The purpose of the global analysis is to analyze the factors that affect the architecture and to develop strategies for designing the architecture. Global analysis begins before any of the views are defined, and it continues during the architecture design [2], [3].

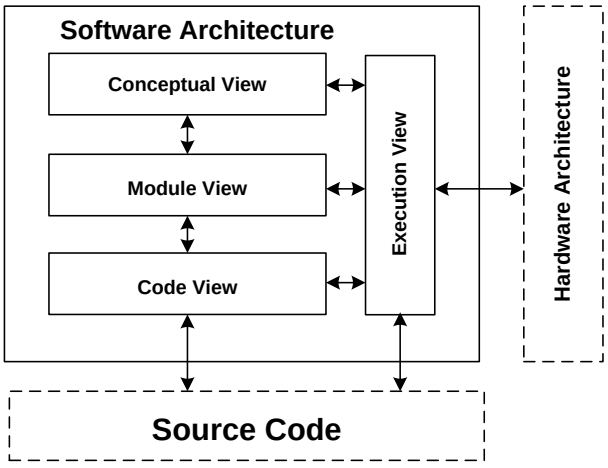


Fig. 3.1. The Siemens four views [2].

3.1 The Conceptual Architecture view

The conceptual view describes the system in terms of its major design elements and the relationships among them. There are three phases in this view: global analysis, central design tasks and final design task. The central design tasks include four coupled tasks which are used to identify components and connectors needed for building the system. In the final design task the results of the central design tasks is required to assign the resources to the components and connectors in the configuration. By finishing the conceptual view there would be the possibility to argue about the ability of the system to fulfil functional requirements of the system [2], [3].

3.2 The Module Architecture view

In the module view relationships among the implementing elements must be made explicit. For instance, how the system uses the underlying software platform. In the module view all the application functionality, control functionality and adaptation should be mapped to modules. Modules are organized into two structures: decomposition and layers. The first one captures the way the system decomposed into subsystems logically; a module is assigned to a layer and constrains its dependencies to other modules. This view also has three phases: global analysis, central design tasks, and interface design. The central design tasks consist of three coupled tasks: modules, layer, and global evaluation; the results of this phase are used by central design tasks of the execution and code architecture views [2].

3.3 The Execution Architecture view

The execution view explains the mapping of functionality to physical resources and runtime characteristics of the system. It is a set of models that describe and document what a software system does at runtime and how it does it. Since the mapping may change over the time, for instance while developing, or those changes which emanate from improvement of hardware and software, it is important to design the architecture in such a way that adapt to the alterations easily [2], [6].

3.4 The Code Architecture view

The main purpose of the code architecture view is to make the construction, integration, installation, and testing of the system easier with respect to other three views. In the global analysis phase those factors and strategies which affect the code architecture view should be identified. During the central design tasks phase all the components and their relationship to elements in the module and execution views should described in detail. In the final design task the decisions related to the build procedures and configuration management should be checked to see if they support those made during the central design tasks [2].

4 THE "4+1" VIEW MODEL VS. SIEMENS FOUR VIEWS

In this section we try to offer a comparison between these two methods with respect to two different aspects: fulfilment of requirements and meeting stakeholders concerns. Generally, the "4+1" view model supports a larger number of stakeholders' concerns than Siemens four views, but both meet a number of requirements. In the following sections a detailed comparison is presented.

4.1 Requirements fulfillment

Table (1) indicates a list of requirements and whether the methods touch these requirements or not (based on [1], [2], [4]). Plus symbol indicates that the method touches that requirement and those which are not explicitly touched, are left empty.

Table 1. Requirements touched by methods [1], [2], [4].

	Performance	Interoperability	Usability	Reliability	Portability	Security	Testability	Reusability	Availability	Scalability
The "4+1" view model	+	+		+	+	+	+	+	+	+
The Siemens four views	+	+		+	+		+	+		

The "4+1" method mostly supports non-functional requirements of the system such as availability, reliability (fault-tolerance), performance (throughput), and scalability by use of physical view; software reusability, portability and security by use of development view; integrity, performance, and availability by use of process view; and finally, understandability by use of scenarios. Among all the views of the "4+1" approach, only the logical view supports the functional requirements-what the system should provide in terms of services to its users-, it actually takes into account only the functional aspect of the requirements [1].

On the other hand, to predict some important system properties such as performance estimation, safety and reliability analysis, and effort estimation the Siemens four view model uses the Conceptual Architecture View; and for management of module interfaces, and change impact analysis it uses the Module Architecture view. The Code Architecture View is being used to achieve a transparent access to all the components needed for a particular development task, and managing versions and releases of the components. And finally, the Execution Architecture View helps to design the runtime aspects of the system, provide a correct implementation, do the testing job, and determine how a change in the runtime platform affects the system (based on the preceding explanations it actually can be used by architects, developers, testers, and maintainers, respectively) [2].

Note: Here by usability we mean usability of the final system and not the usability of the document for the end-user; obviously in the second case both methods support the usability.

4.2 Stakeholders' concerns fulfillment

In this section we consider which of the stakeholders and their needs are fulfilled with these methods, and how much one specific method cares about one stakeholder's needs.

Based on [2], the Siemens four views mostly take care of the architects and engineers requirements. This method is not concerned about the end users needs. On the other hand, besides aforementioned stakeholders, the "4+1" view model also takes care of the end users and their needs [1]. Table (2) indicates a list of those stakeholders who are explicitly touched by these methods according to [1], [2], [4] and shows if the methods meet their needs.

Table 2. Stakeholders affected by methods [2], [4].

	Architects	Developers	Testers	Maintainers	Managers	Integrators	End Users
The "4+1" view model	+	+	+	+	+	+	+
The Siemens four views	+	+	+	+			

5 CONCLUSION

Using the "4+1" view model or Siemens four views avoids from creating confusing documents, indeed, by offering multiple views, they address a set of requirements and concerns separately. But none of them covers all the functional and non-functional requirements, and also the stakeholders and their concerns. The "4+1" view model supports a big set of requirements and most of the stakeholders are being considered in this model. The Siemens four views also support some requirements, but it just takes care of a specific group of stakeholders and their needs, mainly architects and system engineers. However, one cannot claim which method's benefits overweigh the other one. To make the best decision and select an appropriate documenting method among aforementioned methods, one should first consider the system, requirements, stakeholders and their priority thoroughly, and then pick the appropriate model based on them. Also, keep in mind that there would always be a trade off while trying to choose the best method.

REFERENCES

[1] The "4+1" View Model of Software Architecture, Philippe Kruchten, Rational Software Corp, Paper published in IEEE Software 12 (6) November 1995, pp. 42-50.

[2] Applied Software Architecture, Hofmeister, Nord, Soni, 2006 ADDISON WESLEY.

[3] Documenting Software Architectures views and beyond, Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judy Stafford, 2001, 2002 ADDISON WESLEY.

[4] A Survey of Software Architecture Viewpoint Models, Nicholas May, 2005.

[5] Software Architecture: An Executive Overview, Paul C. Clements Linda M. Northrop,February 1996.

[6] Arias, T. B., America, P., Avgeriou, P., Science, C., & Research, P. (n.d.). Defining Execution Viewpoints for a Large and Complex Software-Intensive System.

Depth Cueing and Haloing for Molecular Visualization

Matthew van der Zwan Wouter Lueks

Abstract— In the field of illustrative visualization, different techniques have been developed to enhance depth perception. We compare three examples of such techniques: Depth-dependent halos which applies halos to dense line datasets, ambient occlusion and halos around atoms for molecular visualization, and flexible volumetric halos applicable to volume data. Based on this analysis, we develop a method to render abstractions of a protein in different visualization styles. Furthermore, we created a smooth structural abstraction functions that interpolates between these styles. The result is a tool for exploring the internal structure of proteins.

Index Terms—Illustrative visualization, NPR, depth cueing, molecular visualization, haloing.

1 Introduction

For decades, it has been the goal of computer graphics to produce realistic images. While photorealism is a laudable goal, the field of illustrative visualization has been gaining ground over the past decades. Instead of creating realistic visualizations, concepts from traditional illustration are used to create images that better explain the important and relevant parts of the data.

Illustrative visualization is applicable in many domains, the medical sciences, chemistry, and engineering being among the most important ones. We examine each in turn. Illustration in medicine has two aspects. First, it caters to professionals who want to explore the region around a tumor, for example, and second, it can be applied for education and explanation in textbooks and in communicating procedures to patients. The former requires a more accurate representation of the data than the latter. Among the many medical data sources are CT, MRI, and DTI, all providing measurements based on a volumetric grid. In chemistry and biology, visualization is often used to show the structure of molecules and proteins. Finally, visualization is used in engineering to illustrate the construction of machines and devices.

Because of the development in computer graphics away from photorealism the field of non-photorealistic rendering (NPR) emerged. The goal in NPR is to be inspired by traditional artistic styles and techniques [7, 15]. Among these are the use of hatching as an alternative for traditional grayscale shading. Lines can also be used to illustrate the shape of the object by following contours, suggestive contours, and apparent ridges [4, 8, 10]. These methods are often called low-level since they only deal with how objects are presented, not with what is presented.

Another concept of NPR is the use of abstractions, for instance to emphasize the general shape of an object. A side effect of most abstractions is that they distort the model which is undesirable in, for example, medical applications. High-level techniques primarily deal with "what to show" [18]. Some examples, also known from traditional medical and engineering approaches, are cutaway rendering and exploded rendering. In the first method, part of the object is made completely or almost transparent to reveal the structure inside. In medical applications one could, for example, remove the skin to reveal the muscles. In an exploded view, the different components of an object are disassembled and shown in such a way as to suggest the method of assembly.

In order to apply these high-level techniques a measure of relevance is needed [14]. Acquiring sufficient information to compile a suitable relevance function is not an easy task. While quite a lot of interactive techniques are available [14], they all use some segmenting of the data into classes.

We first focus our attention on comparing three low-level techniques that use illustrative approaches to enhance depth perception, thus yielding a better understanding of the 3D structure. In Section 3 we combine some of these techniques into an interactive high-level method for visualizing protein structures. The paper is concluded in Section 4 with a summary and a brief discussion of future work.

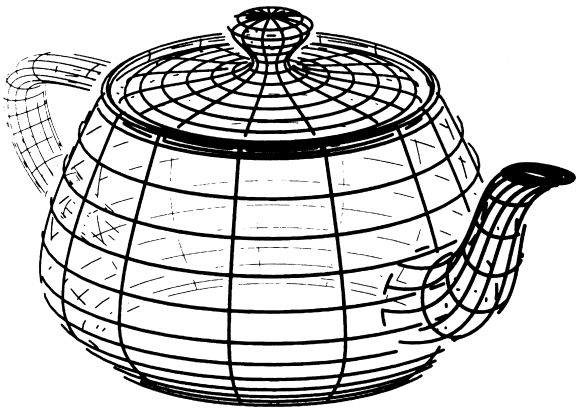


Fig. 1. Illustration of a teapot with the use of halos. Image from [5].

2 Depth cueing in Illustrative Visualization

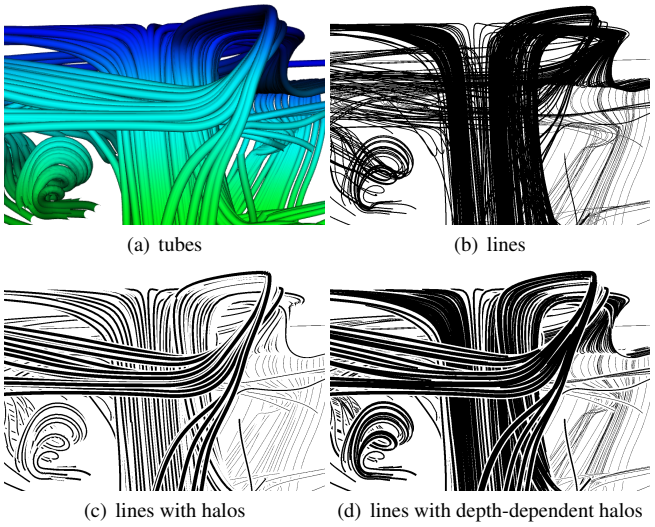
One of the important issues in illustrative visualization is how to enable the user to correctly interpret the 3D structure of an object. The method we study here is how to enhance depth perception, which is one way of clarifying the 3D structure. We will review the following three relevant techniques: enhancing edges and silhouettes, haloing, and ambient occlusion.

The first set of methods clarifies the structure using line techniques. In the introduction we already mentioned the drawing of silhouette edges in order to make the structure more apparent [10]. However, silhouettes alone do not really enhance depth perception, this only happens after adding appropriate depth cues: decreasing the line-width and/or opacity of the lines are common methods.

Another, but related, method to enhance depth perception are halos. A halo is a radiating border around an object, much like the bright silhouette you would see around a back-lit object. In 1979 Appel et al. [1] described a method of using this haloing technique for clarifying line drawings. Fig. 1, produced by Elber [5], uses both haloing and decreasing line-widths for depth cueing. Observe that the halos give an indication of depth.

Halos can also be used to enhance depth perception in solid models [3]. However, there are also other techniques to enhance depth perception for solid objects. Accurate shadows alert the viewer to the fact that some regions are occluded. To produce these, however, a global

• *The authors are students at the University of Groningen, The Netherlands, E-mail: {m.a.t.van.der.zwan|w.lueks}@student.rug.nl.*



(a) tubes (b) lines

(c) lines with halos (d) lines with depth-dependent halos

Fig. 2. Different techniques of rendering dense line data sets, taken from [6].

illumination model is required. Unfortunately, this is computationally expensive. A simpler approach is ambient occlusion. One can think of this as determining, for each point on the surface, which fraction of the hemisphere, corresponding to this point, is occluded by other objects [12]. Note that while this technique gives an approximation of the shadows, it does not take reflexion into account.

We demonstrate these principles by reviewing the following approaches: depth-dependent halos [6], enhanching molecular visualization [16], and flexible volumetric halos [3].

2.1 Depth-Dependent Halos

The technique of depth-dependent halos was developed to visualize dense line data sets. This line data can come from a number of sources, such as DTI, MRI, or fluid simulations. From such data, fiber tracts or stream lines are extracted. Most line data obtained in this way contain a lot of lines that are closely bundled in parts of the space, so the depth-dependent halo technique was designed to deal with these kind of situations.

Traditionally, tube rendering and plain line rendering are used to visualize these line data sets, see Fig. 2(a) and (b). However, tube rendering methods rely on shading to create depth, requiring the tubes to have a certain minimal width and thereby introducing a maximum on the amount of tubes that can be visualized, see Fig. 2(a). Plain line rendering allows us to visualize more data than the tube method, but this may lead to large areas of black in our picture where there is no depth perception, Fig. 2(b). A solution is the use of halos , Fig. 2(c). While this technique succeeds in providing depth information, it does not show clustered data as well as the plain line rendering does. Therefore, a new technique was created that is capable of enhancing depth percertion while still maintaining clustering in line data. This technique is called depth-dependent halos, Fig. 2(d), created by Everts et al. [6].

To visualize the lines, view-aligned triangle strips are generated. This is realized by duplicating the line data points after which they are moved away from each other to form a triangle strip. This triangle strip not only contains the black line that has to be drawn, but also a white stroke on both sides depicting the halo. The difference with the per-line haloing technique is that the halo parts are folded backward, therefore lines that are close together and have the same depth do not have halos. This way, the size of the halos depends on the difference in depth between the lines instead of the order of drawing the lines. This is illustrated in Fig. 3.

All operations are implemented on the GPU. Per point on the line we send two vertices and two corresponding parameters indicating how to move the vertices. The view-alignment and folding of the strip is handled by the shader code on the GPU, resulting in real-time frame rates.

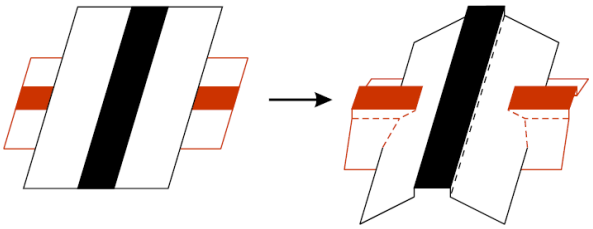


Fig. 3. Schematic view of the creation of the depth-dependent halos, adaptation from [6].

2.2 Enhancing Molecular Visualization

Where the previous technique operated on line data, we will now examine a method that uses both halos as well as ambient occlusion to clarify the structure in solid objects.

Tarini et al. [16] realized that both more and larger molecules were being cataloged and that a good method of visualizing these was necessary. Most traditional methods did not always succeed in showcasing the 3D structure of the (large) molecules. They focused on the space-fill, a rendering in which the size of the balls is increased to the Van der Waals radius, and balls and sticks approaches to rendering molecules. In these methods the individual atoms are rendered as spheres and their bonds as cylinders. Constrast this with the space-fill method where the cylinders are not visible.

The contribution of Tarini et al. is twofold. They combine ambient occlusion, silhouetting, and the optional use of haloing to greatly increase the depth perception, while at the same time allowing interactive visualization of large molecules. We first look into ambient occlusion, then we focus on how to render the atoms, and finally on methods for adding silhouettes and halos. We roughly follow the exposition of Tarini et al. [16] and refer to the original paper for the details.

