

## Software Architecture In Practice Sei Series In Software Engineering Hardcover

This book constitutes the thoroughly refereed post-proceedings of the Second International Conference on the Quality of Software Architectures, QoSA 2006, held in Västerås, Sweden in June 2006, co-located with the 9th International Symposium on Component-Based Software Engineering, CBSE 2006. Coverage includes architecture evaluation, managing and applying architectural knowledge, and processes for supporting architecture quality.

The European Conference on Software Architecture (ECSA) is the premier European conference dedicated to the field of software architecture, covering all architectural features of software engineering. It is the follow-up of a successful series of European workshops on software architecture held in the UK in 2004 (Springer LNCS 3047), Italy in 2005 (Springer LNCS 3527), and France in 2006 (Springer LNCS 4344). It evolved into a series of European conferences whose first edition was ECSA 2007, held in Madrid, Spain during September 24–26, 2007 (Springer LNCS 4758). This year's conference was held at the beautiful Coral Beach Hotel and Resort near Paphos in Cyprus. As with the previous versions of the conference, ECSA 2008 (Springer LNCS 5292) provided an international forum for researchers and practitioners from academia and industry to present innovative research and to discuss a wide range of topics in the area of software architecture. It focused on formalisms, technologies, and processes for describing, verifying, validating, transforming, building, and evolving software systems. Covered topics included architecture modelling, architecture description languages, architectural aspects, architecture analysis, transformation and synthesis, architecture evolution, quality attributes, model-driven engineering, built-in testing and architecture-based support for component-based and service-oriented systems. The conference attracted paper submissions from 29 countries (Australia, Belgium, Brazil, Canada, China, Chile, Denmark, Finland, France, Germany, Greece, Ireland, Italy, Lithuania, Luxembourg, Malta, The Netherlands, Norway, Pakistan, Peru, - land, Portugal, Romania, South Africa, Spain, Turkey, the UK, USA, and Venezuela).

Software architecture is a primary factor in the creation and evolution of virtually all products involving software. It is a topic of major interest in the research community where pronusmg formalisms, processes, and technologies are under development. Architecture is also of major interest in industry because it is recognized as a significant leverage point for manipulating such basic development factors as cost, quality, and interval. Its importance is attested to by the fact that there are several international workshop series as well as major conference sessions devoted to it. The First Working IFIP Conference on Software Architecture (WICSAI) provided a focused and dedicated forum for the international software architecture community to unify and coordinate its effort to advance the state of practice and research. WICSA 1 was organized to facilitate information exchange between practising software architects and software architecture researchers. The conference was held in San Antonio, Texas, USA, from February 22nd to February 24th, 1999; it was the initiating event for the new IFIP TC-2 Working Group on Software Architecture. This proceedings document contains the papers accepted for the conference. The papers in this volume comprise both experience reports and technical papers. The proceedings reflect the structure of the conference and are divided into six sections corresponding to the working groups established for the conference.

This textbook aims to prepare students, as well as, practitioners for software design and production. Keeping in mind theory and practice, the book keeps a balance between theoretical foundations and practical considerations. The book by and large meets the requirements of students at all levels of computer science and engineering/information technology for their Software design and Software engineering courses. The book begins with concepts of data and object. This helps in exploring the rationale that guide high level programming language (HLL) design and object oriented frameworks. Once past this post, the book moves on to expand on software design concerns. The book emphasizes the centrality of Parnas's separation of concerns in evolving software designs and architecture. The book extensively explores modelling frameworks such as Unified Modelling Language (UML) and Petri net based methods. Next, the book covers architectural principles and software engineering practices such as Agile – emphasizing software testing during development. It winds up with case studies demonstrating how systems evolve from basic concepts to final products for quality software designs. TARGET AUDIENCE • Undergraduate/postgraduate students of Computer Science and Engineering, and Information Technology • Postgraduate students of Software Engineering/Software Systems

Researchers and professionals will find in this text the thoroughly refereed post-proceedings of the Third International Conference on the Quality of Software Architectures, QoSA 2007, held in Medford, MA, USA, in 2007. It was mounted in conjunction with the 10th International ACM SIGSOFT Symposium on Component-Based Software Engineering, CBSE 2007. The 13 revised full papers presented together with one keynote lecture were carefully reviewed and selected from 42 submissions.

