SC@RUG 2006 – Final Program

We are proud to present you the final program of SC@RUG 2006.
In this document please find the instructions for speakers, chairmen and leaders of the discussion and the final program for both days of the conference.

Welcome

The organizing committee of SC@RUG 2006 welcomes the participants and thanks them for being able to present their latest research and for coorporating in the review process.

Procedure for presentations

- For each paper we will have a total of 30 minutes. Please have 10 minutes for discussion, so limit the length of your talk to 10 minutes for each of the two speakers.
- The organisating committee of SC@RUG 2006 does not offer additional technical support. The room for presentations will have an overhead projector and a beamer available, but no laptop or pc.
- Each presentation is chaired. The **chairman** keeps track of the time for each of the speakers and signals them ten and five minutes before the speaking time is up. Then the **discussion leader** is supposed to chair the discussion. If there is no immediate first question from the audience, the discussion leader should ask a first question to start up the discussion.
- Speakers for a session have time in the break, immediately before the session, to install their laptop and other technical needs. Chairs for a session are supposed to be present ten minutes before the session start in order to gain some information of the speakers. The chairs can so give a more personal introduction for each of the speakers.
- Both speakers for a presentation are announced at the beginning of the presentation. It is the responsibility of the speakers to change halfway the presentation. The chair only signals that time runs out.
- All speakers for one day should be present the whole day.
- The audience is asked to grade all speakers, chairs and discussion leaders during the day.

Where?

SC@RUG 2006 will take place in room WSN31 of the WSN-building.

Instructions for filling in the forms

Please help the organizers grading the presentations. For each presentation, you have attended, please fill in the corresponding page. Use an overall grading for each of the presenters and also for the chairman and the leader of discussion.
Grading is as follows:
1 – very bad
2 – not good
3 – satisfactory
4 – good
5 – excellent
If needed, you may write additional remarks as well.
Thank you for your help.

Day 1: monday January 23rd, 2006

List of participants

Da Silva Campos, A.I
Dos Santos Pires, L.
Ho, T.T.M.
Hove, H. ten
Kidubuka, J.
Kirtley, N.E.
Kooi, M.R. van der
Lops, J.
Ogole, C.
Rademaker, J.T.
Reker, H.G.
Stiekema, H.
Veldthuis, M.
Verschoor, M.
Vries, J.J.G. de
Vrooland, R.
IJbema, M.

Program

09:15 – 09:20	opening
09:20 – 09:50	(CHO) A Comparison of Haskell and OCaml Mark IJbema, Hilverd Reker *chair:* Stiekema, H. *discussion:* Kidubuka, J.
09:50 – 10:20	(3) Visual Textures for Displaying Multidimensional Datasets Mai Ho, Gert-Jan de Vries *chair:* Reker, H.G. *discussion:* Rademaker, J.T.
10:20 – 10:50	(1) Three Methods for reducing clutter, a review Hubert ten Hove, Marco van der Kooi *chair:* Da Silva Campos, A.I *discussion:* Lops, J.

10:50 – 11:10 coffee break

11:10 – 11:40 (2) A Review of some Perceptual Methods in Information Visualization
Nick Kirtley, Robert Vrooland
chair: Verschoor, M.
discussion: Ho, T.T.M.

11:40 – 12:10 (2bis) Combining perception-based visualization techniques
Han Stiekema,
chair: Ogole, C.
discussion: Ijbema, M.

12:10 – 12:40 (SAD) Software Architecture Document Management System
Anton Rademaker, Marten Veldthuis
chair: Vries, J.J.G. de
discussion: Dos Santos Pires, L.

12:40 – 13:10 lunch break

13:10 – 13:40 (4bis) Evaluation of Information Visualization
Ana Campos, Levi Pires
chair: Veldthuis, M.
discussion: Kooi, M.R. van der

13:40 – 14:10 (5) A review of three Multi-Layer Visualization methods
Joris Lops, Mickeal Verschoor
chair: Hove, H. ten
discussion: Vrooland, R.