2.2.1 Ambient occlusion

Informally, the ambient occlusion term measures how much of the lighting around the object actually arrives at a point [11]. We use the definition of the irradiance, E, to make this more precise [16]. Let p be a point on the surface and n_p the normal in this point, then

$$E(p) = \int_{\Omega} n_p \cdot \omega L(\omega) \mathrm{d}\omega, \tag{1}$$

where $L(\omega)$ is the amount of radiance arriving from direction ω, and Ω is the set of directions ω for which $n_p \cdot \omega \geq 0$. This equation could be used in a global illumination model, but we do not need the full complexity here. In order to simplify the equation we assume only diffuse external lighting and no specular reflection, therefore $L(\omega)$ reduces to $O(\omega)$ that is either one or zero, depending on whether the direction ω is occluded or not. Finally, we assume only a finite number of uniformly sampled discrete directions ω_i and the equation reduces to:

$$E(p) = \frac{1}{4\pi} \sum n_p \omega_i O(\omega_i). \tag{2}$$

In this form the irradiance can easily be computed using graphics hardware. A shadow map is rendered for each of the directions ω_i. This gives all irradiance information for all positions on the atoms, which is subsequently stored in a texture for later access. We will see in the next section how Tarini et al. solved this efficiently.

2.2.2 Rendering and textures

One contribution of the authors is the realization that spheres and cylinders can be more succinctly described using impostors, placeholders that are later expanded to the objects the represent. This way

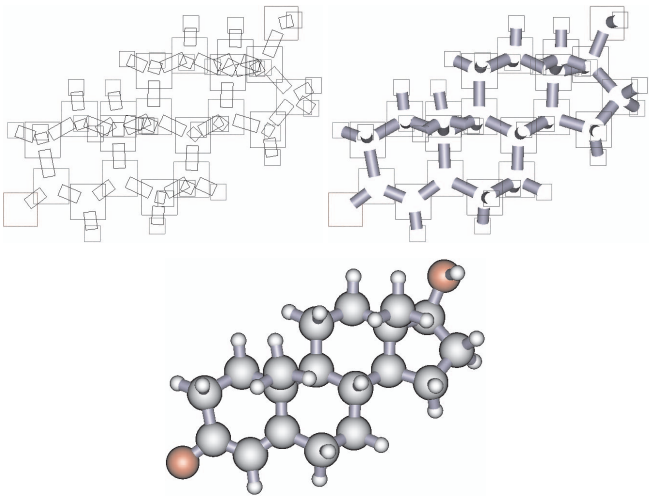


Fig. 4. Rendering a molecule with impostors. In the top left figure the impostors for the balls and sticks are shown. Note that the cylinders are projected onto the viewing plane. In the top right image the cylinders have been drawn, and finally in the bottom image the balls are shown as well. Images from [16].

only four vertices are needed per object instead of a complete tessellation. Both spheres and cylinders are represented by view-aligned rectangular impostors, see Fig. 4.

During rendering, these impostors need to be expanded to the actual objects. We explain the process for spheres, cylinders are handled similarly. The four vertices are mapped to a rectangular patch of texture coordinates, so $(s,t) = (\pm 1, \pm 1)$. The actual reconstruction happens in the fragment shader. Fragments with $|(s,t)| > 1$ are discarded as they are not on the sphere. The remaining fragments do lie on the original sphere, so we can construct their corresponding positions on an that sphere, and hence also their normal.

In order to access the occlusion information, the points on the surface of the sphere, parametrized by (s,t), are mapped by M into $[-1,1]^2$ in (u,v)-space. For each atom, a texture patch is stored containing the occlusion information, this patch is indexed by (u,v). The size of such a patch varies from 4×4 pixels per atom for large molecules (around 64K, so small atoms), to 32×32 pixels per atom for smaller molecules. These patches are stored in a single large texture, individual atoms get an offset into this texture space so the required information can be retrieved.

In the previous section we saw that the ambient occlusion term is the sum of the illuminated directions. Consider each of the directions ω_i in turn. We start by generating a shadow map by rendering the scene using ω_i as view direction. Then for each point p on an atom we use the shadom map to determine wheter it is visible or not. By combining this information for all viewing direction we can assemble the ambient occlusion texture. This requires an operation for each pixel on the large texture and for each direction ω_i, so a lot of preprocessing is needed.

Compare Fig. 5(a) and (b) for the difference between direct illumination and ambient occlusion. The latter seems to produce much better depth perception.

2.2.3 Improving visual quality

The authors use two techniques to further enhance visual quality: silhouetting and halos. Let us first consider the former. In fact, a more sophisticated technique is used: depth aware contour lines. Solid lines are drawn around each primitive, by setting every fragment with radius between R and $R+\varepsilon$ to black. Note that these lines automatically disappear at the intersection of atoms. The information provided by these contour lines can be improved by making them thicker if the jump between the primitives it separates is larger. This effect can be attained

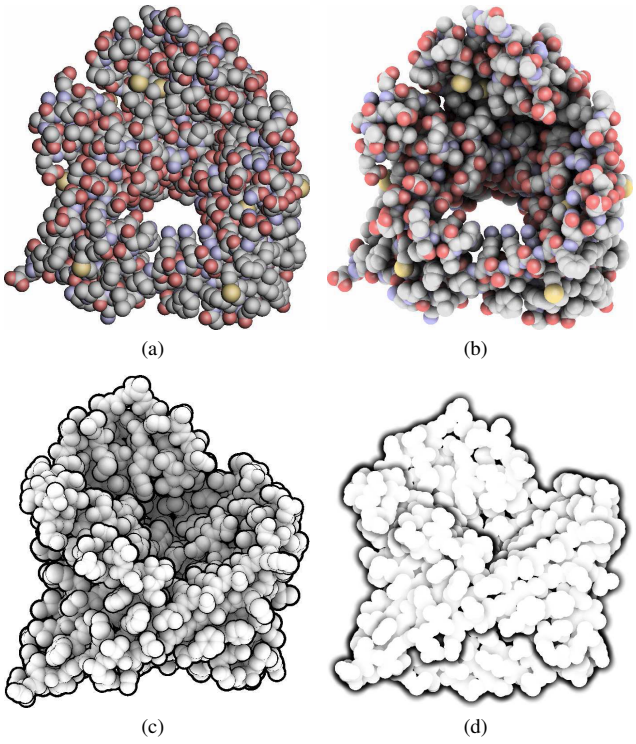


Fig. 5. Results of ambient occlusion and edge cueing method. (a): porin molecule (2219 atoms) rendered using direct lighting. (b): same molecule using ambient occlusion. (c): depth aware contour lines, (d): dark halos. Images from [16].

in the same way as was used for the depth-dependent halos by pushing the borders back.

In addition to the contour lines, the authors propose another haloing effect. A translucent halo is rendered around each object in a second pass. The depth buffer is used to make the halo more opaque if the objects it occludes is further away. See Fig. 5(d) for an example.

2.3 Flexible Volumetric Halos

As we mentioned in the introduction, one of the important topics in illustrative visualization is volume rendering. Bruckner and Gröller [3] developped an interactive technique to combine halos with volumetric rendering. Traditionally, volume renderings are produced by determining view-aligned slices and then accumulating these slices in a useful manner.

The contribution of Bruckner and Gröller is that their technique allows interactive control of the halos in the rendering. The basic idea is that for each slice a halo is generated which is then combined with the original slice. Combining all the slices gives a rendering including halos.

Bruckner and Gröller identify three basic stages for the production of the halo-image per slice: halo seeding, halo generation, and halo mapping and composition. We examine each of these steps in turn, following the general exposition in [3].

2.3.1 Halo Seeding

The volumetric data set is taken to be a scalar-valued function. Let f_P be the value of this function at position P and ∇f_P its gradient vector. The goals of this step is to identify locations where a halo should be placed. We saw in the previous two techniques that halos are placed on the edges of objects. We can use the gradient ∇f_P to mimic this effect. A sample point is on a contour if ∇f_P and the viewing direction are almost orthogonal.

To allow for different types of halo generation, a halo transfer function $h(P)$ is defined. It is the product of the three influence function

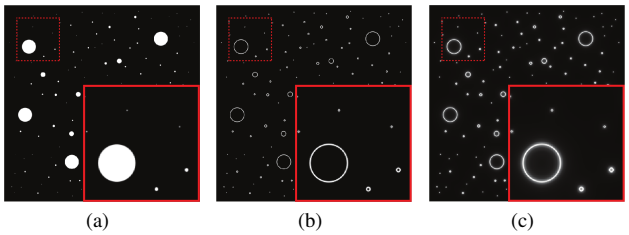


Fig. 6. Different stages in the halo seeding process, (a) the halo seeds, (b) the borders of the halo seeds and (c) the halos created from the border seeds. Images from [3].

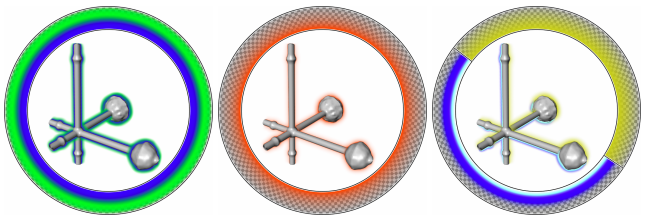


Fig. 7. Different halo profiles functions and their results on a simple data set. Images from [3].

controlling the effect of the value of the point P, the effect of the direction of the eye-space normal and finally the effect of the position of the point P with respect to some focal point. A wide range of effects can be achieved by combining various interpretations of these functions.

To prevent noisy effects in an almost uniform image from causing phantom edges, the magnitude of the gradient is taken into account. Combining these observations gives the following function for the halo seed intensity $s(P)$:

$$s(P) = h(P)|\nabla f_P|^{\alpha}(1 - \nabla f_P \cdot v)^{\beta}, \tag{3}$$

where v is the view-direction and α and β are control parameters (the values $\alpha = 32$ and $\beta = 0.125$ are reported to work well in practice). Fig. 6(a) shows the seeds for an example scene.

2.3.2 Halo Generation

The seeds given by $s(P)$ only define positions where a halo should start, but all of these are actually inside the object to be haloed, instead of outside. The process of halo generation extends the seed-image to a complete halo field that has halos outside the objects. To this end, first the gradient image is taken, Fig. 6(b). This image is subsequently blurred and blended with the original gradient image. This blurring and blending is repeated for a couple of iterations, giving halos of equal size around all seeded objects, Fig. 6(c).

2.3.3 Halo Mapping and Compositing

In the final step, the halo-field is converted to actual colors and opacity. The easy way to do this would be to just convert them to a constant color where the intensity maps to the opacity. In order to provide more control, the authors introduce a halo profile function that performs the mapping from intensity to color and translucency. Fig. 7 shows some examples of various profiles. To obtain the directional halos the normal-function is modified.

Two different kinds of halos are identified: emissive halos, halos that themselves emit light and are thus also visible when not obscuring other pieces of the volume; and occlusive halos that are only visible when occluding other pieces of the volume. Normally, halos are added in an additional front to back rendering. In the case of occlusive halos, the contribution of the halo-field is first limited to visible samples and only then drawn.

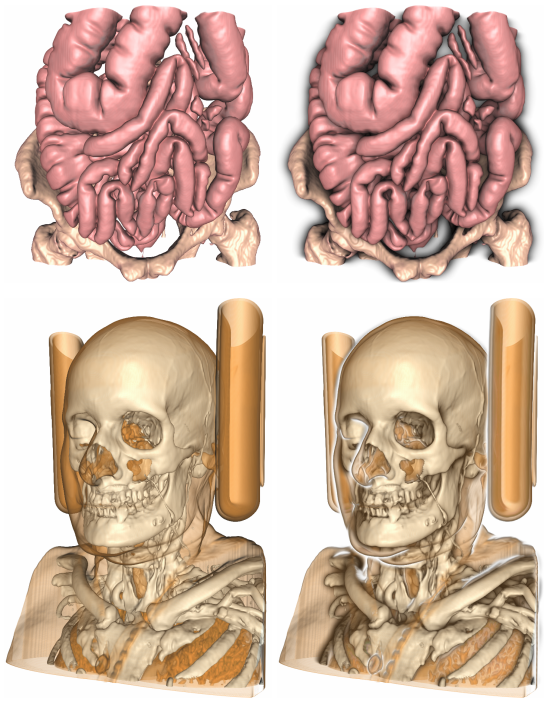


Fig. 8. Left column shows original rendering, right column shows halos. First row: a dark shadow-like halo is added. Second row: bones get a dark halo, while skin gets a light halo. Images from [3].

2.3.4 Results

Combining the three stages results in a method that is highly parameterizable, while still giving almost interactive results. The reference system without using halos produced about 30 fps, while the version with halos managed to attain about 10 fps.

Fig. 8 shows some examples of various renderings obtained by using the haloing methods. Comparison with the original methods shows significant improvements in depth perception.

2.4 Summary

We have seen three techniques for enhancing the depth perception that are designed for visualizing completely different types of data. All three of them have in common that they use illustrative techniques to clarify the 3D structure of the data, in particular halos. Since halos can be used for different types of data to enhance depth perception, they are a versatile tool for clarifying structure.

The three techniques do differ in their approaches. The depth-dependent halo technique works on line data. The flexible volumetric halos allow for a large variety of volumetric haloed renderings that are highly parametrizable. On the other hand, the enhanced molecular visualization techniques focusses on a very specific domain, but gains from this by designing a very specific and fast technique to handle this situation.

The methods also differ in the preprocessing steps needed. Molecule data is widely available in large database, but for volume rendering the situation is quite difficult. While the data is available, it is typically untagged and only by using expert input can the right parts be made visible/invisable. So manual preprocessing is often needed. The depth-dependent halos are somewhere in between, depending on whether the line data is directly available. If it is not it has to be calculated from for example DTI data and then selected for display.

All the techniques that we have presented here can be implemented on modern graphics hardware to give interactive results that are adaptable in real time. Furthermore, we saw that the different techniques can be combined to enhance depth-perception even further. For example, the halos can used with line-width attenuation to enhance the

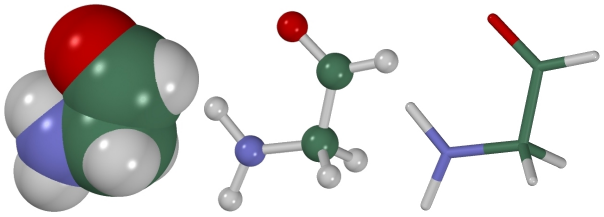


Fig. 9. Molecule representations, from left to right: space-fill, balls-and-sticks, and licorice. Images taken from [17].

effect.

3 Molecular Visualization

In the previous section we have seen the usefullness of adding halos and ambient occlusion in molecular visualization. Furthermore, we saw that it is possible to create high-quality black and white renderings of fibre tracts with good depth perception. In this section we propose a method for combining these approaches into a flexible method that can be used to explore the inner structure of a protein.

3.1 Styles of visualization

Before going into the details of our method, it is worthwhile to take a look at the various methods of visualizing molecules in general. Fig. 9 shows some examples of the three most commonly used methods of rendering molecules: space-fill, balls-and-sticks, and licorice. In a balls-and-sticks rendering the atoms are visualized as spheres, which have a color and size to indicate type. The bonds between the atoms are shown as cylinders. A licorice rendering is obtained by removing the spheres for the atoms, and coloring the bonds according to the atom types, see Fig. 9. Another alternative would be to grow the spheres to the Van der Waals atom radius, the result is a space-fill rendering of the molecule. The last method, ribbons, focuses on the structural aspects of molecules and proteins. Proteins often exhibit long helix-like structures of amino acid chains. In a ribbon rendering these are abstracted to ribbons. See Fig. 10 for two hand-drawn examples.

3.2 Inspiration

Before computers were used to visualize proteins, people drew illustrations of them by hand. Especially for the ribbon visualization, we were inspired by these hand-drawn illustrations. Fig. 10 shows two examples of hand drawn ribbons that both have a very different style. In the left figure, the ribbons are of a solid black color where the are stippled in the other illustration. In the right illustration the front and back of the ribbons are visually different, because in the ribbon structure the front is stippled and the back is plain white, where for the connecting structure, the back is striped.

In both pictures we see the use of arrows to show the direction of the structure, but the style and placement between both is quite different. The first illustration has a big arrow that looks as if it really is part of the structure. In the other illustration, the arrows are placed beside the structure and are a lot smaller. One last difference is in the using of halos, they are used in the left illustration, providing as sense of depth to the otherwise black helices. The use of halos is absent in the second picture, but this illustration still has depth information because of the difference in style for the back- and foreground of structures.

Not only the ribbon structure of a protein is used in illustrations, the atoms themselves are also used at times. Fig. 11 shows three different illustrations of the same protein. One thing that stands out is that it is difficult to see depth due to the lack of cues. The middle picture does show some important molecular structures, like benzene ring, which are not directly visible in the other two illustrations. When looking at the ribbon illustration, we again see the arrows on the structure. The leftmost picture shows what the surface of the molecule looks like, but again has no shading, making it hard in some parts of the picture to tell the exact structure.

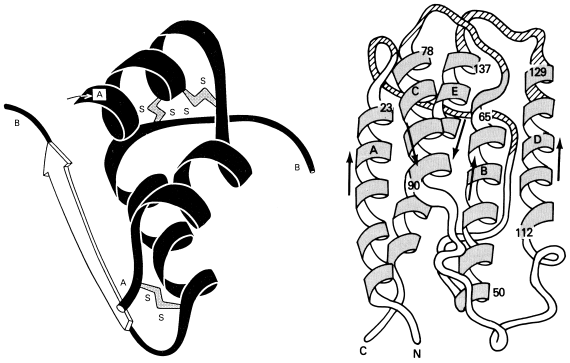


Fig. 10. Two examples of hand-illustrated ribbon visualization. From [13].