This is the first handbook to cover comprehensively both software engineering and knowledge engineering — two important fields that have become interwoven in recent years. Over 60 international experts have contributed to the book. Each chapter has been written in such a way that a practitioner of software engineering and knowledge engineering can easily understand and obtain useful information. Each chapter covers one topic and can be read independently of other chapters, providing both a general survey of the topic and an in-depth exposition of the state of the art. Practitioners will find this handbook useful when looking for solutions to practical problems. Researchers can use it for quick access to the background, current trends and most important references regarding a certain topic. The handbook consists of two volumes. Volume One covers the basic principles and applications of software engineering and knowledge engineering. Volume Two will cover the basic principles and applications of visual and multimedia software

engineering, knowledge engineering, data mining for software knowledge, and emerging topics in software engineering and knowledge engineering. Part of the new series, Advanced Topics in Science and Technology in China, this book aims to introduce the theoretical foundations, various sub-fields, current research, and practical methods of software architecture. First off, readers can acquire a basic knowledge of software architecture, including why software architecture is necessary. They are then shown how to describe a system's architecture with formal language. The authors continue by delineating which architecture styles are popular in practice. ??????????????,?????????????????????----?????

This book presents selected proceedings of the annual convention of the Computer Society of India. Divided into 10 topical volumes, the proceedings present papers on state-of-the-art research, surveys, and succinct reviews. They cover diverse topics ranging from communications networks to big data analytics, and from system architecture to cyber security. This book focuses on Software Engineering, and informs readers about the state of the art in software engineering by gathering high-quality papers that represent the outcomes of consolidated research and innovations in Software Engineering and related areas. In addition to helping practitioners and researchers understand the chief issues involved in designing, developing, evolving and validating complex software systems, it provides comprehensive information on developing professional careers in Software Engineering. It also provides insights into various research issues such as software reliability, verification and validation, security and extensibility, as well as the latest concepts like component-based development, software process models, process-driven systems and human-computer collaborative systems.

This book constitutes the refereed proceedings of the 5th European Conference on Software Architecture, ECSA 2011, held in Essen, Germany, in September 2011. The 13 revised full papers presented together with 24 emerging research papers, and 7 research challenge poster papers were carefully reviewed and selected from over 100 submissions. The papers are organized in topical sections on requirements and software architectures; software architecture, components, and compositions; quality attributes and software architectures; software product line architectures; architectural models, patterns and styles; short papers; process and management of architectural decisions; software architecture run-time aspects; ADLs and metamodels; and services and software architectures.

In the past ten years or so, software architecture has emerged as a central notion in the development of complex software systems. Software architecture is now accepted in the software engineering research and development community as a manageable and meaningful abstraction of the system under development and is applied throughout the software development life cycle, from requirements analysis and validation, to design and down to code and execution level. This book presents the tutorial lectures given by leading authorities at the Third International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2003, held in Bertinoro, Italy, in September 2003. The book is ideally suited for advanced courses on software architecture as well as for ongoing education of software engineers using formal methods in their day-to-day professional work.

Software reuse depicts a great vision for the software industry. It has been widely viewed as a promising way to improve both the productivity and quality of software development. However, despite of the successes we have achieved, there are still many issues that have limited the promotion of software reuse in the real world. Therefore, software reuse has remained an important hotspot of research. ICSR is the premier international conference in the field of software reuse. It has been an important venue for presenting advances and improvements within the software reuse domain, and a powerful driving force in promoting the interaction between researchers and practitioners. The theme of ICSR 10 was "High Confidence Software Reuse in Large Systems. " A high confidence system is one that behaves in a well-understood and predictable fashion. Today's trends towards widespread use of commercial off-the-shelf (COTS) technology, increased integration, continuous evolution, and larger scale are yielding more complex software systems. So, the problem of how to build high confidence complex systems and how to reuse software with a high level of confidence has become a new attractive topic for research. Furthermore, high-level software asset reuse has been a goal for the last 20–30 years, and it can still be considered an unsolved question. Components-based development, MDA-MDE-MDD, extreme programming, and other techniques or methods are promising approaches to software reuse that still need more research. These proceedings report on the current state of the art in software reuse.

