A Component Architecture For High Performance Scientific

Component Models and Systems for Grid Applications is the essential reference for the most current research on Grid technologies. This first volume of the CoreGRID series addresses such vital issues as the architecture of the Grid, the way software will influence the development of the Grid, and the practical applications of Grid technologies for individuals and businesses alike. Part I of the book, "Application-Oriented Designs", focuses on development methodology and how it may contribute to a more component-based use of the Grid. "Middleware Architecture", the second part, examines portable Grid engines, hierarchical infrastructures, interoperability, as well as workflow modeling environments. The final part of the book, "Communication Frameworks", looks at dynamic self-adaptation, collective operations, and higher-order components. With Component Models and Systems for Grid Applications, editors Vladimir Getov and Thilo Kielmann offer the computing professional and the computing researcher the most informative, up-to-date, and forward-looking thoughts on the fast-growing field of Grid studies. These proceedings represent the work of researchers participating in the 15th European Conference on Cyber Warfare and Security (ECCWS 2016) which is being hosted this year by the Universitat der Bundeswehr, Munich, Germany on the 7-8 July 2016. ECCWS is a recognised event on the International research conferences calendar and provides a valuable plat-form for individuals to present their

research findings, display their work in progress and discuss conceptual and empirical advances in the area of Cyberwar and Cyber Security. It provides an important opportunity for researchers and managers to come together with peers to share their experiences of using the varied and ex-panding range of Cyberwar and Cyber Security research available to them. With an initial submission of 110 abstracts, after the double blind, peer review process there are 37 Academic research papers and 11 PhD research papers, 1 Master's research paper, 2 Work In Progress papers and 2 non-academic papers published in these Conference Proceedings. These papers come from many different coun-tries including Austria, Belgium, Canada, Czech Republic, Finland, France, Germany, Greece, Hungary, Ireland, Kenya, Luxembourg, Netherlands, Norway, Portugal, Romania, Russia, Slovenia, South Africa, Sweden, Turkey, UK and USA. This is not only highlighting the international character of the conference, but is also promising very interesting discussions based on the broad treasure trove of experience of our community and partici-pants."

Deployment is the act of taking components and readying them for productive use. There may be steps following deployment, such as installation or m- agement related functions, but all decisions about how to con?gure and c-pose/assemble a component are made at the deployment stage. This is therefore the one opportunity in the software lifecycle to bridge the gap between what the component developer couldn’t know about the deployment environment and what the environment’s developer couldn’t know about the open set of depl- able components. It is not surprising that deployment as a dedicated step gains importance when

addressing issues of system-wide qualities, such as coping with constrained resources or preparing for component adaptation and system evolution. Yet, component deployment is still a discipline in its infancy: it became mainstream practice only in the mid 1990s. Much of the best practice impulse originated in products like Microsoft's Transaction Server and its approach to attribute-based programming and later products like Enterprise JavaBeans and now the Corba Component Model. All these address the speci?c needs of enterprise appli- tion servers. However, the potential of the deployment concept goes far beyond this. Deployment can and should touch e?ectively all truly component-based solutions. The proceedings of Component Deployment 2002 represent a good cro- section of the gamut of deployment issues. From customization to address - source constraints to recon?guration of deployed systems and from architecture to design to languages, the avid reader will ?nd some contribution.

Includes articles in topic areas such as autonomic computing, operating system architectures, and open source software technologies and applications.

Architecting Data Intensive Applications

LDAP Metadirectory Provisioning Methodology

Artificial Intelligence and Simulation

Designing Embedded Hardware

ECCWS2016-Proceedings fo the 15th European Conference on Cyber Warfare and Security "

Euro-Par 2008 Workshops - Parallel Processing

This book constitutes the refereed proceedings of the 11th International Conference on Software Reuse, ICSR

2009, held in Falls Church, VA, USA, in September 2009. The 28 full papers were caryfully selected from numerous submissions. 2009 was the year that ICSR went back to its roots. The theme was Formal Foundations of Reuse and Domain Engineering. The theory and formal foundations that underlie current reuse and domain engineering practice were explored and current advancements to get an idea of where the field of reuse was headed, were looked at. Many of the papers in these proceedings reflect that theme, e.g. component reuse and verification, feature modeling, generators and model-driven development, industry experience, product lines, reuse and patterns, service-oriented environments.