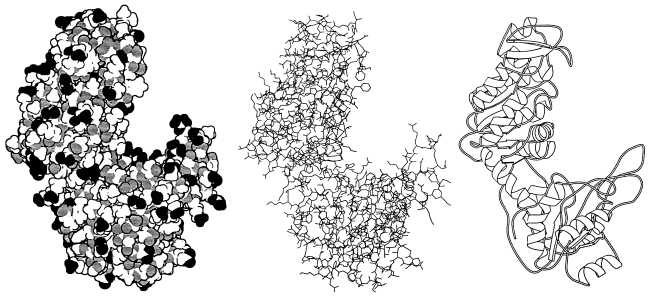


Fig. 11. Different illustrated representations of the same protein. From left to right: space-fill, licorice, and ribbon. Scanned from [9].

3.3 Data

A lot of research has been done involving proteins, leading to a vast amount of molecules that have been minutely researched. In order to produce the visualizations, the positions of the atoms, the amino-acid structures, the atom bonds, and the locations of the actual helices need to be known. Fortunately, all of this data is available via the RCSB Protein Data Bank [2].

3.4 Concept

Based on this data and guided by the inspiration we now describe how to create a structural abstraction function, that lets the user to interactively explore the inner structure of a protein. It smoothly changes the rendering style from completely concrete space-fill, to entirely abstract ribbon structures. In between, there is a transition from space-fill to balls and sticks by shrinking the molecule radius. The balls and sticks are then transforemd to a licorice rendering by decreasing the atom radii. Next, the internal structure is made more apprarent by successively removing atoms, starting with the atoms furthest away from the amino-acid chain. Finally, smoothed ribbons are drawn trough the amino-acid chains. In addition to the structural abstraction function, we have a halo-control function that determines the width of the halos, both around the atoms, like in section 2.2, as well as along the ribbons, like the depth-dependent halo method. Fig. 12 shows some results combining these methods.

3.5 Implementation

We applied various methods from Section 2 to create the different visualization styles. More precisely, we use the molecular visualization techniques from Section 2.2 for visualizing the atoms and atom bonds. Furthermore, we use the depth-dependent halos technique for adding halos.

The depth-dependent halos cannot immediately be applied to ribbon structures. The original method is designed to visualize line data, which are always represented as view-aligned strips. Ribbons, on the

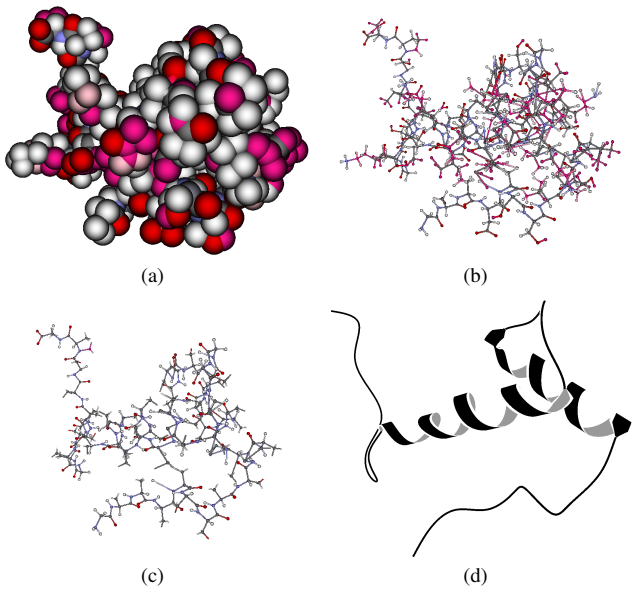


Fig. 12. Four different steps along the abstraction process are visualized: (a) shows a space-fill rendering. (b) and (c) show a balls and sticks rendering, where the latter lost some of the outer atoms. (d) shows the ribbon view with halos enabled.

other hand, are true 3D structures that can, in principle, be placed in any direction in space, so view-plane aligning is not possible.

First, a curve is fitted trough the nitrogen atoms in the amino-acid chains. This gives a parametrization $l(t)$ of the center of the ribbons. Let D be the direction of the helix. A ribbon can then be created by placing two vertices on the center of the ribbon and moving them away from the center along helix direction D.

In order to create the halos, the same approach is used as for the line data: a white border is drawn around the ribbon. This border still needs to be folded away from the view-aligned plane, which is no longer perpendicular to the ribbon.

3.6 Results

All of the combined techniques individually achieve real-time frame rates. The same holds for our implementation. In addition, we are capable of producing ribbon renderings that exhibit a combination of some of the illustrative techniques used for the hand-drawn examples in Fig. 10. We used a combination of the haloing in the left image, while we assigned different colors to the in- and outside of the helix, just as in the right image.

3.7 Future Work

While our abstraction function works the way we envisioned, we believe the program can be improved further. The first possible addition would be a function that controls the visualization style. Such a function would move from realistic shading, through gray-scale shading, to toon-shading and finally arive at illustrative rendering. The latter would produce pure black and white high-resolution images that use stippling or hatching to identify atom types. This achieves a rendering style similar to the right-most handdrawn example in Fig. 11.

At this moment we can only control the width of the halo, but we would like to expand this to also include some of the other techniques for a more sophisticated control of the spatiality indications. These include ambient occlusion and line attenuation. So we would then have three independent controls: abstraction, visualization style, and spatiality.

As another improvement, we think it might be useful to overlay two different levels of abstraction. For example the ribbon structure on top of the space-fill method. This can be accomplished by letting the top one fade away around the edges and thus revealing the structure underneath.

Finally, lens-based approaches could be used to locally show the structure at a different level of abstraction. For example, by completely showing one amino-acid, while the others are abstracted.

4 Discussion

In this paper, we presented an overview of three techniques to enhance depth and structure perception of different kinds of data sets. We discussed the ideas behind these methods, the applications and their similarities. Finally, we indicated difficulties with these approaches.

Our contribution is an innovative combination of some of these techniques to explore the inner structure of a protein. We believe that the smooth transition function we constructed provides a good method for exploring the structure of a protein. The combination with illustrative techniques have led to results that are comparable to the hand drawn ribbons in Fig. 10.

References

[1] A. Appel, F. J. Rohlf, and A. J. Stein. The haloed line effect for hidden line elimination. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 151–157, New York, NY, USA, 1979. ACM.

[2] H. M. Berman, J. Westbrook, Z. Feng, T. N. Gilliland, G. Bhat, H. Weissig, I. N. Sindyalov, and P. E. Bourne. The Protein Data Bank. *Acta Crystallographica Section D*, 58(6 Part 1):899–907, Jun 2002.

[3] S. Bruckner and E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007.

[4] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[5] G. Elber. Line illustrations $\in$ computer graphics. *The Visual Computer*, 11(6):290–296, 1995.

[6] M. H. Everts, H. Bekker, J. B. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, 2009.

[7] B. Gooch and A. Gooch. *Non-photorealistic rendering*. AK Peters, Ltd., 2001.

[8] A. Hertzmann. Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. In *Non-Photorealistic Rendering, SIGGRAPH Course Notes*, 1999.

[9] E. R. S. Hodges, editor. *The Guild handbook of scientific illustration*. John Wiley & Sons, 2003.

[10] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developer's guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl.*, 23(4):28–37, 2003.

[11] J. T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[12] H. Landis. Production ready global illumination. In *SIGGRAPH 2002 Course Notes*, pages 331–338, 2002.

[13] M. Perutz. *Protein Structure: new approaches to disease and therapy*. W.H. Freeman and Company, 1992.

[14] P. Rautek, S. Bruckner, E. Gröller, and I. Viola. Illustrative visualization: new technology or useless tautology? *SIGGRAPH Comput. Graph.*, 42(3):1–8, 2008.

[15] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002.

[16] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244, 2006.

[17] M. Valle, 2008. `http://personal.cscs.ch/~mvalle/ChemViz/representations/index.html` last visited 12 March 2010.

[18] I. Viola, M. E. Gröller, K. Bühler, M. Hadwiger, B. Preim, D. Ebert, M. C. Sousa, and D. Stredney. IEEE Visualization tutorial on illustrative visualization, 2005.

Creating Artistic Effects With Edge And Corner Preserving Smoothers

Sander Kikkert and Daniël Kok

Abstract— Painterly rendering is an image processing technique that focuses on turning a realistic image, e.g. a photograph, into one that looks like it has been painted by an artist. Edge and corner preserving smoothers (ECPS) are a class of filters that are able to generate artistic effects by blurring details away in low-contrast areas while preserving edges and corners. We present a survey of the Bilateral, Kuwahara, Papari-Petkov-Campisi and Kyprianidis-Kang-Döllner filters. All these filters (in this order) improve on a shortcoming of another filter. The new Kyprianidis-Kang-Döllner filter produces excellent results and overcomes almost all limitations of previous proposed filters.

Index Terms—Edge and corner preserving smoothers, Non-photorealistic rendering, Painterly rendering

1 Introduction

Non-photorealistic rendering (NPR) is a generic term for rendering techniques that produce images that resemble a style of digital art like drawings, (technical) illustrations and paintings. In this article we focus on painterly rendering, an image processing technique that focuses on turning a realistic image, e.g. a photograph, into one that looks like it has been painted by an artist. One way to obtain a painterly effect is by using Edge and corner preserving smoothers (ECPS). ECPS are a class of filters that smooth out texture details while preserving edges and corners. ECPS are used in different fields of image processing e.gimage preprocessing for other filters or image enhanching (removing noise). More recently ECPS are also used in artistic imaging to transform realistic images like photographs into images that resemble a style of painting (like an aquarel painting or pencil drawings).

There are several other ways to transform realistic images into painterly abstractions. One of them is the mean shift [2] filter which smooths low-contrast regions while preserving edges and corners [5]. However, it fails for high contrast images where either no abstraction is performed or too much detail is removed. A different approach to creating artistic images is stroke-based rendering. [3]. Stroke based rendering works by placing discrete elements such as paint strokes or stipples. There are several stroke based rendering algorithms. It can produce many different painting styles by using different types of strokes. However this approach also has some drawbacks: most algorithms are relativly slow so they cannot be used in interactive applications.

Our goal is to present a survey of ECPS filters that produce a painterly look. We have chosen filters that are related to each other in a way that they are all in some way built upon each other. Limitations from one are tackled in the next. We present the idea and output of each filter and how it relates to the previous filter and their limitations. We start our discussion with the Bilateral filter [8] which is the easiest filter to implement. After that we discuss the Kuwahara [6] filter, a well known edge and corner preserving filter which performs reasonable but has some drawbacks. Papari-Petkov-Campisi filter [7] is an improvement on the Kuwahara filter but fails to preserve local textures. Furthermore we discuss the Kyprianidis-Kang-Döllner [5] filter which is an improvement on the Papari-Petkov-Campisi filter. It incorporates directional information in the rendering to maintains local texture which makes it suitable for use in animation.

The remainder of the paper has the following structure: In section 2 we give a detailed overview of the filters. In section 3 we take a closer look at the results of the different filters. Finally we discuss conclusions and further research in section 4.

- *Sander Kikkert is with University of Groningen, Student, E-mail: S.C.Kikkert@student.rug.nl.*
- *Daniel Kok is with University of Groningen, Student, E-mail: D.C.Kok@student.rug.nl.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

2 Edge and Corner Preserving Smoothers

An important property of painting-like images is the absence of texture details. They are often smoothed while the edges and corners of objects are usually sharper than in photographic images. A linear convolution filter removes the texture details but it also decreases the sharpness of the edges, making the image look blurry.

In this section we give a detailed description of the following edge and corner preserving smoothers: Bilateral filter [8], Kuwahara filter [6] , Papari-Petkov-Campisi filter [7] and the Papari-Petkov-Campisi filter [7].

2.1 Bilateral filter

Bilateral filtering [8] is a non-linear filtering technique closely related to Gaussian filtering. It extends the concept of a Gaussian filter by weighting pixels in the vicinity of the center pixel based on intensity similarities. It uses two concepts of similar pixels: in terms of distance (domain) and in terms of range. In this case range is the range of intensity values. The power of the bilateral filter lays in the combination of these two. Domain filtering averages an area of local pixels to smooth out areas whilst the range filter makes sure that only pixels are taking into account that have similar intensity, preserving edges.

This filtering is applied to a square neighbourhood surrounding each pixel x in an image. In general, we need a combination of two common filters. The variables $\frac{1}{k_d(x)}$, $\frac{1}{k_r(x)}$ and k in (1), (2) and (3) are normalization constants. The first filter is a domain filter which can be defined as follows:

$$\phi(x) = \frac{1}{k_d(x)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) c(\varepsilon, x) d\varepsilon \tag{1}$$

Where f is the original image and $c(\varepsilon, x)$ is a closeness function producing the closness of a neighboutring pixel to the neighbourhood center x.

The second filter is a range filter and is defined in a similar way:

$$\phi(x) = \frac{1}{k_r(x)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) s(f(\varepsilon), f(x)) d\varepsilon \tag{2}$$

In this equation $s(f(\varepsilon), f(x))$ is a similarity function defining the similarity of the neighbourhood center x in terms of intensity value to another pixel in the neighbourhood ε.

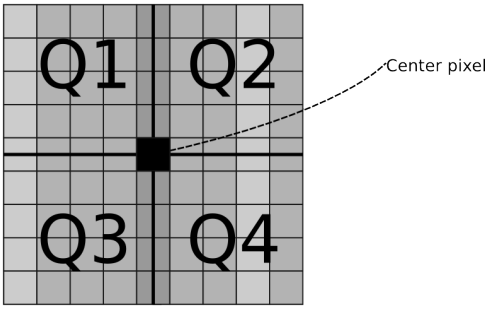


Fig. 1: A squared region divided into four equal regions used for Kuwahara filtering.

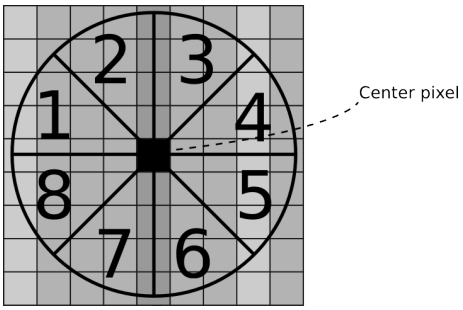


Fig. 2: A circular region divided into 8 equal sectors used in Papari-Petkov-Campisi filter.

2.1.1 Output

For the output we want a combination of filters (1) and (2). For range filtering we only want to take pixels into account that are near the the center pixel x i.e. pixels are most likely to be relevant to x. So, we combine it with domain filtering. Because both filters have the same integral we can multiply them giving us the following equation for the output:

$$\phi(x) = \frac{1}{k(x)} \int_{\infty}^{\infty} \int_{\infty}^{\infty} c(\varepsilon, x) f(\varepsilon) s(f(\varepsilon), f(x)) d\varepsilon \tag{3}$$

The result is a filter that preserves contours in the image because a pixel on an edge is only compared with nearby pixels that also have a similar colour value. Thus pixels on a black edge will not be averaged with nearby white pixels.

2.1.2 Limitations

Although the bilateral filter does a good job at preserving edges and smoothing out noise, it does not really give that painterly effect that we are looking for (fig. 6b). Especially in high contrast areas there is little abstraction performed while too much structure details are removed in low contrast areas [5].

2.2 Kuwahara filter

The Kuwahara [6] filter is an ECPS-filter with a relatively simple concept. The Kuwahara [6] filter uses a rectangular shaped mask. This mask is divided into four equal subregions with the current pixel in the center. For each subregion, we calculate the variance and the average value. The variance is used to determine the homogenity of a region. The region with the lowest variance is the most homogenous and is expected to give a better result on the average value i.e. it is least influenced by outlying pixel values. This is specifically important when regions are situated on edges or if they are corrupted by noise. We choose the average value of the region with lowest variance as the new value for the center pixel. If there are multiple subregions with an equal lowest variance, one of them is chosen randomly. Fig. 1 shows the mask divided into four regions: Q1 to Q4.

2.2.1 Output

When we let $m_i(x,y)$ be the average value of subregion Q_i the output of the Kuwahara filter can be defined like this:

$$\Phi(x,y) = \sum_i m_i(x,y) f_i(x,y) \tag{4}$$

Where $f_i(x,y)$ equals 1 if the standard deviation of the current subregion i is the minimum of all subregions.

2.2.2 Limitations

The Kuwahara filter suffers from block artifacts caused by the rectangular shape of the selection kernel. This is particular noticeable in textured areas. However this filter method has another problem: it does not always produce a single answer. If there is not much noise than the local averages could be almost the same. But if there is noise present than the local averages could be differ considerably resulting in a devastating effect on the final image.

2.3 Papari-Petkov-Campisi filter

The Papari-Petkov-Campisi [7] filter is an improvement on the Kuwahara filter. The shape of the regions is changed from a square to a circular region. The mask that is used is not a regular mask, but a Gaussian mask i.e. pixels close (smaller euclidian distance) to the center pixel get a higher weight than pixels that are further away (larger euclidian distance). The circular region is divided into N sectors. Like Kuwahara, the average and variance are calculated, but now they are based on the weighted pixels within the circular region. The pixels are weighted with the Gaussian mask. Fig. 2 shows an example of such region. In the equations to follow, I represents the original image.