This book constitutes the refereed proceedings of the First European Conference on Software Architecture, ECSA 2007, held in Aranjuez, Spain. The 12 revised long papers presented together with four short papers cover description languages and metamodels, architecture-based code generation, run-time monitoring, requirements engineering, service-oriented architectures, aspect-oriented software architectures, ontology-based approaches, autonomic systems, middleware and web services.

This book constitutes the proceedings of the 9th European Conference on Software Architecture, ECSA 2015, held in Cavtat, Croatia in September 2015. The 12 full papers and 15 short papers presented together with three education and training papers in this volume were carefully reviewed and selected from 100 submissions. They are organized in topical sections named: adaptation; design approaches; decisions and social aspects; education and training; cloud and green; agile and smart systems; analysis and automation; services and ecosystems. Agile software development approaches have had significant impact on industrial software development practices. Today, agile software development has penetrated to most IT companies across the globe, with an intention to increase quality, productivity, and profitability. Comprehensive knowledge is needed to understand the architectural challenges involved in adopting and using agile approaches and industrial practices to deal with the development of large, architecturally challenging systems in an agile way. Agile Software Architecture focuses on gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox. Readers will learn how agile and architectural cultures can co-exist and support each other according to the context. Moreover, this book will also provide useful leads for future research in architecture and agile to bridge such gaps by developing appropriate approaches that incorporate architecturally sound practices in agile methods. Presents a consolidated view of the state-of-art and state-of-practice as well as the newest research findings Identifies gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox Explains whether or not and how agile and architectural cultures can co-

exist and support each other depending upon the context Provides useful leads for future research in both architecture and agile to bridge such gaps by developing appropriate approaches, which incorporate architecturally sound practices in agile methods

Much of a software architect's life is spent designing software systems to meet a set of quality requirements. General software quality attributes include scalability, security, performance or reliability. Quality attribute requirements are part of an application's non-functional requirements, which capture the many facets of how the functional requirements of an application are achieved. Understanding, modeling and continually evaluating quality attributes throughout a project lifecycle are all complex engineering tasks which continue to challenge the software engineering scientific community. While we search for improved approaches, methods, formalisms and tools that are usable in practice and can scale to large systems, the complexity of the applications that the software industry is challenged to build is ever increasing. Thus, as a research community, there is little opportunity for us to rest on our laurels, as our innovations that address new aspects of system complexity must be deployed and validated. To this end the 5th International Conference on the Quality of Software Architectures (QoSA) 2009 focused on architectures for adaptive software systems. Modern software systems must often reconfigure their structure and behavior to respond to continuous changes in requirements and in their execution environment. In these settings, quality models are helpful at an architectural level to guide systematic model-driven software development strategies by evaluating the impact of competing architectural choices.

The award-winning and highly influential *Software Architecture in Practice*, Third Edition, has been substantially revised to reflect the latest developments in the field. In a real-world setting, the book once again introduces the concepts and best practices of software architecture—how a software system is structured and how that system's elements are meant to interact. Distinct from the details of implementation, algorithm, and data representation, an architecture holds the key to achieving system quality, is a reusable asset that can be applied to subsequent systems, and is crucial to a software organization's business strategy. The authors have structured this edition around the concept of architecture influence cycles. Each cycle shows how architecture influences, and is influenced by, a particular context in which architecture plays a critical role. Contexts include technical environment, the life cycle of a project, an organization's business profile, and the architect's professional practices. The authors also have greatly expanded their treatment of quality attributes, which remain central to their architecture philosophy—with an entire chapter devoted to each attribute—and broadened their treatment of architectural patterns. If you design, develop, or manage large software systems (or plan to do so), you will find this book to be a valuable resource for getting up to speed on the state of the art. Totally new material covers Contexts of software architecture: technical, project, business, and professional Architecture competence: what this means both for individuals and organizations The origins of business goals and how this affects architecture Architecturally significant requirements, and how to determine them Architecture in the life cycle, including generate-and-test as a design philosophy; architecture conformance during implementation; architecture and testing; and architecture and agile development Architecture and current technologies, such as the cloud, social networks, and end-user devices

This book illustrates the role of software architecture and its application in business. The author describes enterprise architecture along with business architecture to show the role of software architecture in both areas. The place of software architecture in business is outlined from many perspectives in this context. The book outlines quality attributes and how managers can use software architecture to build high quality products. Topics include business software architecture, dealing with qualities, achieving quality attributes, managing business qualities, software product line, Internet of Things (IOT), and Service Oriented Business Architecture. The book is intended to benefit students, researchers, software architects, and business architects. Provides quick and easy access to all the important aspects of software architecture in business; Highlights a wide variety of concepts of software architecture in a straightforward manner, for students, practitioners, or architects; Presents different applications of software architecture in business.