14:10 – 14:40 (5bis) Multi-layer Visualization: A Review of Selected Methods
Caesar Ogole, Julius Kidubuka
chair: Kirtley, N.E.
discussion:

14:40 – 14:50 closing

Day 2: tuesday January 24th, 2006

List of participants

Babai, M.
Best, J.F.M.
Boersma, S.W.
Buuren, R. van
Craens, G.D.
Donker, R.
Dijk, D.R. van
Hoenderboom, B.J.A.
Hoorn, H. van der
Jong, R. de
Knap, G.H.
Kamphorst, R.J.
Kremer, F.H.
Offringa, A.R.
Postma, B.J.J.
Rink, H.A. van
Speelman, M.
Vandeursen, F.
Veen, H.H. van der

Program

09:15 – 09:20	opening
09:20 – 09:50	(6) Java versus C++ Bart Postma, Remko de Jong *chair:* Knap, G.H. *discussion:* Hoorn, H. van der
09:50 – 10:20	(6bis) Java versus C# Andre Offringa, Daan van Dijk *chair:* Craens, G.D. *discussion:* Rink, H.A. van
10:20 – 10:50	(7bis) Tree-based Image Representation, Filtering and Segmentation Joris Best, Roel Donker

chair: Kremer, F.H.
discussion: Veen, H.H. van der

10:50 – 11:10 coffee break

11:10 – 11:40 (8) (In)security of the Needham-Schroeder public-key protocol
Freek Vandeursen, Mark Speelman
chair: Babai, M.
discussion: Kamphorst, R.J.

11:40 – 12:10 (8bis) The (in)correctness of a security protocol
Gerard Knap, Bart Hoenderboom
chair: Boersma, S.W.
discussion: Buuren, R. van

12:10 – 12:40 (9) Capturing The Missing Link: Design Decisions
George Craens, Hielko van der Hoorn
chair: Postma, B.J.J.
discussion: Jong, R. de

12:40 – 13:10 lunch break

13:10 – 13:40 (9bis) Determining the impact of design decisions
Frans Kremer, Herman van Rink
chair: Offringa, A.R.
discussion: Dijk, D.R. van

13:40 – 14:10 (10) Three methods for modelling variability in software products families
Mohammad Babai, Henk van der Veen
chair: Donker, R.
discussion: Best, J.F.M.

14:10 –14:40 (11) Evolution of Architectural Patterns From the original concept to the architects toolbox
Reinder Kamphorst,
chair: Speelman, M.
discussion: Vandeursen, F.

14:40 –15:10 (11bis) On the evolution of architectural patterns from the original concept to the architects toolkit
Sjouke W. Boersma, Rick van Buuren
chair: Hoenderboom, B.J.A.
discussion:

15:10 – 15:20 closing

List of abstracts

Day 1: monday January 23rd, 2006

09:20 – 09:50	(CHO) A Comparison of Haskell and OCaml Mark IJbema, Hilverd Reker

This paper presents a brief overview of the differences and similarities between the programming languages Haskell (version 98) and OCaml (version 3.08). We provide the reader with a comparison according to a number of theoretical criteria. First we compare the major characteristics of both languages: their type system, evaluation strategy, and module system. Then we examine the available constructs for imperative and object-oriented programming, and discuss some less significant distinctions. Throughout this article, we present source code examples to illustrate various language features, and to help explain the criteria themselves. Comparing (these) two languages on more than just a syntactical level helps one better understand their fundamental properties, and those of programming languages in general. Even though Haskell and OCaml are both functional programming languages and have a lot in common, there are still a number of important differences worth looking at.

09:50 – 10:20	(3) Visual Textures for Displaying Multidimensional Datasets Mai Ho, Gert-Jan de Vries

A technique for displaying multidimensional datasets is the use of visual textures. Based on outcomes of psychophysical research and experiments, methods have been developed to generate visual textures. This paper discusses four of those methods: natural textures, using patterns found in nature; oriented sliver textures, which use oriented stripes and luminance; textures with paper strips, with varying height, density and regularity; and OSC textures (with varying orientation, scale and contrast) using Gabor functions.