A function V_i, called the cutting function, defines each sector S_i of the circular region. The final weighting function for a single sector is then obtained by multiplying the Gaussian function with the cutting function:

$$w_i = g_\sigma * V_i$$

2.3.1 Output

The function for the output stays the same as in (4), but in this case not only the subregion with minimum standard deviation is chosen but a weighted standard deviation. This is obtained by dividing the standard deviation of a sector by the sum of all standard deviations. This gives us the following equation for the output:

$$\phi(x,y) = \frac{\sum_i m_i s_i^{-q}}{\sum_i s_i^{-q}} \tag{5}$$

Where m_i is the resulting image of convolving with the weighting function w_i and s_i convolved with the weighted standard deviation which are calculated as follows:

$$m_i = I * w_i \tag{6}$$

Note that multiplying by w_i results in a weighted average because

$$s_i^2 = I^2 * w_i - m_i^2 \tag{7}$$

In (5) q is a paramater to change the nature of the function. When q is 0 then then 5 equals a linear Gaussian filter i.e. the standard deviations do not influence the averages anymore resulting in a gaussian weighted average. When $q \to \infty$ the function reduces to the Kuwahara filter. For all values in between the filter behaves like a Gaussian filter in homogenous areas and an edge preserving filter around object contours: In an homogenous area the standard deviations of each sector will not differ a lot. On object edges, the sectors that are on the boundary of an object will have a higher standard deviation and thus a different weight than sectors that are completely on the object. Fig. 3 shows how multiple sectors of the disc can influence the output value. Because some sectors lay over the object edge their standard deviation

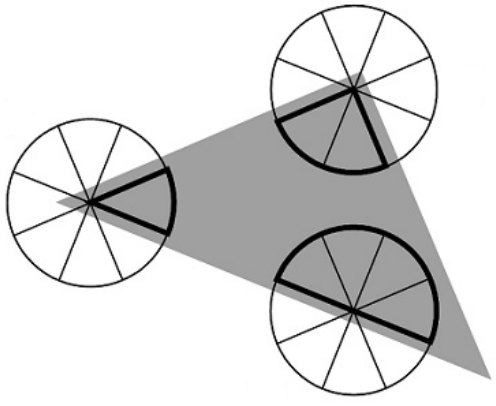


Fig. 3: Example of how the multiple sections can influence the output value (image from [7])

is significantly higher than the the sectors that lay completely on the object thus giving them a lower weight.

2.3.2 Limitations

While this filter is a huge improvement on the Kuwahara filter it does not capture directional information of local structures in an image which is especially useful when using the filter for animation purposes. Also this filter still suffers from (circular) clustering artifacts although less noticeable than the block artifacts of the Kuwahara filter. Also shape boundaries are less than optimal preserved due to the isotropic nature of the subregions.

2.4 Kyprianidis-Kang-Döllner filter

Kyprianidis, Kang and Dollner [5] made another improvement on the Kuwahara filter that preserves the local feature directions than the filter proposed by Papari-Petkov-Campisi. It uses a smoothed structure tensor to adapt the filter to the local structure, resulting in sharper edges and a feature conserving painterly effect. The direction of the local structure is used to adapt the shape of the circular mask. Because directional information is taken into account, the filter maintains temporal coherence which is useful when working with animations. There is a GPU implementation of this filter that processes video in real-time.

2.4.1 Implementation

As stated above the Kyprianidis-Kang-Döllner filter [5] uses directional information of the input image. This information is obtained by using a so called *structure tensor*. A structure tensor is matrix representation of the partial derivatives in x and y direction. (For a better understanding of a structure tensor see [1]) The definition of the structure tensor matrix is as follows:

$$S = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{8}$$

By definition, the eigenvalues of the structure tensor correspond to the squared minimum and maximum rate of change. The eigenvectors correspond to the respective directions. When we select the eigenvector with the minimum rate of change, we retrieve a vector field shown in fig. 4 on the left. The vectors in the vector do not seem to match the image very closely. To overcome this problem a Gaussian smoothing filter is applied to the structure tensor which results in a smooth vector field shown in fig. 4 on the right.

The next step is orienting the region mask to correspond to the structure of the image. This is done by measuring the anisotropy by using a method proposed by Yang-Burger-Firmin-Underwood [9] where an anisotropic region is directionally dependent and an isotropic region is not i.e. a homogenous region. The anisotropy is measured as follows:

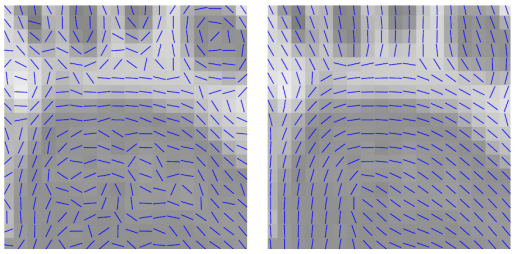


Fig. 4: Left: vector field obtained by computing eigenvalues from structure tensor. Right: vector field after Gaussian smoothing. (image from [5])

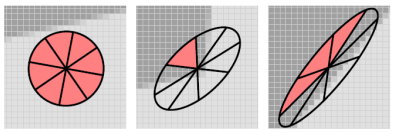


Fig. 5: Kyprianidis: circular regions adapted to the local structure. (image from [5])

$$A = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \tag{9}$$

Where $\lambda 1,2$ are the eigenvalues of the structure tensor. This produces a value A between 0 and 1

The image in fig. 5 from [5] shows three examples of an ellipse shaped regions adapted to the local structure of the image. The major axis of the ellipse is directed along the direction of the local structure. With this information weighting functions are defined in a similar way to that in Papari-Petkov-Campisi [7].

2.4.2 Output

The final output operation looks similar to the one described in Papari-Petkov-Campisi [7].

2.4.3 Limitations

Kyprianidis-Kang-Döllner [5] describe one limitation of this filter. It cannot create a "rough and brushy" look like oil paintings of Vincent van Gogh. If one would create such more strong texture brush effects the paper suggests to incoparate background paper texture or a directional, cumulative alpha into this filter.

3 Results

Fig. 6 shows a comparison of the described filters. The Bilateral filter (fig. 6b) blurs low-contrast areas while preserving edges and corners. The disadvantage of this filter is that in high-contrast areas (such as the fur of the lion) no abstraction is performed while in low-contrast areas too much details are removed.

The Kuwahara et al. filter performs more abstraction in the high contrast areas as can be seen in fig. 6c (the fur looks now more painterly) while still preserving edges and corners in the low-contrast areas. However the Kuwahara has some limitations. One of them is the block structure of the output which can be clearly seen in (especially on strongly textured areas) fig. 7b. Another problem is the instability of the filter. However this cannot be seen in the sample image.

The Papari-Petkov-Campisi filter solves the instability issue by defining a new criterion for the selection function (see the implementation section for more details). This results in less artifacts and smoother edges and corners (fig. 7c). However (circular) clustering artifacts are still present and it fails to capture directional image features.

Fig. 6d shows the outstanding results of the Kyprianidis-Kang-Döllner filter. Due to the structure tensor and adaptive filter kernel

(a) Original image (b) Bilateral filter (c) Kuwahara et al.

(d) Kyprianidis-Kang-Döllner (e) Papari-Petkov-Campisi

Fig. 6: Comparison of all the described filters (images from [4]).

directional information is excellently preserved. The landscape of fig. 7d for instance has much more dept due to better preservation of directional features. This filter also does not suffer from artifacts, noisy pixels or rough boundaries. Even the lion whiskers (fig. 6d) are not blurred away (like the Papari-Petkov-Campisi filter and the Kuwahara et al. filters).

4 Conclusion & Future Research

We presented a short survey of the most important and promising edge and corner preserving filters. The Bilateral filter [8] smooths low-contrast region while preserving edges and corners. However this filter fails when smoothing high-contrast areas. Too much details are removed or (depending on the variables) no abstraction is performed.

A better edge and corner preserving filter is the Kuwahara filter. This filter uses a special rectangular based selection kernel where sub-region with the lowest variance is selected to determine the value of the output pixel. Due to this more complex filter method the filter handles successfully the limitations of the Bileral filter. Details in high-contrast regions are removed while edges and corners are preserved. However this filter has some drawbacks: the output is unstable in presence of random noise and the outputs has some artifacts.

To overcome the limitations of the Kuwahara filter Papari-Petkov-Campisi has made an improved edge and corner preserving filter. They introduced a different shape selection kernel and a new selection rule. That results in improved output however artifacts are still noticeable.

The Kyprianidis-Kang-Döllner filter is the last filter described is this article. It uses a structure tensor that determines the orientation of the ellipse shaped filter kernel. This anisotropic filter kernel is used as the selection function of the output. This innovative approach results in a filter that gives a nice form of painterly abstraction, maintains edges and corners and preserves local texture directions. Unlike the other described filters this filter is solid against noise but does not blur all details in low-contrast areas resulting in a uniform level of abstraction of the whole image. This filter is also suitable for frame for frame video rendering due to the temporal consistency of the output. All in all this filter gives by far the best results.

Some suggestions for future research include: to optimize the Kyprianidis-Kang-Döllner filter to use less resources so it can be used in embedded systems like digital still cameras or video as a nice effect.

References

[1] S. Arseneau. Tutorial and demonstration of the uses of structure tensors using gradient representation, Sept. 2006. http://www.cs.cmu.edu/ sarsen/structureTensorTutorial/.

[2] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603 –619, may 2002.

[3] A. Hertzmann. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications*, 23:70–81, 2003.

[4] J. E. Kyprianidis. Image and video abstraction by anisotropic kuwahara filtering, June 2009. http://www.kyprianidis.com/pg2009.html.

(a) Original image

(b) Kuwahara et al.

(c) Papari-Petkov-Campisi

(d) Kyprianidis-Kang-Döllner

Fig. 7: Unlike the Papari-Petkov-Campisi and Kuwahara filters the Kyprianidis-Kang-Döllner preserves texture details (images from [4]).

[5] J. E. Kyprianidis, H. Kang, and J. Döllner. Image and video abstraction by anisotropic kuwahara filtering. *Computer Graphics Forum*, 28(7), 2009. Special issue on Pacific Graphics 2009.

[6] S. E. M. Kuwahara, K. Hachimura and M. Kinoshita. Processing of ri-angiocardiographic images. *Digital processing of biomedical images*, pages 187–203, 1976.

[7] G. Papari, N. Petkov, and P. Campisi. Artistic and corner enhancing smoothing. *IEEE Transactions on Image Processing*, 16(10), 2007.

[8] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 839–846, 1998.

[9] G. Z. Yang, P. Burger, D. N. Firmin, and S. R. Underwood. Structure adaptive anisotropic image filtering. *Image and Vision Computing*, 14(2):135 – 145, 1996.

E-Government based on service architecture

Margreth Venaely Kileo, Alexander Bograd

Abstract— Governments around the world are aware of the role of information and communication technologies (ICT) in delivering public services. Accordingly, governments are increasingly utilizing ICT as a key facilitator for the purpose of meeting their stakeholders' expectations. This helps to improve the relationship between government, citizens, businesses and other governments. However, there are a lot of challenges in implementing e-Government initiatives. Architectural approach can be used as the solution to overcome those challenges. In this paper we present the Service Oriented Architecture (SOA) as an appropriate approach for implementing e-Government initiatives. This approach aims at allowing reuse of experience and resources. Such resources are usually used by government agencies in implementing e-Government initiatives. Finally, we present the layers decomposition of SOA and how it can be useful in contributing to approach the challenges in implementing e-Government.

Index Terms— E-Government, e-Governance, Service-Oriented Architecture, SOA, Services, XML, Web services

1 Introduction

E-Government is the use of modern information and communication technologies to transform government. The transformation is intended to make the government more accessible, effective and efficient in providing its public services. E-Government is envisaged that government departments will use their processes more efficiently and reduce costs in the administration [1].This improves efficiency and service delivery to citizens and promotes transparency.

In the traditional e-Government implementation, various challenges are often faced. They include lack of knowledge in granularity programming, agility and expansibility. Additionally, most of the government systems have been developed with different programming languages for different operating systems [6]. This is because each local government has its own information system for providing public services. The departments do not share data or services. It happens because of the lack of interoperability and collaboration between government departments. This jeopardise the performance of delivering public services and makes them not efficient and effective.

Nowadays, most of the governments are aware of the problems, where they use communication tools as powerful means to increase collaboration and cooperation between different public institutions. Those communication tools facilitate and encourage development of homogenous web platform. It would provide only single authentication to access multiple public services, such as car registration services, tax services, birth certificates services and other social services [5]. Due to that there are a lot of challenges in implementing e-Government initiatives in the distributed systems because of the traditional background of e-Government explained above.

We present the Service Oriented Architecture (SOA) as an appropriate approach for implementing e-Government initiatives. SOA is an abstraction of the systems to guide and study the development of these systems [11]. SOA enables Government to achieve interoperability and dynamic reconfiguration of e-Government services. Due to that, the performance of the public services increases with better accessibility, more transparency and manageability.

SOA relates to the philosophical approach for creating distributed systems. There are different SOAs which have different standards in the implementation level. This includes web services based on SOAP, GRID services based on OGSI and REST services based on HTTP and XML. But this paper focuses on the web services that are based on SOAP. However, it is necessary to wrap the existing software systems in a service interface in order to be more accessible to the client. This provides a wide range for distributed systems to interact each other [12].

The aim of this paper is to propose SOA as an approach for overcoming the challenges in implementing e-Government. The goals of e-Government are first discussed, followed by the challenges of e-Government implementation. The paper also aims at identifying decomposition layers of SOA and describes features of e-Government Portal, which is based on SOA**.** Finally, advantages and shortcomings of using SOA in implementing e-Government are discussed.

2 The Goal of E-Government

The goal of e-Government is to provide effective delivery of public services to the stakeholders of government. The stakeholders include citizens, businesses, employees of the government and other governments. E-Government also aims to provide the stakeholders with accessible and integrated services in a more efficient manner. In advanced stages of e-government, it is possible to allow stakeholders to access multiple public services through a single entry and authentication. This allows stakeholders to access public services without the need to physically contact the government. The overview of the e-Government for Dutch Municipalities is shown in figure 1, as an example, which presents objective of the Dutch e-Government. The figure shows that government municipalities can interact with citizens through single application called myGov [8]. The interaction between them is called "Government to Citizens" interaction, in which it allows different citizens to access public services from different municipalities. Government also can interact with different existing systems and external systems such as, SMS gateway, Google maps and DigiD. Those interactions increase accessibility and reliability of the service and it provides interoperability and integribility between different existing systems.

- *Alexander Bograd is a CS student at the Rijkuniversiteit Groningen, E-Mail A.Bograd@student.rug.nl*
- *Margreth Venaely is a CS student at the Rijkuniversiteit Groningen, E-Mail M.V.Kileo@student.rug.nl*

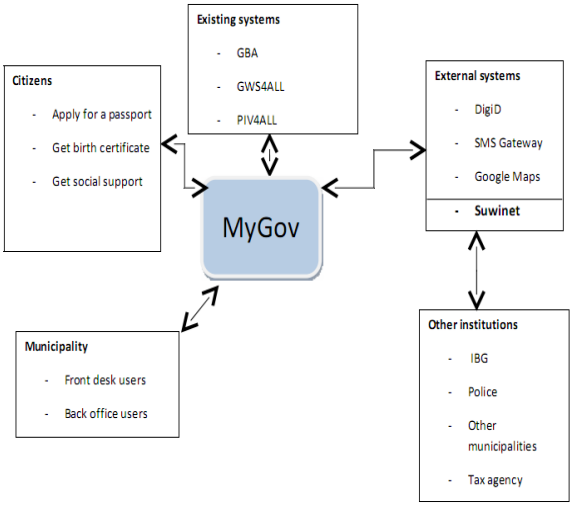


Figure 1. E-Government [8].

3 Challenges of Implementing e-Government Initiatives

There are many challenges that have to be considered in implementing e-Government initiatives. In this paper we discuss some of them, like interoperability, reusability, accessibility, scalability and security of public services. These challenges can be grouped into three categories, which are: service related, technology related and governance related.

3.1 Service Related

In implementing e-Government there are many challenges in improving public services; accessibility of the public services is one of it. Accessibility demonstrates to what extent is the service accessible at anytime or anywhere when it is needed. The goal of e-Government is to have effective and efficient service, which can be accessible at any time. Affordability of the service shows to what extent is the service affordable in terms of cost, relative to the services offered. User friendliness of the service should favour the citizens and public sectors, where the service has to be easy to use and government's staff should be able to use and run it in a friendly way.

.

3.2 Technology Related

One of the technology related challenges in implementing e-Government initiatives involves integribility. Government departments could have different specifications of formats, operating systems and information systems. Interoperability is the ability to provide service to the citizens, businesses and public administration in a collaborative manner. However, exchanging data can be a challenge, since data from different organizations or departments are interpreted differently. Therefore, the collaboration takes time, difficult and not efficient.

3.3 Governance Related

Reusability and processing re-engineering is a challenge for implementing e-Government. This aims to improve the efficiency of a business processes by redesigning and rethinking of business processes, in order to achieve improvement in quality and speed in the service offered. Business analysts handle all the processes. The challenge in scalability is how to handle many applications, so that they can be accessed by the user as multiple services. Challenge in Governance is how to ensure the confidentiality of the citizen's information, this means trust worth, which intends to address the security matter.

Nowadays, many researchers are working to find a way to improve the services provided by e-Government. However, preliminary analyses from those researchers show that the use of SOA and web services tends to increase the performance of the service provided by the government. SOA can enhance the agility and flexibility of using e-Government [1].

Therefore, in the next section we will describe how SOA used in implementing e-Government in order to meet its goal.

4 Service Oriented Architecture (SOA)

Service Oriented Architecture is a technical term which provides flexible design principles, which are used in implementing e-Government initiatives. It provides a loosely-integrated suite of services which can be used within the multiple application domains and these services can communicate with each other [11].