On behalf of the Organizing Committee for this event, we are glad to welcome you to IWASE 2006, the First International Workshop on Advanced Software Engineering. We hope you will enjoy the traditional Chilean hospitality and, of course, please tell us how we can make your visit a pleasant and useful experience. The goal of this Workshop is to create a new forum for researchers, professionals and educators to discuss advanced software engineering topics. A distinctive feature of this Workshop is its attempt to foster interactions between the Latin-American software engineering community and computer scientists around the world. This is an opportunity to discuss with other researchers or simply to meet new colleagues. IWASE 2006 has been organized to facilitate strong interactions among those attending it and to offer ample time for discussing each paper. IWASE 2006 attracted 28 submissions from 14 countries, 8 of them outside Latin-America. Each of the 28 articles was reviewed by at least three members of the Program Committee. As a result of this rigorous reviewing process, 13 papers were accepted: nine full papers and four work-in-progress papers. These papers were grouped in four tracks; software architecture, software modeling, software development process and experiences in software development.

Abstract: "Architectural decisions have a great impact on the consequent quality of software systems. As a result, it is important to evaluate how a software architecture meets its quality demands. Though much focus has been placed on modeling and describing the software architecture as a design artifact, we found that relatively little is known about the current experience with software architecture evaluation. This report details the results of two workshops on software architecture evaluation, held at the Software Engineering Institute (SEI) on November 9- 10, 1995 and May 9-10, 1996. The purpose of the workshops was to determine the state of industrial practice in the evaluation of software architectures with respect to a set of desired quality attributes, and to uncover recommendations for best practices. In this report, we summarize the findings of the two workshops, define a set of dimensions to characterize various software architecture evaluation techniques, and make concrete recommendations for implementing architecture

evaluation practices."

Software architectures have gained wide popularity in the last decade. They generally play a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software systems. Component-oriented and aspect-oriented programming enables software engineers to implement complex applications from a set of pre-defined components. Software Architectures and Component Technology collects excellent chapters on software architectures and component technologies from well-known authors, who not only explain the advantages, but also present the shortcomings of the current approaches while introducing novel solutions to overcome the shortcomings. The unique features of this book are: evaluates the current architecture design methods and component composition techniques and explains their shortcomings; presents three practical architecture design methods in detail; gives four industrial architecture design examples; presents conceptual models for distributed message-based architectures; explains techniques for refining architectures into components; presents the recent developments in component and aspect-oriented techniques; explains the status of research on Piccola, Hyper/J®, Pluggable Composite Adapters and Composition Filters. Software Architectures and Component Technology is a suitable text for graduate level students in computer science and engineering, and as a reference for researchers and practitioners in industry.

This is the first handbook to cover comprehensively both software engineering and knowledge engineering -- two important fields that have become interwoven in recent years. Over 60 international experts have contributed to the book. Each chapter has been written in such a way that a practitioner of software engineering and knowledge engineering can easily understand and obtain useful information. Each chapter covers one topic and can be read independently of other chapters, providing both a general survey of the topic and an in-depth exposition of the state of the art. Practitioners will find this handbook useful when looking for solutions to practical problems. Researchers can use it for quick access to the background, current trends and most important references regarding a certain topic. The handbook consists of two volumes. Volume One covers the basic principles and applications of software engineering and knowledge engineering. Volume Two will cover the basic principles and applications of visual and multimedia software engineering, knowledge engineering, data mining for software knowledge, and emerging topics in software engineering and knowledge engineering.

?????????:?????;?????:????????,????????;?:????????????????

The concepts and practice of software architecture are introduced--what a system is designed to do and how its components are meant to interact with each other. The authors cover not only essential technical topics for specifying and validating a system, but, for the first time, emphasize the importance of the business context in which large systems are designed.