10:20 – 10:50	(1) Three Methods for reducing clutter, a review Hubert ten Hove, Marco van der Kooi

Every computer user has encountered images or screens that were unreadable because of the fact that too much data was presented or because the data was too closely clustered. This phenomenon is called clutter. Scientists have addressed the problem of clutter and presented some methods to prevent and locate clutter. The Visual Information Density Adjuster (VIDA) uses layering to reduce the amount of information displayed and works with the constant information density principle. The Feature Congestion method is used to locate clutter within images and screens. It provides methods to solve cluttering, for example with the use of colours to bring the

useful information to the front of the image. Other methods involve sampling, displacement and user perception of 2D scatter plots.

11:10 – 11:40	(2) A Review of some Perceptual Methods in Information Visualization Nick Kirtley, Robert Vrooland

Arranging information in a way that a human can understand is of great importance. Perceptual methods in information visualization helps a user to understand complex data by arranging it in a specific way. The type of visualization depends on a number of factors like user type, amount of data and the complexity of the data. We will discuss several methods for the visualization of information. These methods are needed in order to maintain the optimal flow of information to any user. The better you understand the strengths and weaknesses of visual perception, the better equipped you will be to make use of your reader's abilities to detect structure and patterns in data when it is visually displayed.

11:40 – 12:10	(2bis) Combining perception-based visualization techniques Han Stiekema,

In this article, four papers on perceptual visualization techniques will be reviewed. This paper will cover two things: first, the described techniques are briefly explained and their applicability is discussed. Second, a combination of the techniques is recommended that might result in an even better visualization of the data. An example from one of the papers will be extended to give an indication of where this combination of techniques could work.

12:10 – 12:40	(SAD) Software Architecture Document Management System Anton Rademaker, Marten Veldthuis

Abstract. Today there are many tools involved in the creation of a software architecture document. These tools all maintain different files formats and do not communicate which each other about the semantics of the data itself. Because of this, a lot of information is lost into the document, like design decisions. Later in the software project this causes problems, for example in the area of change management. In this paper, we give an introduction about what a Software Architecture Document Management System (SADMS) is in our opinion and how it helps improving the software engineering process and prevents the erosion of architectural knowledge. A SADMS will include all tools involved in the process of writing a software architecture document. Further, we will investigate possible features for SADMS and their benefits to the software engineering process.

13:10 – 13:40	(4bis) Evaluation of Information Visualization Ana Campos, Levi Pires

The purpose of information visualization is to understand the relationships between the data that is being analyzed. If done properly, easy perception and meaningful conclusions extrapolation are intuitive and can almost be independent of the user. When incorrectly presented, it can lead the user to fruitless reasonings or severe lack of usability. This paper describes several evaluation methods that have been used to test visualization techniques, as well as exploring and comparing different investigations on the field and their respective results.

13:40 – 14:10	(5) A review of three Multi-Layer Visualization methods Joris Lops, Mickeal Verschoor

Visual perception of humans plays an important role when graphical visualization of data is considered. Several psychophysical experiments showed how the low-level human visual system perceives several visual features. These features include orientation, luminance, blur and texture. When complex multidimensional datasets have to be visualized, problems may occur when the data has to be interpreted. In this paper we review three methods for visualizing multi-dimensional data as a set of different data layers. The final image contains all data of each layer.