The SOA is not a new approach to support integration for distributed systems, DCOM or Object Request Broker (ORBS) which has CORBA specification was the first architecture to be. In the mid-90's those technologies were used in order to achieve the desired level of integration between different information systems. However, the challenge was complexity of the systems; they were growing fast, due to the increasing number of endpoints. Therefore, specific adapters to define the API'S were used, so each time when the endpoints changed then the interface had to be redefined [7].

Today the integration application between different information systems is at high level of abstraction, where web technology is used to provide the integration. Web technology is based on web services, Common Gateway interface (CGI) and Service registration language (UDDI, WSDL). Web service is a technology which uses XML to receive and send messages in order to retrieve services. It uses web service standards to distribute services through the interface. Therefore, SOA is not really new, but it has been around for 20 years. Moreover, web services open new integration possibilities [9].

The general concept of service oriented model is used to describe service as a unit of work or application which was created or published by the service provider [11]. This model shows that there is a service provider to offer services by identifying its interfaces and implementing the service functionality. Also, there are service consumers who can access the service, provided by service producer [3].

As shown in figure 2, service provided by service provider can be published by registering it to the service directory in which the service consumer can access it. All the interactions between service provider and service consumer are based on web service technologies. Those services can be accessed through internet by a user [11]. This model can be used for implementing e-Government, where different public sectors can be used as service provider to publish the services, so that they can be shared. Citizens and other public sectors can be used as service consumers by accessing those services.

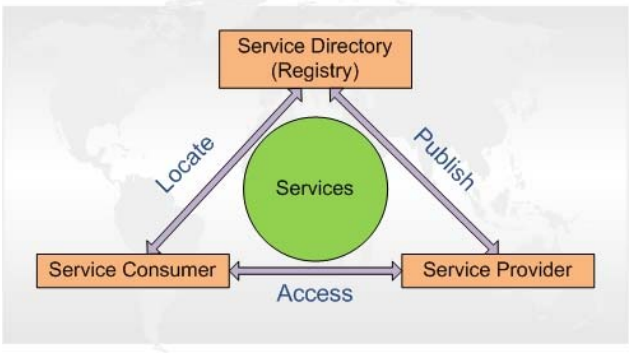


Figure 2. Service Oriented Architecture [4].

5 LAYER DECOMPOSITION OF SOA

Layered designs in information system are based on the separation of concerns. SOA can separate concerns into services by decomposing into layers. Decomposition of SOA into layers enables it to overcome most of the challenges in implementing e-Government initiatives. This provides an efficient way of achieving accessibility, reusability, integribility, security and interoperability of public services provided by government. SOA as an architectural approach can implement e-Government initiatives by reusing components as shared services [10]. These services can be integrated into different layers where they can be reused [2]. In this section we will describe layers of SOA and explain how these layers enable implementation of e-Government initiatives.

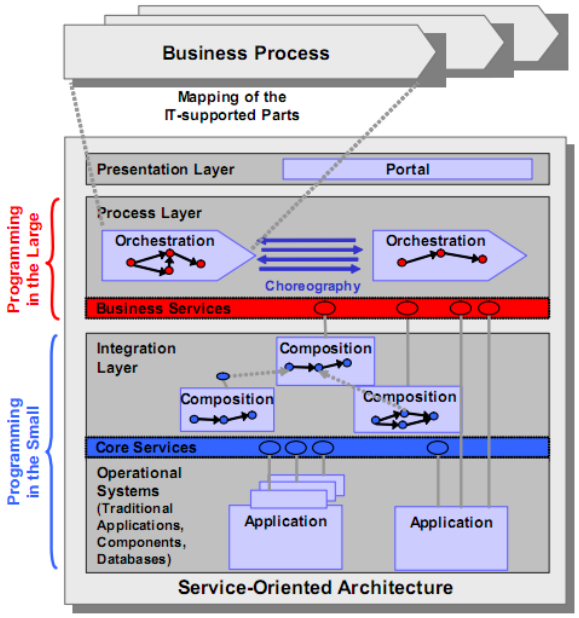


Figure 3. Layers of SOA [2].

5.1 Operational Layer

This is the bottom layer of SOA, as shown in figure 3; it consists of operational systems or legacy systems. These systems are traditionally developed and not Service Oriented. It does not matter from the SOA perspective if these legacy systems are internally monolithic or belong to multi-tier architectures. SOA approach has the task to leverage these systems by exposing their functionality as reusable services [2].

This layer exposes the interface of core services. The wrapping of existing functionality to services achieves at core service border. This can be done with common interface description and communication protocol. Wrapping is the process of capturing all the system functionality in a new service interface and converting them to the appropriate API [12].

Software developers do the act of exposing this functionality as service [2]. This enables the service to gain reusability that is held in service directory (Registry), where it can be accessed and reused, as shown in Figure 2.

Operational layer is also responsible to handle the scalability of the services, because it has to provide different multimedia information, such as: pictures, sounds and video through the same platform. The scalability is enabled in the layer in which different stakeholders can access the multiple public services that are already registered in the service directory [3].

As shown in Figure 3, services are grouped into core services and business services.

5.2 Integration Layer

This layer, as shown in Figure 3, exposes the composition of web services that are exposed in bottom layer. In this layer, the core services define the border at which SOA is reached by the existing applications. Therefore, services can now compose at the integration layer.

The term composition denotes the combination of services that yield more complex service [2]. There are different approaches in creating service composition. It enables different government organization units to interconnect their applications and allow data sharing in distributed environment.

These approaches are briefly explained in this section. BPEL4WS (Business Process Execution Language for Web Services) supports process-oriented form. Each BPEL composition is a business process or workflow, which can interact with the set of web services to achieve a certain task. Another approach for creating service composition is OWL-S (Web Ontology Language), which is a part of semantic web vision. OWL-S is a service ontology, which enables service automation, interoperation, composition and execution monitoring. This approach treats the services as web components, which present their interface and operation in class definition, where it can be published and reused. Pi-Calculus is another approach used to create service composition where it is an algebraic method, which is used to describe processes. Petri net is one more approach, it uses graph nodes to represent places, transitions and token [11].

In the integration layer, the field of classical Enterprise Application Integration (EAI) is used. It is now standardized to service interface description (e.g. WSDL, Web Service Description language). Systems can integrate at this layer by using approaches explained above for creating service composition (e.g. BPEL) through communication protocol (e.g. SOAP, Simple Access Protocol) [2].

As shown in Figure 3, when we reach the business service border, the dependencies underlying the software system have decreased to a minimum. In which the relation to the underlying software system will be hidden .This allows business analysts to map their business process to the process layer.

Integration layer enables the implementation of e-Government initiatives, because it allows the integration of different government applications. This integration forms a single application (e.g. e-Government Portal), which involves service provider and service consumers; despite of the deference's in government applications, operating systems, programming language and data formats. So, this layer is used as communication bus infrastructure to provide communication between government applications and the stakeholders of the government, which can be implemented by web services technology, as explained in the last section [1].

Also, this layer in implementing the e-Government initiatives provides collaboration between different government departments. In which those departments can share data or document by using XML messages. In addition, this layer enables the effective and efficient way of government to provide its public services [1].

Therefore, this layer overcomes the challenges related to technical in implementing e-Government.

5.3 Process Layer

In this layer business analysts are free to create or modify their new process through the orchestration of various business services. The term Orchestration in this context means the act of plugging together different business services in order to accomplish business logic and processes. The choreography is used to coordinate business process and describe the communication protocol between business services. Therefore, as shown in Figure 3, it enables collaboration between orchestrations [2].

Large programming is done by business analysts in order to combine the composite service and process orchestration. This is because orchestration and choreography are related to business process, so, this has to be handled by business experts, rather than system integrators. System integrators are doing small programming in which integration aspects have to be added [2].

In this layer the Governance related challenges are solved because it allows Business Process Redesign. This improves the performance of the process within the government departments, in which the redesign process ensures the effectiveness in delivering services.

The main stages of Business Process Redesign methodology are map existing process, defining the end state, gap analysis, redesign of workflow and processes. The adoption of this methodology to e-Government enables reusability of services, because this layer consists of different components, which are: transaction services, workflow services, form services, search and notification [1]. These services can be reused by other government departments through Process Orchestration and choreography [2].

5.4 Presentation Layer

In presentation layer, human users can integrate with the present process on the process layer. As shown in Figure 2, there is a portal, which is used as delivery channel to access the service through the internet [2]. In this layer accessibility of services is enabled through the secured gateways. These gateways are: XML gateway, Short message services (SMS) gateway and web gateway [1].

This layer solves the service related challenges to implement e-Government initiatives. It allows accessibility of the service through the interface component, where different users can access the public services in which it depends on which portal is used. For example, if it is a web portal, user can access the services through a web browser. It provides user friendliness of accessing the services because, they can be accessed anytime, anywhere and when needed [1].

Also, security is one of the issues, which is solved by SOA approach in the implementation of e-Government. The decomposition of layers of SOA enables security to the services because there is no direct connection with the applications database [2].

6 Features of E-Government Portal

E-Government portal, discussed in this section, is used as an example of e-Government, which has been implemented by using SOA. E-Government portal consists of different government applications, which are based on Service Oriented Framework. It acts as the middleware between government departments and their clients [1]. In this section, features of e-Government portal are explained. The portal helps in fulfilling the goal of the e-Government. To begin with, e-Government portal has to adapt SOA approach in order to solve challenges as explained in the previous sections.

Second, it is able to develop business functionality as a service. E-Government portal must provide web based interface which helps users to access the public services, as shown in figure 4; it uses different delivery channels for users to access it. These are: web, common service center (CSC), mobile and call centre. Different users can access public services from government departments by using those channels. Therefore, e-Government portal must be compactable to those delivery channels [1]. Next, it has to ensure the confidentiality of citizens' data, by using the security mechanism, such as authentication and authorization [1].

Finally, e-Government portal must be able to integrate with the government departments. The integration between those governments' department applications has to adapt SOA Framework. Therefore, e-Government portal has to be realized by technology platforms. These platforms are: web server, application server, middleware and directory server [1]. Refer to figure 4.

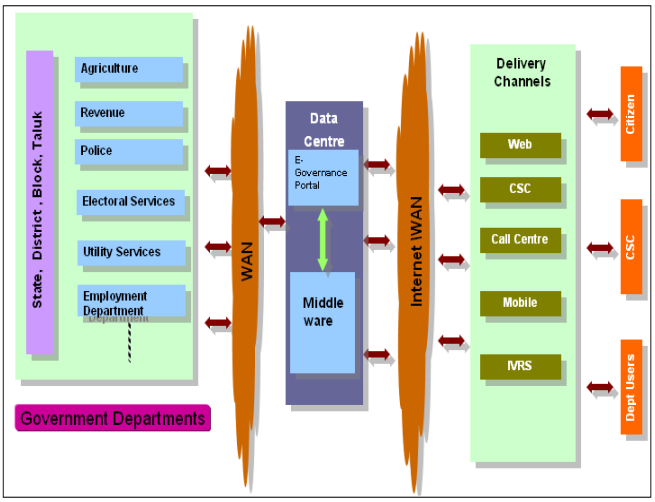


Figure 4. E-Government Portal [1].

7 Conclusion

This paper has outlined some goals of e-Government and challenges in implementing e-Government. SOA has been proposed as an approach for implementing e-Government initiatives, where a specific emphasis is on using enabled services. The paper also proves SOA as an appropriate approach to facilitate in designing public systems. The decomposition of layers of SOA also enables us to overcome different challenges in implementing e-Governments initiatives. Therefore, SOA is the relevant solution for improving accessibility, security, scalability, reusability and integribility of public services. This is because it supports collaboration and transparency within public sectors. It also saves time, where user can access multiple services at once through the internet. This reduces physical interaction with the public sectors and the cost of administration.

It is possible to minimize the need for re–writing the code every time, because SOA enables reusability of services and data sharing which increases efficiency and performance of the public services.

Government can use SOA for workflow management and automating business processes. It also provides service consolidation by making existing legacy system more accessible through a single e-Government portal.

Although SOA is relevant approach for implementing e-Government, but also it has some important limitations.

First of all, SOA security is difficult to implement in e-Government, because its security has to reconcile and interoperate between multiple security models and mechanism in real time. The issue is not only to how to deal with integrity and confidentiality, but also how to federate and govern those security policies between government departments.

Second, SOA lacks business service life cycle management, which means that collaboration should be highly needed between government and their stakeholders. All the stakeholders should have the same information within their own context.

Finally, SOA has service level compliance, which means there are some inconsistencies, which can be hindrance, because business services are composed of different web services.

REFERENCES

[1] Gopala K. B., Vishnu V.V. and Madhusudhana R., October 2009, Service oriented architecture for E-Governance.

[2] Christian E., Kim L., Karsten K., Stefan L., Christof M., Sebastian A., SOA layers.

[3] Sommerville I., 2007, Software Engineering, United State of America

[4] Technology spotlight, cloud computing, Service oriented Architecture [online] (updated 11 October 2008) Available at: http://www.tekspotlight.com/ [Accessed 16 April 2010]

[5] University of Oslo ,Web service architecture: A solution of e-Government application [online] (Updated April 2006) Available at: http://whitepapers.zdnet.com/abstract.aspx?docid=286018 [Accessed 13 April 2010]

[6] Zhang N., Li Y., 2008. Study and application of the SOA based E-Government system.

[7] Barry & Associates, web services and SOA [Online] Available at: http://www.service-architecture.com/web-services/articles/dcom.html [Accessed 13 April 2010]

[8] Tofan D., T., Gauke V., 2009. E-Government and SOA –Software patterns assignment.

[9] University of Oslo, Web service architecture: A solution of e-Government application [online] Available: http://www.emu.edu.tr/aelci/COMPSAC/PapersToReview/Paper56.pdf[Accessed 16 April 2010]

[10] Marijn J and René W.W. Developing Generic Shared Services for e-Government [Online] 2(1) Electronic Journal of E-Government Available at: http://www.ejeg.com/volume-2/volume2-issue-1/v2-i1-art4.htm [Accessed 16 April 2010].

[11] Ralph W.F., Marijn J and René W.W.,2007. Evaluating web services composition method [Online] 5(2), Electronic Journal of E-Government Available at: http://www.ejeg.com/volume-2/volume2-issue-1/v2-i1-art4.htm [Accessed 16 April 2010].

[12] David E. Millard, Yvonne H., Swapna C, Hugh C.D., Ehtesham-Rasheed J., Lester G., Gary B.W., Design Patterns for Wrapping Similar Legacy Systems with Common Service Interfaces.

Scaling Websites to Retain Availability

Meiburg, Y. Naber, A.L.

Abstract—The internet becomes larger and larger, and websites become ever more popular. Social websites are growing exponentially, with no end in sight yet. Scaling these websites to retain availability becomes a necessity, because companies rely on their website. Scaling static pages is a relatively easy task, which mainly consists of buying more hardware. Scaling dynamic pages requires a bit more thought to serve the dynamic data fast and consistently. A bigger challenge is to scale hyperdynamic sites, such as social networking sites. One page view can take several hundreds of internal data requests to retrieve the latest information from "friends", which all needs to happen in real time. These networks are generally hard to divide in segments so there is no good partitioning algorithm to use when scaling these sites. A new technique which is uprising is cloud computing. Unfortunately some mayor drawbacks regarding legal issues and steep learning curves prevent the majority of the developers of going in this direction.
The goal of this paper is to provide general information regarding different scaling techniques, as well as an approach to determine which technique is the most appropriate for a certain website.

Index Terms—Scalability, replication, synchronisation, availability, web application.

1 Introduction

Twitter, *Facebook*, *MySpace*, *Google*, they are all large, famous players on the Internet market nowadays. They all provide their services to millions of users worldwide. They started as small hobby or study projects and went through an immense growth. The little home server at the attic did not suffice anymore, so more and more servers were added, leading to the huge server farms used nowadays. But how do these companies make sure these servers work together, communicate with each other and all provide the same data, pretending for the user to be just one single server.

In this paper we will research various methods to scale up a growing website. Scaling is more than just adding more hardware to your server farm. Sometimes smarter techniques are available, to make the hardware operate more efficiently. Sometimes adding more servers requires to think about how to distribute and synchronise the data among the different servers.

As the applicable scaling techniques depend on the profile of a website, we will make a distinction between three different profiles: static (Section 5), dynamic (Section 6) and the superlative case: hyperdynamic (Section 7) websites. For each profile, we will show which forces apply and which techniques are used to optimise performance and availability of the website. As the hyperdynamic websites are upcoming currently, a lot of interesting techniques are introduced in this area.

After having described the techniques which are already used nowadays, we will look into the currently ongoing development in Section 8. Several techniques are developed to make sure websites can continue their growth.

In the evaluation (Section 9) we will provide a summary of all the techniques and the website profiles they are applicable to.

2 What is Availability?

The lexical meaning of the word 'available' is *"obtainable or accessible and ready for use or service"*, according to [10]. For websites it means the percentage of time that the service is functioning.

- *Yuri Meiburg is MSc. Computing Science student (Computational Science and Visualisation) at the University of Groningen, e-mail: Y.Meiburg@student.rug.nl*
- *Allard Naber is MSc. Computing Science student (Software Engineering and Distributed Systems) at the University of Groningen, e-mail: A.L.Naber@student.rug.nl*

The availability of a system is usually expressed in "number of nines". Four nines then means the system is available 99.99% of the time.

2.1 Why is availability needed?

The provider of a web service usually benefits from the service being available to users. Several business models can apply. Commercial websites offer the potential customer to get to know a company, or to buy a product online. An unavailable website may lead the customer to another company, resulting in a loss of profit. The newer web applications are more focused on social networking. These websites are not directly linked to commercial activities, but usually gain profit out of advertising. To make advertising profitable a lot of active users are needed, but to attract these users the application must "always" be available.