Designing Software Architectures is the first step-by-step guide to making the crucial design decisions that can make or break your software architecture. SEI expert Rick Kazman and Dr. Humberto Cervantes provide comprehensive guidance for ensuring that your architectural design decisions are consistently rational and evidence-based. Drawing on their own extensive experience, they demonstrate how to craft designs that are practical and effective, and support all phases of architectural development, from requirements to documentation. You'll learn how to successfully integrate the design process in an organizational context, including designing systems that will be built with agile methods. The authors begin with a general review of software architecture concepts and the software architecture lifecycle. Next, they explain what architecture design really means, introduce key design concepts and principles, and walk through both conventional and alternative design processes. Building on this foundation, they introduce the new Attribute-Driven Design (ADD) 3.0 process, walk the reader through two extended ADD 3.0 case studies, and demonstrate how ADD 3.0 can lead to more successful designs. You'll learn how to scale design and analysis up and down - for example, to design for pre-sales processes and lightweight architecture reviews. Kazman and Cervantes illuminate the relationships between analysis and design, introduce a set of reusable design primitives, and identify issues and solutions for new domains, including cloud, mobile, and big data. Design is the core activity for software designers and architects, but for most practitioners, it's been a black art. This book offers the systematic guidance you need to consistently do it rationally, and do it right.

This is the eagerly-anticipated revision to one of the seminal books in the field of software architecture which clearly defines and explains the topic.

Abstract: "Architecture documentation has emerged as an important architecture-related practice. In 2002, researchers at the Carnegie Mellon[registered trademark] Software Engineering Institute completed Documenting Software Architectures: Views and Beyond (V & B), an approach that holds that documenting a software architecture is a matter of choosing a set of relevant views of the architecture, documenting each of those views, and then documenting information that applies to more than one view or to the set of views as a whole. Details of the approach include a method for choosing the most relevant views, standard templates for documenting views and the information beyond them, and definitions of the templates' content. At about the same time, the Institute of Electrical and Electronics Engineers (IEEE) was developing a recommended best practice for describing architectures for software-intensive systems -- ANSI/IEEE Std. 1471-2000. Like V & B, that standard takes a multi-view approach to the task of architecture documentation, and it establishes a conceptual framework for architectural description and defines the content of an architectural description. This technical note summarizes the two approaches and shows how a software architecture document prepared using the V & B approach can be made compliant with Std. 1471-2000."

Embedded Systems Architecture is a practical and technical guide to understanding the components that make up an embedded system's architecture. This book is perfect for those starting out as technical professionals such as engineers, programmers and designers of embedded systems; and also for students of computer science, computer engineering and electrical engineering. It gives a much-needed 'big picture' for recently graduated engineers grappling with understanding the design of real-world systems for the first time, and provides professionals with a systems-level picture of the key elements that can go into an embedded design, providing a firm foundation on which to build their skills. Real-world approach to the fundamentals, as well as the design and architecture process, makes this book a popular reference for the daunted or the inexperienced: if in doubt, the answer is in here! Fully updated with new coverage of FPGAs, testing, middleware and the latest programming techniques in C, plus complete source code and sample code, reference designs and tools online make this the complete package

Visit the companion web site at <http://booksite.elsevier.com/9780123821966/> for source code, design examples, data sheets and more. A true introductory book, provides a comprehensive get up and running reference for those new to the field, and updating skills: assumes no prior knowledge beyond undergrad level electrical engineering. Addresses the needs of practicing engineers, enabling it to get to the point more directly, and cover more ground. Covers hardware, software and middleware in a single volume. Includes a library of design examples and design tools, plus a complete set of source code and embedded systems design tutorial materials from companion website.

?????:??

This book constitutes the joint refereed proceedings of two colocated events: the First International Conference on the Quality of Software Architectures (QoSA 2005) and the Second International Workshop on Software Quality (SOQUA 2005) held in Erfurt, Germany, in September 2005. The 18 revised full papers presented were carefully reviewed and selected from 48 submissions. For QoSA 2005 only 12 papers - of the 31 submitted - were accepted for presentation; they are concerned with research and experiences that investigate the influence a specific software architecture has on software quality aspects. The papers are organized in topical sections on software architecture evaluation, formal approaches to model-driven QoS-handling, modelling QoS in software architectures, software architectures applied, architectural design for QoS, and model-driven software reliability estimation. The 6 papers accepted for SOQUA 2005 - from 17 submissions - mainly focus on quality assurance and on software testing. They are organized in topical sections on test case selection, model-based testing, unit testing, and performance testing.