14:10 – 14:40	(5bis) Multi-layer Visualization: A Review of Selected Methods Caesar Ogole, Julius Kidubuka

While the advances in scientific visualization have made it possible to convert contextual data sets into conspicuous meaningful images, some areas still need further exploration. One of these challenges, which is the focus of this review, is the problem posed by the question: ”Given large, complex and multi-dimensional data sets that represent overlapping surfaces and fields in the real world, what visualization technique can be applied to optimize the display of this class of images?” This problem is particularly difficult owing to the fact that solution methods to multi-layer visualization problems do not only involve many variables such as textures, colors, orientation and (the degree of) transparency of overlaying surfaces, but also, have to integrate user-centered issues such as user feedback. Human perception is core to the considerations. Moreover, the integration of these factors into a typical visualization system takes the form of parametizations of complex and highly interactive algorithmic procedures. No standard guidelines to select suitable sets of parameters exist. In this paper, we review three different techniques of enhancing multi-layer visualization.

Day 2: tuesday January 24th, 2006

09:20 – 09:50	(6) Java versus C++ Bart Postma, Remko de Jong

Java was originally intended to replace C++. However, nowadays both languages still coexist and neither one has been able to replace or subdue the other. In this paper we will make a critical comparison of both languages and try to discuss both differences and similarities of the two languages.

09:50 – 10:20	(6bis) Java versus C# Andre Offringa, Daan van Dijk

Nowadays, there are dozens of languages to choose from when one wants to write a computer program. Many people debate the various differences that the languages have, but because of the various opinions one often cannot conclude much from these discussions. This review article surveys the differences between languages and by comparing the various arguments in discussions about languages, tries to objectively extract some properties which make a language being considered as a ”good” language. Besides the grammar itself, we concern the contextual properties of a language, such as the way it is compiled and for which purposes it can be used. In this survey we focus on the difference and similarities of two languages in special: the difference between C# and Java. Obviously there are slight differences in syntax, but we will discuss other differences and similarities like certain programming constructs, memory management, and other specific advantages and disadvantages of both languages. Finally, we will conclude the paper by using our elaborated definition of ”goodness” of a language and try to draw the conclusion which language ”is best”: C# or Java.

10:20 – 10:50	(7bis) Tree-based Image Representation, Filtering and Segmentation Joris Best, Roel Donker

Images can be stored in many ways, one of the most promising way is storing an image in a tree-structure. Processings on an image means in this case processing the tree-structure which represents the image. The use of a tree-structure makes processings on a tree both simple as efficient. This paper describes three different ways of representing images by a tree-structure. The three representations being researched in this paper are: ”binary partition tree”, ”component tree” and ”foresting transform”. The last section of this paper evaluates the efficiency of all three tree-structures related to image-processing (especially filtering and segmentation).

11:10 – 11:40	(8) (In)security of the Needham-Schroeder public-key protocol Freek Vandeursen, Mark Speelman

The role of security in computer communication is greatly increasing since more and more important communication takes place over computer networks, e.g. the Internet. The starting point of secure communication is the authentication, verifying the identity of the participants. Authentication can be established with the Needham-Schroeder protocol. This protocol was first published in 1978 by Roger Needham and Michael Schroeder, and it took 17 years before it was broken by Gavin Lowe. Lowe proved that this protocol could be broken by a ”man-in-the-middle” attack. In 1990 Gong, Needham and Yahalom published an article in which they presented a method to reason about security protocols. This method emphasizes the reasoning about what the participants in the protocol know and belief. We will show that the vulnerability found by Lowe could also have been found using this method from Gong, Needham and Yahalom, and subsequently we show that the improved protocol is secure. The fact that the vulnerability was not spotted before shows how difficult reasoning about security protocols is, and how difficult it is to guarantee full security.

11:40 – 12:10	(8bis) The (in)correctness of a security protocol Gerard Knap, Bart Hoenderboom

In this paper we explain the mutual authentication protocol from Needham and Schroeder. Although in the beginning the protocol was assumed to be correct, it was later shown that there was a crucial weakness in it. The protocol was vulnerable for the so called “man-inthe- middle attack”. Different adaptations to the protocol are proposed to remove this weakness. The vulnerability is superficially analyzed and an improved version is given. Finally it is shown that the new version of the protocol is more secure. The importance of systematic analysis is also covered.