3 Classification of techniques

Several techniques to improve the availability of websites have been researched in the past. We will provide an overview of some of the techniques. For this overview we first distinguish client-side and server-side techniques. Client-side techniques are applied by internet service providers (ISPs) or employers in a business environment to improve the speed experience for its subscribers or employees. Server-side techniques are techniques which are applied by web services on their own servers to keep up the speed of their service and to prevent their servers from denial of service.

To review the several server-side techniques we consider three different profiles for websites: static, dynamic and hyperdynamic pages. Static pages change almost never, dynamic web sites are changed regularly while hyperdynamic sites are continuously updated. This distinction has been selected because it provides an incremental order for the use of the several techniques. Every profile also uses the techniques from the previous profiles.

4 Client-side techniques

The first performance optimisations can take place on the client-side. This already takes some load away from the web servers and immediately improves the user experience for a lot of different users.

Most internet service providers (ISP) today incorporate some form of caching. This can be performed with software such as Harvest [3] or Squid [18]. These programs implement what is called the Hyper Text Caching Protocol (HTCP, [17]). This protocol provides a way to find cached instances of a page amongst other servers. If it finds such a page, it places a "conditional GET" request through HTTP. A "conditional GET" requests a page from a server, which is only sent if it actually differs from the cached version. This is done by adding the

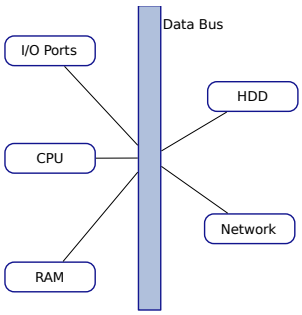


Fig. 1: Schematic of the most basic server architecture.

Fig. 2: 1U Server mount, demonstrating the slim set up used for servers.[1]

'If-Modified-Since' header (for specifications of the HTTP/1.1 protocol see [14]. If the page is modified since the cached page was generated, the "conditional GET" works just like a regular GET request. If it was not modified, the web server will respond with HTTP response code "304 - Not Modified". This will relieve the web servers in most cases, since a page is usually more often visited than it is revised.

Iyer et al. [8] propose a new method to find cached versions of a web page, using decentralized peer-to-peer searches. Using as little as 10MB cache size per node in the network already shows a significant drop in the total external bandwidth.

5 Static pages

A static page is a page which does not require any further processing before it can be sent to the viewer. These pages are the most basic type of pages and they require the least of the server. This makes it ideal for use in websites which do not often change content. We will discuss server-side techniques to keep web sites like this available under heavy use.

5.1 Single Server Architecture

The most basic hardware set that is needed to run a website consists of a motherboard, a hard disk, a processor and some memory (optionally a graphics card to configure the server). This combination results in a setup similar to Figure 1. Note that the data bus is the connection between all the components in the system and is located on the motherboard.

The first bottleneck that will arise is the hard disk, since each request has to be read from the disk and requires little memory and little processing power. Every element (such as an image) on a web page is processed using a separate request, so each page request requires multiple data requests to the hard disk. If we also take into account that the average seek time of a hard disk lies around $10 \sim 15$ ms [19], it becomes obvious that this puts quite some stress on the hard disk.

5.1.1 Data Storage

In the late '80s it was proposed to use *"Redundant Arrays of Inexpensive Disks"* (RAID) [11]. Rather than improving the performance of a single disk, Patterson et al. suggest the use of multiple disks in parallel. While there already existed something called *"Just A Bunch Of Disks"* (JBOD), which basically merges two or more disks by "aligning" them and writing to the first one that has space available, RAID poses various advantages over JBOD. RAID allows for parallel usage of the hard disk, and redundant data storage, to overcome the unreliability of hard disks. As it is unacceptable for a web host to lose data due to a disk failure, it is standard for web servers to accommodate some form of RAID. Implementing a RAID construction in the architecture requires a controller card, which usually communicates through PCI.

5.1.2 Server Architecture

It is possible that the optimized data storage is still insufficient to attain the desired availability. The next step in such cases is usually switching from normal hardware to server hardware. A server motherboard has several advantages over ordinary motherboards. A server motherboard contains a faster bus, thus allowing more data to be sent back and forth between the processors and other hardware, such as hard disks. Besides a bigger bus, it usually also supports multiple processors. Most of these motherboards support 4 processors, where each processor can contain up to 4 cores. This results in a total of 16 cores per motherboard. Another advantage is that it has support for special types of memory, so called "ECC Memory". ECC stands for "Error Correction Codes" and is a method used to detect and correct errors introduced during storage or transmission of data. Because servers are in operation 24 hours a day, they have a much higher likelihood of generating an error in RAM memory, thus error detection and correction is a necessity. Furthermore most server motherboards also have native support for RAID configurations, which means that the data storage optimization as mentioned in Section 5.1.1 can be performed even closer to the processors and is thus faster.

Another advantage which might be less obvious is that a server is extra slim, in order to put it in so called Server Mounts (see Figure 2). These server mounts can easily be mounted in towers, which are basically racks of server mounts.

5.2 Multiple Server Architecture

When one server does not suffice anymore, the next step is to add more hardware in the form of more servers. Then through the use of a so called "load balancer", these servers will work parallel. This results in the ability to process more requests at the same time. A load balancer is little more than a server to which all incoming requests go, but rather than processing all requests it forwards the requests to the actual servers which in turn process the request. Load balancing can happen on different levels, and there are various techniques. The simplest technique is a so called "Round Robin" method, which simply iterates over the list of web servers. More sophisticated methods take the current load of a server in to account, and can even base their decision on the shortest distance from server to client (in case of distributed server centers). A typical architecture is shown in Figure 3.

5.2.1 Extreme Scaling

Should one encounter the case where even the use of a load balancer is not enough, it is possible to scale one step further. It is possible to configure load balancing on the DNS level. DNS stands for "Domain Name System", which is a hierarchical naming system for computers and services. An analogy to a DNS server would be a phone book. When a request is sent for a particular website, a DNS server responds with an address which corresponds to another DNS server which knows more about where to find the particular server. This continues until a DNS server is reached which knows the exact IP address of the server responsible for the requested URL. However, a

[1] Source: http://www.stealthcomputer.com

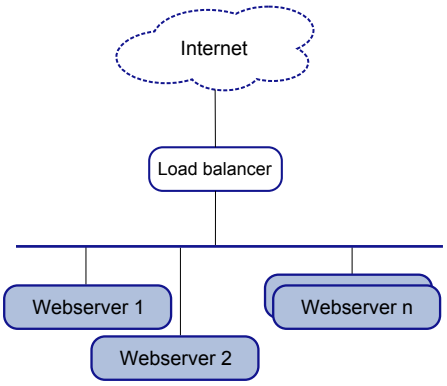


Fig. 3: A typical architecture for web servers, using a load balancer.

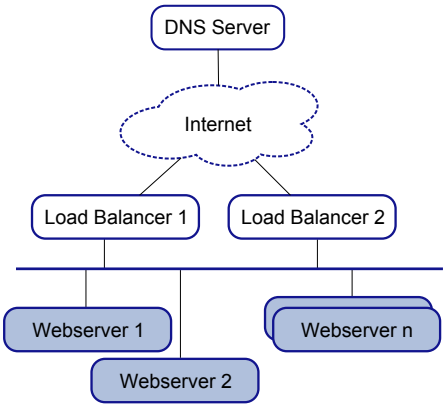


Fig. 4: Load balancing on DNS level, where one website uses multiple load balancers, which in turn lead to multiple servers.

DNS server is also capable of storing multiple servers responsible for a URL. This means that it is possible to configure multiple load balancers which are responsible for the same servers. The DNS servers will distribute the load over the load balancers, which can reroute the requests to the servers which will process the requests. The schematic representation of this is shown in Figure 4.

6 Dynamic pages

The next set of server-side techniques are the ones for dynamic web sites. Note that also the techniques for the static pages are applicable to these types of sites. Dynamic web sites are updated regularly, by users or by administrators. This poses some extra limitations on the way data can be replicated to multiple servers. However, if a change takes a while to be cascaded to all clients, it is generally not an immediate problem.

The problems which need to be solved for this type of websites are load balancing, data replication and synchronization.

6.1 Server-side caching

Making use of server-side caching can dramatically improve the performance of a web application which requires much computational power to render a single page. If this page is viewed more often than it is changed, the results of the rendering can be stored on the web servers. The next time the page is requested, the pre-rendered page can be served immediately. An illustration of these steps is shown in Figure 5. This approach requires that the web server does have access to the most recent data. The issue yet to be solved is when to update the cache.

A simple approach is to refresh the page based on a maximum lifetime. If the cached version of the page is older than the specified maximum time, the cache is refreshed. This might result in showing a page that is outdated, but it is an easy approach which does not require additional communication between refresh actions.

If serving outdated pages is unacceptable, it is possible to check whether the cache should be refreshed, each time a request is done. This requires to compare only two timestamps, so it does not have too much impact on the performance but it will guarantee the page is up to date.

The third approach is to update the cached pages when a change is made. However, most of the times it is harder to incorporate this in the architecture of a web application as caching and updating are distinct features of a web application.

6.2 Load balancing with multiple servers

The main principle for load balancing for dynamic websites is analogous to the situation for static pages, as described in Section 5.2. An extra issue is that users provide data on each of the servers. This implies extra effort is required to keep data consistent among the servers.

A lot of applications use a database to store their (user) data. An advantage of using a database system is that data is easily modifiable, better than when using a file system to store data. On top of that, a lot of database systems are designed to support load balancing using multiple servers. Several synchronisation strategies are available, which will be considered in the following sections [1, 16].

6.2.1 Updating copies

The first strategy uses a master server, together with a few slave servers, all having the same data initially. The master server is expected to always have the correct data. When a data modification occurs on one of the slave servers, it notifies the master server of this change. The master server in turn sends the new data to all slave servers asynchronously. While this process is running the functionality of slave servers is limited, they can only serve read-only queries, as the data could become inconsistent otherwise.

6.2.2 Invalidation of data

Another approach is to still consider the idea of a master with multiple slave servers, but the slave servers initially have no data. When data is requested, the slave server will contact the master server to retrieve the requested data and keeps a local copy. By performing multiple data requests, the slave server is populated with data. When data is modified, the master server will send an invalidation message to all slave servers. The slaves mark the specified data object as invalid, or out-dated, but take no further action. Once an invalid data object is requested, the slave server will retrieve the newest version of the object from the master server.

This approach is in general easier and more lightweight than the "updating copies"-strategy, because invalidation messages are smaller than the complete data objects. Additionally, it saves overhead in case

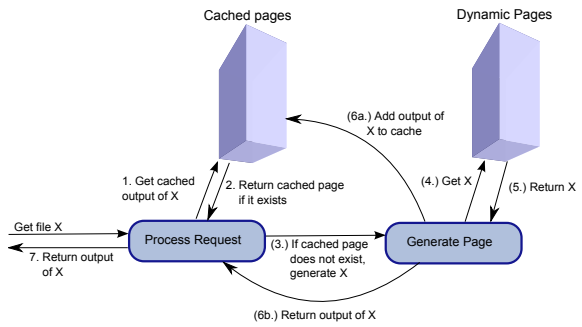


Fig. 5: Step by step explanation of server side caching.

a lot of slaves use a small set of data objects. These slaves do not need to have all the data objects, but just the one they actually use.

7 Hyperdynamic pages

The third class of websites is called hyperdynamic. These websites are heavily updated and viewed by a lot of users. Think of social networks, like *Facebook*, *Twitter* and the like. Caching like described in Section 6.1 is not possible here, as every page is rendered entirely based on user data. Each page contains information from all users linked to the current users, their so-called friends, or contacts. The number of friends per user varies from about ten to hundreds or even thousands, in the case of *Twitter*.

These websites are relatively new and the type of use makes that traditional methodologies for increasing performance do not always work well. A lot of new concepts are developed to keep these networks up and running. In the next sections we will review the way large networks handle all the traffic and highlight the several projects with originate from this effort.

The heavy user based pages are not the only problem for these social networks, another challenge is the rapid growth of these networks. In Figure 6 it is shown that the number of *Twitter* messages has grown exponentially over the last few years. The same holds for the number of users. This growth is also seen at the other applications. In this section we consider several projects which are to solve the posed issues.

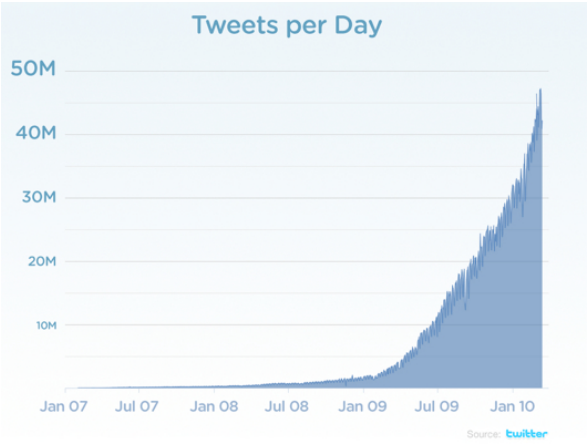


Fig. 6: Timeline showing the number of *Twitter* messages over time.[2]

7.1 Distributed memory cache

Most hyperdynamic web services retrieve data from databases. Under a heavy load, the stress on the database servers will become too high. One solution to relieve the database servers is to cache the most used data on the client side, in the memory of a number of servers. These caching servers are supplementary to each other to create a bigger cache. The concept and an implementation called 'Memcached' have been developed by the social networking site *LiveJournal* [5]. Currently they are running 28 Memcached instances with a total cache size of 30 GB. The cache hit rate achieved is 92%. These facts illustrate that by using a relatively small cache, a reasonable amount of load can be taken off the database servers.

7.1.1 Method

The complete cache can be viewed as a big hash-table: keys and values stored in a bucket, identified by a hash of the key. However, in the case of a distributed memory cache, the cache more resembles a layered hash-table: the request of a key-value pair requires two steps, as

[2]Source: `http://www.twitter.com/`

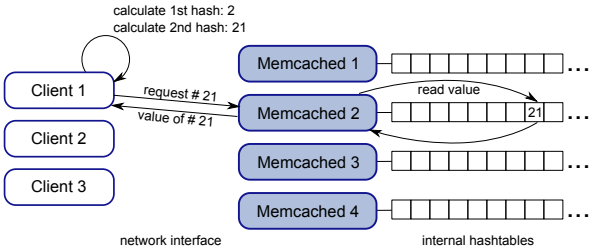


Fig. 7: Process of retrieving a key-value pair from the distributed memory cache.

these pairs are distributed over multiple Memcached instances. By calculating the hash of a key the client determines which instance holds the requested data, if it is available in the cache. This is comparable to the top-layer bucket in the hash-table. The client then connects to that instance to request the actual key-value pair, which resides in the second-layer bucket.

Figure 7 shows the process of retrieving a key-value pair for which the first hash calculates to 2 and the second hash calculates to 21.

7.2 Distributed Storage System

One of the side issues of hyperdynamic websites is the involvement of large amounts of data. To maintain quick writing and reading times it is not possible for one server to manage all data, meaning that the data has to be distributed over multiple servers. This raises the problem of how to manage the data. In 1990 Popek et al. [12] proposed "Ficus", a distributed file system which uses replication of data across servers at the expense of consistency. Ghemawat et al. [6] proposed the "Google File System" in 2003, which is widely deployed within Google. In 2003 this file system managed multiple hundreds of terabytes data, across thousands of disks. The Google File System was designed with the failure of hardware as a standard, rather than an exception. This makes the system very robust against hardware failures. It is designed to work on inexpensive commodity components. One of the features of this filesystem is that it is heavily optimised for large files (multiple gigabytes being common), but there is little optimisation for small files. Another design decision is the favouring of high sustained bandwidth over a low latency. These design decisions might make a large difference in performance, but only for a small number of hyperdynamic websites.

In 2009, Lakshman and Malik [9] proposed a decentralized structured storage system named "Cassandra". Just as the Google File System, Cassandra is developed to be deployed on hundreds of nodes (even spread across different data centers), and takes hardware failure as a standard as well. Cassandra provides a structured key-value store with eventual consistency. It does not offer "eager consistency", even though that makes a better environment for programmers, because it reduces update performance and increases transaction response times [7]. An important aspect of Cassandra is that it is designed to handle high write throughput, while not sacrificing read efficiency. Testing the performance of Cassandra was done with Facebook data. Using ~50TB of data and a Cassandra cluster consisting of 150 nodes, finding a search term usually takes between 15–20 ms.

8 Future techniques

As the social networking sites are still growing, they all continue their research on improving the availability of their services. In this section we will consider some of the techniques that are currently being researched.

8.1 Script compiling

Although we have shown various techniques which are used to scale websites, another level of scaling is becoming more popular: Further optimization of underlying software. There are various programs

which try to speed up PHP[3]. For instance "Zend Server™" [20], which tries to optimize the code using "operation codes" (or opcodes), and caching. Some developers take it even further and write translators to convert PHP to some other language. E.g. "Road Send" [13] and "phc"[4] convert PHP to C code, "Quercus™" [2] translates PHP to Java, and "Phalanger" [15] converts PHP to Python. The fact that all of these software packages have their disadvantages shows from a recent development from *Facebook*, called "Hip-Hop".[4] "Hip-Hop" converts PHP to compiled C++ code, which provides an average speed-up of 50%. Hip-Hop has only undergone 6 months of development so far, and with such results like these it means that there is a lot of work left to do regarding optimization on this level.