DevOps promises to accelerate the release of new software features and improve monitoring of systems in production, but its crucial implications for software architects and architecture are often ignored. In *DevOps: A Software Architect's Perspective*, three leading architects address these issues head-on. The authors review decisions software architects must make in order to achieve DevOps' goals and clarify how other DevOps participants are likely to impact the architect's work. They also provide the organizational, technical, and operational context needed to deploy DevOps more efficiently, and review DevOps' impact on each development phase. The authors also address cross-cutting concerns that link multiple functions, offering practical insights into compliance, performance, reliability, repeatability, and security. This guide demonstrates the authors' ideas in action with three real-world case studies: datacenter maintenance for business continuity, management of a continuous deployment pipeline, and migration to a microservice architecture. Comprehensive coverage includes:

- Why DevOps can require major changes in both system architecture and IT roles
- How virtualization and the cloud can enable DevOps practices
- Integrating operations and its service lifecycle into DevOps
- Designing new systems to work well with DevOps practices
- Overcoming cultural and communication differences between Dev and Ops
- Integrating DevOps with agile methods and TDD
- Handling failure detection, upgrade planning, and other key issues
- Managing consistency issues arising from DevOps' independent deployment models
- Integrating security controls, roles, and audits into DevOps
- Preparing a business plan for DevOps adoption, rollout, and measurement

Presents three methods for evaluating the structure of large software systems during the design phase. The three techniques separately test for whether quality goals are met and how they interact; for modifiability and functionality; and for the feasibility and suitability of a set of services provided by a portion of the system. The authors, who are members of Carnegie Mellon's Software Engineering Institute, illustrate how to apply each step of the methods through case studies. c. Book News Inc.

?????:?????

Abstract: "This report represents the first milestone of a work in progress. That work is a comprehensive handbook on how to produce high-quality documentation for software architectures. The handbook, tentatively entitled *Software Architecture Documentation in Practice*, will be published in mid- to late-2000 by Addison Wesley Longman as a book in the SEI series on software engineering. Aimed squarely at the practitioner, the handbook is intended to fill a gap in the literature: There is a complete lack of language-independent guidance about how to actually capture an architecture in written form so that it can fulfill its purpose as a communication vehicle providing a unified design vision to all of the varied stakeholders of a development project. The theme of the work is that documenting an architecture entails documenting the set of relevant views of that architecture, and then completing the picture with documentation of information that transcends any single view. The report lays out our approach and organization for the complete book, and provides full guidance for one of the most commonly used architectural views: the layer diagram. The audience for this book is the community of practicing architects, apprentice architects, and developers who are on the receiving end of architectural documentation."

This book contains the proceedings of the Fourth International Workshop on Product Family Engineering, PFE-4, held in Bilbao, Spain, October 3–5, 2001. This workshop was the fourth in a series started in 1996, with the same subject, software product-family engineering. Proceedings of the second and third workshops have been published as LNCS 1429 and LNCS 1951. The workshops were organized within co-operation projects of European industry, the first two by ARES (Esprit IV 20.477) 1995–1999. This project had three industrial and three academic partners, and focused on software architectures for product families. Some of the partners continued in ITEA project 99005, ESAPS (1999–2001). ITEA is the software development program (?!2023) within the European Eureka initiative. ITEA projects last for two years and ESAPS ? was succeeded by CAFE (ITEA ip00004), which started in 2001 and will terminate in 2003. This workshop was initially prepared within ESAPS and the ? preparation continued in CAFE. Due to the attacks in the USA of September 11, several people were not able to ?y and therefore did not show up. However, we have included their submissions in these proceedings. The session chair presented these submissions, and their inputs were used during the discussions. It was planned that Henk Obbink be workshop chair, and Linda Northrop and

Sergio Bandinelli be co-chairs. However, because of personal circumstances Henk Obbink was not able to leave home during the workshop. Moreover both co-chairs had already enough other duties. Therefore the chairing duties were taken over by the program chair, Frank van der Linden.

[Copyright: ccf0512125349ef8bc808f4f5175e227](#)