12:10 – 12:40	(9) Capturing The Missing Link: Design Decisions George Craens, Hielko van der Hoorn

One of the most important artifacts in the software engineering process is the software architecture. Errors made in the design can lead to huge problems in a later stage of the project. Usually only the architecture itself is described, but not the decisions that are made and the rationale behind these decisions. Rationale in the context of architectures describes and explains the used concepts, considered alternatives, and structures of systems [2]. Leaving design decisions and the rationale behind them in the minds of the architects has several drawbacks. The first problem is that this information can easily be lost. Not only when the architect leaves the company, but also the reasoning behind a decision can easily be forgotten. This can cause problems when the architecture will be expanded. Another problem is that an architecture without explicit documented design decision is less clear to stakeholders such as developers and customers. The logical solution that is proposed is to capture this information in the architecting process. It is not clear how this should be done. Some people suggest to include this information in text in the architecture document, while other people propose the use of software tools to capture this

information. In this paper we describe and compare different approaches to capture design decisions.

13:10 – 13:40	(9bis) Determining the impact of design decisions Frans Kremer, Herman van Rink

Recent research in software architecture has resulted in an increasing effort into explicitly capturing architectural design decisions in the software architecture using a variety of means. Although these design decisions are captured, the relations between them are as of yet undefined, or defined on a very basic level. The result is that it is unclear how a change in one design decision affects other design decisions. We propose a means to classify, identify and measure the impact of a change in one design decision, to other design decisions. This is done by providing an ontological view of design decisions where relations are made and graded based on key characteristics, such as component relations (if any), quality attributes and architectural constraints. We present a means how these relations can be defined in AREL.

13:40 – 14:10	(10) Three methods for modelling variability in software products families Mohammad Babai, Henk van der Veen

Abstract. Variability is one of the increasingly important attributes of the modern software development. Variability makes it possible that the software artefacts can be reused and configured to different contexts thus easing development. But managing variabilties brings its own diffculties and there is no consensus on how this must be done. In this paper we describe three methods for modelling variability: Koalish, VSL and COVAMOF.We will give a brief description of each method in order to evaluate them based on their approach and strengths and weaknesses. These properties will be presented and a comparison will be given based on their respective effectiveness in processing different types of requirements (e.g. functional or non-functional). Key words: Variability, product families, variability management, variability in software systems, Variation specification.

14:10 –14:40	(11) Evolution of Architectural Patterns From the original concept to the architects toolbox Reinder Kamphorst,

A pattern is a re-usable concept that has been applied in several working systems. It is not a litteral description of the used structure, rather an abstraction from it that can be used in other systems. A pattern thus provides the architect with a recipe to compose a certain architecture or part of an architecture. A pattern language is a collection of patterns, with clearly stated interdependencies. With a generative pattern language, a whole family of architectures can be generated. The first to propose the idea of pattern languages is not a software architect, but a

bricks n walls architect: Christopher Alexander. He published a number of books on patterns and pattern languages in the late 1970s. In 1987, Ward Cunningham and Kent Beck were the first to bring in practice the archtiectural ideas of Alexander to make a pattern language for software. The authors known as the Gang of Four (GoF) published their book Design Patterns [GoF94] in 1994. A very recent milestone is the publication of POSA books. There have been three stages in the evolution of architectural patterns for software. This paper identifies them by comparing three papers on this subject, and drawing parallels with the aforementioned milestones in the evolution of (architectural) patterns.

14:40 –15:10	(11bis) On the evolution of architectural patterns from the original concept to the architects toolkit Sjouke W. Boersma, Rick van Buuren

Throughout the years the size of software products has grown. A development which started with small individual programs has evolved to large and complex cooperating programs. This development increased the need for a structured way to design the software architecture. After the introduction of design patterns in software architecture, patterns gained a lot of popularity within a relatively short period of time. Nowadays, patterns are widely used within the software architecture and development on many levels. In this paper we will analyze how the usage of patterns has evolved from single patterns to combined patterns and pattern languages.