8.2 Cloud computing

Another approach to scaling websites which is emerging is so called "cloud computing". This technique differs from the previous techniques because the user does (usually) not own any of the hardware anymore. Rather than paying for the hardware, a user must now pay for using hardware. The hardware itself is provided by an external company, such as *Amazon* with their "Elastic Compute Cloud" (EC2).[5] This has the advantage that there is no need for a mayor investment, because the only cost is the price of the hardware used for a certain amount of time. It also means that security is managed (up to a certain point) by the company offering cloud services. Unfortunately, the inability to arrange where files are located in the cloud poses some serious legal issues. Cloud servers can be located anywhere in the world, thus content which might be legal in one country might be located on a server which is located in a country where it is illegal. This and the steep learning curve – it takes a whole new design approach to scale cloud services – make it promising for the future, if these issues are addressed.

9 Evaluation

The most important aspect of scaling a website is determining which approach is best suitable for a specific website. Although all websites benefit from a hyperdynamic structure, maintaining this structure is harder and the initial set-up costs are higher than applying basic scaling techniques. Figure 8 shows a feature list, along with examples, demonstrating that for most purposes scaling on a dynamic level will suffice. They key to scaling dynamic pages is to try and reduce the dynamicity such that at all times the webserver contains a cached version, essentially making it a static website as far as scaling is concerned.

Hyperdynamic scaling is only necessary when all data is user-generated and user-maintained. Combined with user based views this makes caching of pages practically impossible, thus a more sophisticated approach (such as described in Section 7) needs to be implemented.

10 Discussion

With this paper we provided an overview of several available scaling techniques for web applications. We first considered some general techniques for optimising server hardware. These techniques always apply, as this will be beneficial for every type of web application. Thereafter we considered the three website profiles together with the scaling techniques. It became clear that especially in the field of hyperdynamic web applications research should be done. The time of static web pages is over, so doing research focused on these websites is possibly a loss of time. The dynamic web pages occur a lot on the internet and they will continue to exist. A lot of companies, associations and educational institutes use them. Several scaling techniques are available and they seem to suffice so far.

The real challenge is in the field of hyperdynamic pages, where a lot of data exists and partitioning of this data is hard or sometimes even impossible. The current trend is that after some basic synchronisation techniques have been invented, the environment is optimised. Frameworks or external applications, like language interpreters or database systems, are being rewritten to increase performance. This process is now heavily going, but is likely to end. At a certain moment, these applications are almost optimal and there is nothing more to gain. However, the growth of websites or communities will continue, so a next step should be made. Where this next step will lead is, is subject of future research.

Features	Type	Examples
Techniques		
All pages are pre-generated No user-input Low frequency of content changes	**Static**	Simple personal website Help Files Pre-generated image gallery
5.1.1 RAID 5.1.2 Server architecture 5.2 Load balancing using load balancers 5.2.1 Load balancing on DNS level		
Administrators and users add data Frequent content changes User-indepent views	**Dynamic**	Blog Corporate website Online auction Advanced personal website Dynamic image gallery
6.1 Server-side caching 6.2 Advanced load balancing (with synchronisation)		
Users create (almost) all data Users are interconnected High popularity (>100 hits per second) All data is active all the time	**Hyperdynamic**	Social Networking Online Bookmarking Online Gaming Online Dating
7.1 Distributed memory cache 7.2 Distributed file storage		

Fig. 8: Feature list of the different website types.

References

[1] Bal, H., Kaashoek, M., Jansen, J., and Tanenbaum, A. Replication techniques for speeding up parallel applications on distributed systems. *Concurrency Practice and Experience 4*, 5 (1992), 337–355.

[2] Caucho Technology, I. Quercus™.

[3] Chankhunthod, A., Danzig, P. B., Neerdaels, C., Schwartz, M. F., and Worrell, K. J. A hierarchical internet object cache. In *ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 1996), USENIX Association, pp. 13–13.

[4] de Vries, E., Gilbert, J., and Biggar, P. PHC.

[5] Fitzpatrick, B. Distributed Caching with Memcached. *Linux Journal* (August 2004).

[6] Ghemawat, S., Gobioff, H., and Leung, S.-T. The google file system. *SIGOPS Oper. Syst. Rev. 37*, 5 (2003), 29–43.

[7] Gray, J., Helland, P., O'Neil, P., and Shasha, D. The dangers of replication and a solution. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1996), ACM, pp. 173–182.

[8] Iyer, S., Rowstron, A., and Druschel, P. Squirrel: a decentralized peer-to-peer web cache. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing* (New York, NY, USA, 2002), ACM, pp. 213–222.

[9] Lakshman, A., and Malik, P. Cassandra: structured storage system on a p2p network. In *PODC '09: Proceedings of the 28th ACM symposium on Principles of distributed computing* (New York, NY, USA, 2009), ACM, pp. 5–5.

[10] Miller, G. A. WordNet – About Us. http://wordnet.princeton.edu, 2009.

[11] Patterson, D. A., Gibson, G., and Katz, R. H. A case for redundant arrays of inexpensive disks (RAID). In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1988), ACM, pp. 109–116.

[12] Popek, G. J., Guy, R. G., Page, J. T. W., and Heidemann, J. S. Replication in Ficus distributed file systems. In *Proceedings of the Work-*

[3] PHP Hypertext Preprocessor, website: http://www.php.net/

[4] Facebook url:`http://www.facebook.com`, Hip-Hop is open-source, and available from:`http://github.com/facebook/hiphop-php`

[5] EC2 is available at: `http://aws.amazon.com/ec2/`

shop on Management of Replicated Data (November 1990), University of California, Los Angeles, IEEE, pp. 20–25.

[13] ROAD SEND INC. Road Send PHP.

[14] THE INTERNET SOCIETY. Hypertext Transfer Protocol – HTTP/1.1, June 1999.

[15] THE PHALANGER TEAM. Phalanger.

[16] THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL 8.2 Manual*, 2006.

[17] VIXIE, P., AND WESSELS, D. Hyper Text Caching Protocol (HTCP/0.0), 2000.

[18] WESSELS, D. ICP and the Squid Web Cache. *IEEE Journal on Selected Areas in Communication 16* (1998), 345–357.

[19] ZEDLEWSKI, J., SOBTI, S., GARG, N., ZHENG, F., KRISHNAMURTHY, A., AND WANG, R. Modeling hard-disk power consumption. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2003), USENIX Association, pp. 217–230.

[20] ZEND TECHNOLOGIES LTD. Zend server.

Does Architectural Knowledge Management Forget People?

Dan Tofan, *PhD Student, Groningen University*

Abstract— Documenting the software architecture of a system helps communication among stakeholders, by capturing the early important design decisions and preserving them for subsequent use. The lack of proper documentation increases the risk of knowledge vaporization, because the initial rationales for various design decisions are gradually lost, or the people that worked on a specific project leave the company. To address this issue, the field of Architectural Knowledge Management proposes various approaches and tools.

In this paper, we review some basic concepts of AKM, from a few important articles in the field. Next, we go into the benefits of providing knowledge management support to developers and software architects, along with the various activities of the architecting process and the involved actors. Then we examine some existing representations of architectural knowledge and their evolution over time. In addition, we look at the relevant tools and technologies for managing such knowledge, which consists mainly of the design of the software systems and the important decisions involved in creating it.

Our contribution is to consider some perspectives on knowledge, which take into account its human specific nature. We attempt to show that such view, inspired from the field of knowledge management at large, can benefit AKM.

Index Terms—Software architecture, architecture knowledge, architecture knowledge management, information

1 Introduction

One of the widely accepted definitions of the software architecture of a system is "the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them" [1]. In addition to helping communication among stakeholders, the software architecture of a system can be later inspected and reused for new projects. The software architecture is largely seen as a set of early and important design decisions. Documenting software architecture is done by describing various aspects of the system through architectural views, each offering a perspective that is relevant to its intended audience. An example of such set of views is the '4+1' one, part of the Rationale Unified Process methodology.

Changing a software system throughout its lifetime is inevitable; however studies show that it can be very expensive to do that, especially when such modifications have architectural impact. Such costs can be reduced by capturing the knowledge and information regarding the outcomes of domain analysis, the various architectural styles and patterns used and the other design decisions. Bosch [2] argues that failing to do so results in the phenomenon of 'knowledge vaporization'. By regarding software architecture as a set of design decisions, and capturing such knowledge, the impact of this phenomenon can be greatly reduced [3]. Furthermore, Kruchten et al. [4] define architectural knowledge (AK) as the sum of Design Decisions and the Design itself. Efficient use of AK implies the need for its management, which can be assisted by tools and technologies.

The next section presents the categories and main views of architectural knowledge. Section 3 outlines the role of knowledge management in software architecting. Section 4 summarizes the state of the art in representing architectural knowledge, and section 5 presents tools and technologies that support AKM. Section 6 discusses some perspectives on knowledge and their potential impact on AKM and we draw conclusions in section 7.

2 Basic concepts of AKM

There is no widely accepted definition of Architectural Knowledge, as outlined by Farenhorst and de Boer [5], who conducted a comprehensive literature review, attempting to clarify what it entails. They identified four important views on it, each with its importance and applicability, which we will briefly mention.

1. The pattern-centric view focuses on the use of design patterns, as a means of reusable solutions applicable to recurring problems, and on forming a common vocabulary that eases communication among developers, thus facilitating the sharing of architectural knowledge. An important aspect of patterns is their suitability for human consumption, and lack of it for automated tools.
2. The dynamism-centric view is a more formal approach to architectural knowledge that uses graph-based approaches for architectural reconfiguration of dynamic software systems. Non-human agents should easily consume such knowledge, which is preloaded in the software application itself.
3. The requirements-centric view is rooted in the strong relation between architecture and requirements. The architectural knowledge enables traceability between them.
4. The decision-centric view marks a shift from documenting the end result of the architecture of a system, to recording the rationale behind it. The architect needs to balance the concerns of the stakeholders, justify and communicate the design decisions to them.

The recent definitions of architectural knowledge in the literature use mostly the decision-centric view, while regarding the other ones as ultimately equivalent to it.

To further understand the different manifestations of architectural knowledge, we can use the distinction between tacit and explicit knowledge, credited to the work of Nonaka and Takeuchi. Generally speaking, tacit knowledge refers to cognitive entities that are difficult to transfer to another person by writing or verbalizing, like the ability to use complex equipment, learning a language or riding a bike. In contrast to it, explicit knowledge can be articulated, codified, and stored in repositories, documents or any other proper media. Applying this idea to architectural knowledge allows us to distinct between tacit AK, based on experience and expertise, in contrast with the explicit one, captured in documentation, like various technical characteristics, or decisions.

Lago and Avgeriou (quoted in [5]) proposed the distinction between application-generic and application-specific architectural knowledge. The former is "a form of library knowledge", applicable in multiple applications, independent of the domain, while the latter involves all the decisions taken during architecting a particular software system. In Figure 1, the combination between application and architectural knowledge is presented.

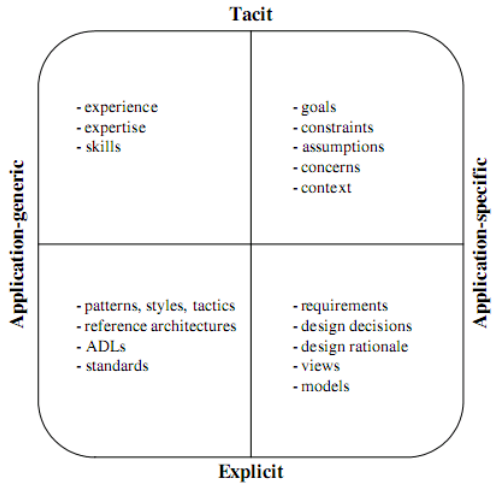


Fig. 1. Architectural knowledge categories.

The four categories have the following characteristics:

1. Application-generic tacit AK includes knowledge gained from experience, like architectural methodologies, concepts and internalized solutions.
2. Application-specific tacit AK is about contextual domain knowledge regarding factors influencing an actual architectural solution, like stakeholder concerns, business goals, and the application context.
3. Application-generic explicit AK refers to design knowledge captured in books, standards, discussions or other types of communication.
4. Application-specific explicit AK includes all externalized knowledge of a certain system, like documentation on the architectural views of that system, models used, requirements, specified design decisions and their rationale.

Knowledge from each of the four categories described above can be converted to another category, or even in the same one. Farenhorst and de Boer [5] describe in more detail such conversions, for each of the existing four views on architectural knowledge.

3 How does Knowledge Management help the software architecture process?

After understanding the basic concepts of AKM, the next interesting question is about its actual use in practice. We will use Babar's ideas [6], to support the case for the role of KM in the software architecture process.

First, a suitable model of the architecture life cycle is needed and the one from Figure 2 is used, with the following explanations about the presented activities:

1. *Architectural analysis* targets defining the problems to be solved. The architect examines the various concerns and their context, so that a set of architecturally significant requirements is identified.
2. *Architectural synthesis* aims to design solutions for the identified requirements, by considering multiple available design options and selecting the most appropriate one.
3. *Architectural evaluation* has the role of verifying the fitness of the chosen architectural solutions, against architecturally significant requirements.
4. *Architectural implementation* requires designers and developers to take more decisions on the detailed design and implementation, while ensuring conformance to the already existing architecture.
5. *Architectural maintenance* involves on-going architectural changes, which might be needed for maintenance and evolution purposes. The old design decisions need to be reassessed for possible impact, and new decisions can be made for satisfying the new requirements, while preserving the architectural integrity of the system.

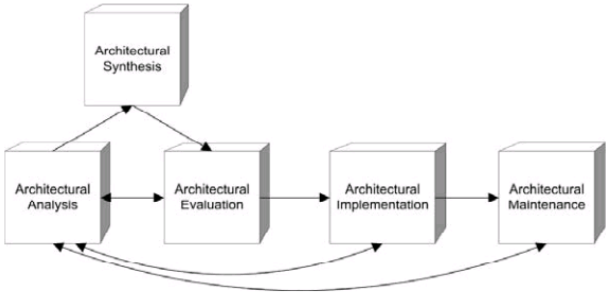


Fig. 2. Model of the architecture life cycle from [7].

Each of the architectural activities presented above has specific AK needs, which we summarize below:

1. *Architectural analysis* needs knowledge about business goals, organizational processes, scenarios, guidelines for workshops, and similar systems that have been developed.
2. *Architectural synthesis* benefits from knowledge about generic design options, architectural styles and patterns, tactics, rationale on older design decisions and existing or future systems to be integrated.
3. *Architectural evaluation* is helped by knowledge on the rationale for design decisions, reasoning framework, patterns, tasks to be performed, evaluation process models and methods, procedures and standards for recording evaluation findings.
4. *Architectural implementation* needs reasoning knowledge for understanding the architecture design, implementation standards, strength and weaknesses of implementation frameworks.
5. *Architectural maintenance* benefits from the above types of knowledge, in addition it needs information on the justification for the changes and impact analysis techniques.

The activities mentioned above do not form a sequential process, but they are performed as iterations. Tasks related to a particular activity may be revisited while performing any other activity. Moreover, various other stakeholders can join the architect in performing the tasks associated with each activity. For example, architectural maintenance involves developers, designers and architects, who need to make detailed design and implementation decisions. A common characteristic of all activities is their knowledge-intensive nature. Because of their role in the software architecture process, it is important to manage such knowledge.

4 Representing Architectural Knowledge

A practical aspect of managing architectural knowledge is to focus on displaying it, so that it can be easily stored, communicated, and reused. Using Kruchten's paper [8], we briefly present the evolution of architectural knowledge representations, from the very intuitive and informal, to more abstract and formal notations. It is important to notice that only the lower half of Figure 1 is considered, that is the explicit part of knowledge, with more attention to the application-specific and less on the generic one. Also we use the definition of architectural knowledge as design plus design decisions [4], so we review the representation of each of them.

4.1 Representing Architectural Design

During 1970s and most of 1980s, software architecture was mixed with software or high-level design, and little agreement existed on

how to document the architecture of a system. Mixes of boxes and arrows were used, with unclear meaning and little rigorousness. This form still persists today in the so-called PowerPoint level of documentation, which is valuable for communication with less technical stakeholders, like marketing persons, sponsors or project managers.

During late 1980s, software architecture gained a clearer identity, summarized in a simple formula from Perry and Wolf: Architecture = {Elements, Form, and Rationale}. The elements can be described in terms of components and connectors, with the desired non-functional properties of the system shaping the form of the system. The architect takes various design decisions, based on rationing on the various options.

From early 1990s, the architecture of complex systems started to be regarded as an entanglement of structures, and that the use of a single type of blueprint is not realistic. The problem was later presented by the authors of [9], in terms of how can somebody describe the wing of a bird? One can consider it as a collection of feathers, while somebody else may be interested in its movement dynamics, or maybe in the blood circulation inside the wing, the point is that each aspect is of interest to different groups of people, and the same is the case for the software architecture of a system.

Fig. 3. Picture on the cover of [9], how can one describe the wing of a bird?

By recognizing that different stakeholders are concerned with different aspects of the architecture of a system, various views can be introduced, each presenting an abstraction or simplification of a more complex reality. A so-called *viewpoint* consists of a set of conventions for the construction, interpretation and use of a given view. Intuitively, a viewpoint is to a view what the legend is to a map. One of the most popular set of views is shown in Figure 4, where we can see the target audience of each view, while the Scenarios view contains use cases that glue together the rest of them.

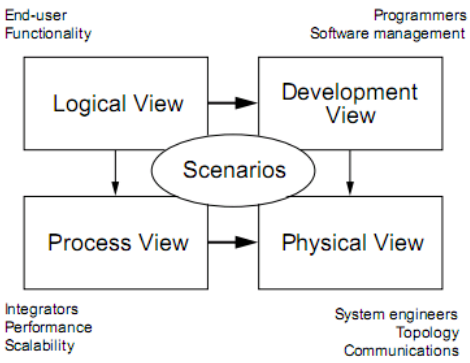


Fig. 4. Rationale Unified Process' 4 + 1 views, from [10].

From 2000s, it is important to mention the first formal standard for architectural description, IEEE Std 1471, which became an international standard in 2007. Currently it evolves under a joint revision by ISO and IEEE, under the name ISO/IEC 42010, "Systems and Software Engineering – Architecture Description". It tries to overcome challenges like view correspondences for linking between views, and increasing reusability among architecture frameworks, enabling architects to work easier with multiple paradigms.

A more formal approach was to create architecture description languages (ADLs), for capturing, representing and reasoning about the essential components of a software system. ADLs offer textual and graphical notations, for human and machine consumption, with various degrees of success. Most relevant such languages are Rapide (Stanford), ACME (CMU), Wright (CMU), C2 (UCI), Darwin (Imperial College), and Koala (Philips). Almost all of them have seen little success outside the academic world, and the only one that has seen large use is the Unified Modeling Language, which is the winner of the so-called software modeling languages wars, with a large industrial use, adoption rate and available tools.

4.2 Representing Design Decisions

Since 2003, design decisions have received increased attention from the software architecture community, due to their role in specifying the structure of a system. Here are the most important attributes of an architectural design decision, from [8]:

1. The *Epitome* is a short textual description of the decision itself.
2. The *Rationale* is the justification for a decision, possibly pointing to external sources, but paying attention not to repeat information from other attributes.
3. The *Scope* delimits the part of the system, life cycle or part of the organization to which the decision applies.
4. The *State* contains the current condition of the decision, like: idea, tentative, decided, approved, challenged, rejected or obsolesced.
5. The *Author* also has information on timestamp and history of the decision, for traceability purposes.
6. The *Categories* provide a useful way of grouping similar decisions, based on specific concerns or quality attributes.
7. The *Cost* contains the associated price that some design decisions may have, and which may be useful when reasoning about alternatives.
8. The *Risk* is related to the uncertainties in the problem domain, novelties and immaturity of solutions, and other unknown factors.
9. The *Related Decisions* attribute contains possible relationships between decisions like constraints, enablers, conflicts, alternatives and others.
10. The *Relationship* with External Artifacts refers to the traceability to upstream technical artifacts like requirements, and downstream ones like design and implementation elements.

After understanding the attributes of design decisions themselves, we can notice that they play different roles in the architecting process, and that some are general properties or constraints, or that others are linked to a more general context of the software system. Design decisions are classified as following:

1. *Existence* decisions or "ontocrises" determine that some element will exist in the design of the system. Such decisions are mostly captured for their subsequent interaction with more subtle decisions and their alternatives.
2. *Bans* decisions or "anticrises" are the opposite of the previous ones, as they specifically state that some specific artifact will not exist in the design. Such decisions mark the elimination of certain alternatives.
3. *Property* decisions or "diacrises" specify an enduring quality of the system, in the form of design rules, guidelines or constraints. They are useful for tracing decisions on specific characteristics of the software systems.

4. *Executive* decisions or "pericrises" are not directly related to the design elements or their qualities, but to the business environment, development process, organizational context and choices of tools and technologies.

The visualization of the sets of design decisions can be done by simply using an Excel spreadsheet, but there is little benefit in that, because of low readability. Another possibility is to display design decisions as graphs, and perform various operations on them, like filtering, focusing or sequencing. Another approach is to embed a decision view in the '4+1' views themselves, proposed in [11] and summarized in Figure 5. The envisioned advantage is to capture the design rationale that underlies and motivates the selection of the design options [8].

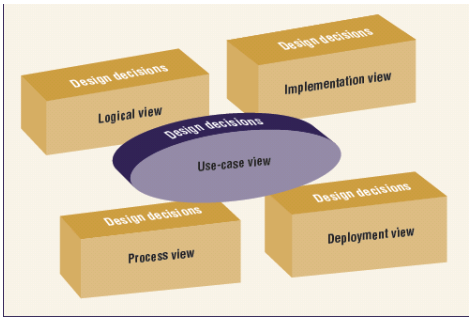


Fig. 5. Decision View Embedded in the 4 + 1 views, from [11].

Representing architectural knowledge enables the analysis, evaluation, implementation and the future evolution of the software architecture of a system. However, various tools and technologies contribute to the representation of architectural knowledge, and we present them in the next section.

5 Tools and Technologies that Help AKM

Liang and Avgeriou [13] argue that efficient management of architectural knowledge requires tools and technologies for its support, with the longer-term objective of automating or semi-automating it. However a strategy for AKM needs to be defined beforehand, that will influence the choice of such tools. There are three distinct such strategies: codification, personalization and hybrid, combining the previous two. Figure 6 shows the difference between the personalization and codification strategies, which is given by the importance placed on either the tacit form of knowledge or the formal one.

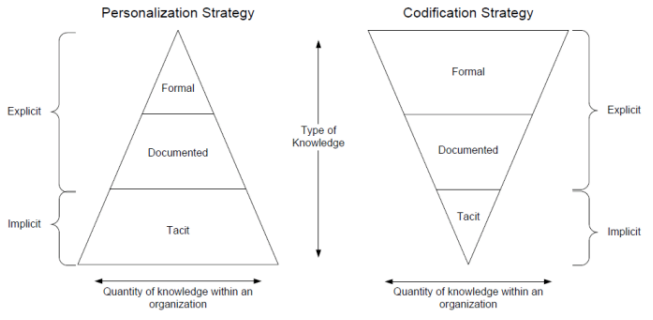


Fig. 6. Pyramid of knowledge types and associated AKM strategies, from [12].

5.1 AKM Use Cases

In order to evaluate the existing tools and technologies, a set of relevant actors and use cases needs to be defined. The targeted audience of AKM tools is made of:

1. *Architects* that design software systems, who are interested in documenting knowledge or retrieving it.
2. *Reviewers* who are interested in judging the quality and progress of an architecture.
3. *Requirements engineers* who are interested to learn more about AK, and the developers involved in implementing and actual architecture.
4. *Maintainers* who evolve a system need to understand previously taken decisions that can be relevant for them.
5. *Users* as the whole set of system stakeholders, in addition to the specialized actors mentioned above.

The use cases for an AKM system are grouped into four categories: consuming AK, producing AK, knowledge management and intelligent support. Each of these categories groups more use cases, covering all the mentioned cases in the literature.

5.2 Tool Support for AKM

The tools supporting AKM cover sets of the possible use cases, each having its own characteristics. We briefly present these tools, starting with the ones helping the codification strategy.

1. SEI-ADWiki is a wiki style of collaborative environment, useful for creating and maintaining architecture documentation in a dynamic manner. It provides editing and version management tools.
2. ADkwik is a Web 2.0 application supporting collaborative decision-making work of software architects, with a structure based on a decision modeling framework, enabling formal AK sharing.
3. ADDSS is a web-based solution for storing and documenting architectural design decisions, and providing traceability between requirements and architecture through decisions. Its strong points are flexibility for decision capturing and the stress on applying general AK like patterns.
4. Archium is an ambitious tool aiming at providing traceability among requirements, decisions, architecture description and implementation. It facilitates maintenance of AK throughout the life cycle of a system, by using its own language.
5. AREL is a UML-based application, focusing on documenting architectural decisions and design rationale. It focuses on uniformly linking design concerns to outcomes, through design decisions.
6. Knowledge Architect is a set of tools for capturing, using, translating, sharing and managing AK, based on a common repository accessed by various clients. It focuses on capturing AK by annotating information from sources like Office documents.
7. SEURAT is an Eclipse plug-in targeted at capturing rationale knowledge in an integrated development environment.

There are also tools for supporting the hybrid strategy, and we summarize them below.

8. EAGLE is an AK sharing portal, implementing best practices from knowledge management for increased AK sharing, by focusing on connecting stakeholders and presenting "who knows and does what".
9. PAKME is a web-based tool aiming to provide KM support, for geographically distributed stakeholders through online collaboration.

As we can see, there are many tools supporting AKM with various approaches and degrees of success. However, most of them are still immature and lack wide adoption by the industry.

5.3 Technology support for AKM

The tools discussed above are based on existing technologies that are widely used in the industry. Here is their list, using [13]:

1. The *Web Portal* is a web application integrating databases, yellow pages, news, documents managers and others, offering the possibility to add easily more content.
2. *Blogs and wikis* are popular editable web pages enabling content contribution and sharing.
3. *Voting and ranking* enable different users to rate content in an online community, thus allowing identifying the experts on a specific topic.
4. *Natural language processing* deals with the understanding of human natural languages by computers, by automatically mining the documentation text, for enriching AK.
5. *Ontologies* are formal structures supporting knowledge representation, management, reusing and sharing, widely use in the field of semantic web. Formal AK can be represented by ontology models.
6. *Plug-ins* consists of programs interacting with a host system for providing a specific functionality. For example, the Knowledge Architect tool uses a Word plug-in for identifying decisions in a document.
7. *Version management* concerns the management of multiple revisions of the same unit of information. Wikis and source code repositories like SVN or CVS are typical examples of this technology.
8. *Web 2.0* brings techniques like tags, context-aware mash-ups, and RSS for producing dynamic web pages that combine AK elements from multiple sources.

Tools and technologies bring an important contribution to AKM, by enabling efficient manipulation of explicit knowledge, as part of codification AKM strategy. Additionally, they facilitate collaboration among stakeholders, under the hybrid strategy.

6 Discussion

We have presented an introduction to Architectural Knowledge Management, but we need to consider that the field itself is a particular instance of Knowledge Management at large. So problems and discussions from it may well resonate with AKM, and help it. We selected a few KM articles, based on their perceived relevance, following a search with Google Scholar.

Galliers and Newell [14] call for a return to fundamentals in the developing of KM systems. Considering a widely accepted definition of knowledge as 'justified true belief', and that belief refers to an individual's or group's idea about what 'truth' is. This implies that 'truth' is always problematic, and that searching for a unique truth is a useless exercise, due to its social construction. In turn, that means 'knowledge' is always contestable and elusive to capture in a software system, however exactly this contestability of knowledge and truth leads to creativity and innovation, which are fundamental attributes of humans. Knowledge does not exist outside of a knower, and it is the result of cognitive processes, taking place in the mind of a person. On the other hand, various tools and technologies, like the ones presented in the previous section, are really efficient in storing and retrieving data. By distinguishing between data, information and knowledge, we can better understand the subtle differences between them.

Data	Information	Knowledge
Explicit	Interpreted	Tacit/embedded
Exploit	Explore	Create
Use	Build/construct	Rebuild/reconstruct
Accept	Confirm	Disconfirm
Follow old recipes	Amend old recipes	Develop new recipes
No learning	Single-loop learning	Double-loop learning
Direction	Communication	Sense-making
Prescriptive	Adaptive	Seminal
Efficiency	Effectiveness	Innovation/redundancy
Predetermined	Constrained	Flexible
Technical systems/networks Context-free	Socio-technical systems/networks Outer context	Social networks Inner context

Fig. 7. Key characteristics of data, information and knowledge, from [14].

The characteristics from Figure 7 imply that the architectural tools for AKM are actually more architectural information management systems, rather than knowledge ones. However, they can clearly facilitate cognitive processes, so we can accept the term AKM tools.

Knowledge implies the presence of a knower, making it subjective and very personal. Is architectural documentation a form of knowledge? We argue that it is only information, but it may turn into architectural knowledge in the head of the one studying it.

In [15], the authors argue for the duality of knowledge, as having a *hard* aspect, equivalent to the explicit knowledge discussed above, that can be articulated, captured and stored, and a *soft* aspect, similar to the tacit one, that cannot be externalized. All knowledge consists of these complementary facets, but in various degrees. Such perspective can help understand the phenomenon of architectural knowledge junkyards, for example it is easy to set up a wiki for capturing AK, but if we do not have an active community to actually populate it, then it will be useless. An active community implies participation, learning and the development of a person's own identity in the relation to that community. Perhaps managing architectural knowledge should not only investigate the tools, but also the social mechanisms that stimulate the sharing of architectural implicit knowledge. The impression we get from the reviewed architectural articles is that the importance of tacit knowledge is not fully recognized.

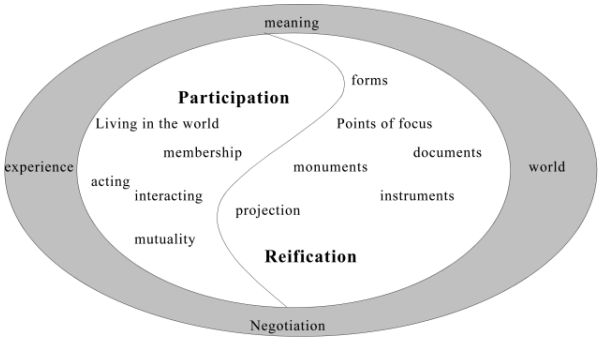


Fig. 8. The duality of participation and reification, from [15].

In Figure 7 a model of the duality between participation and reification is presented, that maps closely on the soft/hard duality of knowledge. Participation is described as "the social experience of living in the world in terms of membership in social communities and active involvement in social enterprises", involving both conflictual and harmonious relations. On the other hand, reification is described as the process performed by communities of practice for giving form to our experience by producing objects, like tools, procedures, stories and language. Participation and reification may be analyzed separately, but each of them cannot be replaced with the

other, as they determine each other, playing a crucial role in the negotiation of meaning. For example, mutuality is very important to participation as members of community need self-recognition in each other and through reification meanings are projected in the external world, gaining an independent existence. For the field of Architectural Knowledge, an example of applying the above model would be to consider the community of practice made up of software architects, who have worked together for some period. Through interacting and mutuality, as elements of Participation, the community of architects negotiates the meanings of various concepts (i.e. quality attributes, important design decisions), while the Reification part implies capturing such meanings into artifacts like documents.

The duality model calls for a balance between Participation and Reification. In case of AK, if the former prevails and too little is left unreified, there may be too little material to anchor specific conditions and uncover diverging assumptions (i.e. not documenting the important early design decisions). On the other side, if the latter dominates, and there is too little interactive negotiation, then the shared meaning may not emerge (i.e. resulting in important misunderstandings about the software system to be designed).

7 Conclusion

We think that software architecture, as an inter-disciplinary field, can benefit from borrowing models and concepts from the knowledge management discipline. We believe there are clear benefits in doing so, as presented in the previous section for the case of enriching Architectural Knowledge Management with existing perspectives on knowledge.

So does AKM, as briefly summarized in the cited articles, forget the people that benefit from it? After reviewing its basic concepts, the tools and technologies targeted to their users, and alternative views on knowledge, we can say that AKM does not forget persons, but it does neglect certain aspects of the human nature, that are considered in knowledge management at large. As future work, we plan to bridge the gap between KM and AKM, which we only previewed in this paper.

Acknowledgements

The author wishes to thank Paris, Pavel, Klaas, and Elena for their useful feedback.

References

[1] L. Bass, P. Clements, and R. Kazman, Software architecture in practice, 2003

[2] J. Bosch, "Software architecture: The next step," Lecture notes in computer science, vol. 194, 2004, p. 194–199

[3] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005, pp. 109-120.

[4] P. Kruchten, P. Lago, and H. van Vliet, "Building up and reasoning about architectural knowledge," Lecture Notes in Computer Science, vol. 4214, 2006, p. 43.

[5] R. Farenhorst and R. de Boer, "Knowledge Management in Software Architecture," Software Architecture Knowledge Management: Theory and Practice. Springer, 2009, pp. 21-38.

[6] M. Ali Babar, "Supporting the Software Architecture Process with Knowledge Management," Software Architecture Knowledge Management: Theory and Practice. Springer, 2009

[7] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar, "A comparative study of architecture knowledge management tools," Journal of Systems and Software, vol. 83, 2010, pp. 352-370

[8] P. Kruchten, "Documentation of Software Architecture from a Knowledge Management Perspective – Design Representation," Software Architecture Knowledge Management: Theory and Practice. Springer, 2009, pp. 39-57.

[9] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, Documenting software architectures: views and beyond, Pearson Education, 2002.

[10] P. Kruchten, "Architectural blueprints—The "4+1" view model of software architecture," IEEE Software, vol. 12, 1995, p. 42–50.

[11] P. Kruchten, R. Capilla, J.C. Dueas, "The Decision View's Role in Software Architecture Practice," *Software, IEEE* , vol.26, no.2, pp.36-42, March-April 2009

[12] A. Jansen, "Architectural design decisions", PhD thesis, University of Groninten (2008)

[13] P. Liang, and P. Avgeriou, "Tools and Technologies for Architecture Knowledge Management," Software Architecture Knowledge Management: Theory and Practice. Springer, 2009

[14] R.D. Galliers and S. Newell, "Back to the future: from knowledge management to the management of information and data", Information Systems and E-Business Management, Springer, 2003 (http://is2.lse.ac.uk/Support/ECIS2001/pdf/059_Galliers.pdf)

[15] P. M. Hildreth and C. Kimble "The duality of knowledge," Information Research, Vol. 8 No. 1, October 2002 (http://informationr.net/ir/8-1/paper142.html)

rijksuniversiteit groningen

faculteit wiskunde en natuurwetenschappen

informatica