Presenting the state of the art in component-based software testing, this cutting-edge resource offers you an in-depth understanding of the current issues, challenges, needs and solutions in this critical area. The book discusses the very latest advances in component-based testing and quality assurance in an accessible tutorial

format, making the material easy to comprehend and benefit from no matter what your professional level.
important, and how it differs from traditional software testing. From an introduction to software components, testing component-based software and validation methods for software components, to performance testing and measurement, standards and certification and verification of quality for component-based systems, you get a revealing snapshot of the key developments in this area, including important research findings. This volume also serves as a textbook for related courses at the advanced undergraduate or graduate level.
With the increasing complexity and interdisciplinary nature of scientific applications, code reuse is becoming increasingly important in scientific computing. One method for facilitating code reuse is the use of components technologies, which have been used widely in industry. However, components have only recently worked their way into scientific computing. Language interoperability is an important

underlying technology for these component architectures. In this paper, we present an approach to language interoperability for a high-performance parallel, component architecture being developed by the Common Component Architecture (CCA) group. Our approach is based on Interface Definition Language (IDL) techniques. We have developed a Scientific Interface Definition Language (SIDL), as well as bindings to C and Fortran. We have also developed a SIDL compiler and run-time library support for reference counting, reflection, object management, and exception handling (Babel). Results from using Babel to call a standard numerical solver library (written in C) from C and Fortran show that the cost of using Babel is minimal, where as the savings in development time and the benefits of object-oriented development support for C and Fortran far outweigh the costs.

This volume presents the accepted papers for the 4th International Conference onGridandCooperativeComputin g(GCC2005),heldinBeijing,China,during November 30 – December 3, 2005.The

conferenceseries of GCC aims to provide an international forum for the presentation and discussion of research trends on the theory, method, and design of Grid and cooperative computing as well as their scienti?c, engineering and commercial applications. It has become a major annual event in this area. The First International Conference on Grid and Cooperative Computing (GCC2002)received 168submissions.GCC2003received550submissions,from which 176 regular papers and 173 short papers were accepted. The acceptance rate of regular papers was 32%, and the total acceptance rate was 64%. GCC 2004 received 427 main-conference submissions and 154 workshop submissions. The main conference accepted 96 regular papers and 62 short papers. The - ceptance rate of the regular papers was 23%. The total acceptance rate of the main conference was 37%. For this conference, we received 576 submissions. Each was reviewed by two independent members of the International Program Committee. After carefully evaluating their originality and quality, we accepted 57

regular papers and 84 short papers. The acceptance rate of regular papers was 10%. The total acc- tance rate was 25%.

Software Applications: Concepts, Methodologies, Tools, and Applications

Software Producibility for Defense

Enterprise Security Architecture Using IBM Tivoli Security Solutions

A Case Study

Algorithmic and Architectural Gaming Design: Implementation and Development

4th International Conference, Beijing, China, November 30 -- December 3, 2005, Proceedings

Scientific computing has often been called the third approach to scientific discovery, emerging as a peer to experimentation and theory. Historically, the synergy between experimentation and theory has been well understood: experiments give insight into possible theories, theories inspire experiments, experiments reinforce or invalidate theories, and so on. As scientific computing has evolved to produce results that meet or exceed the quality of experimental and theoretical results, it has become indispensable.Parallel processing has been an enabling technology in scientific computing for more than 20 years. This book is the first in-depth discussion of parallel computing in 10 years; it reflects the mix of topics that mathematicians, computer

scientists, and computational scientists focus on to make parallel processing effective for scientific problems. Presently, the impact of parallel processing on scientific computing varies greatly across disciplines, but it plays a vital role in most problem domains and is absolutely essential in many of them. Parallel Processing for Scientific Computing is divided into four parts: The first concerns performance modeling, analysis, and optimization; the second focuses on parallel algorithms and software for an array of problems common to many modeling and simulation applications; the third emphasizes tools and environments that can ease and enhance the process of application development; and the fourth provides a sampling of applications that require parallel computing for scaling to solve larger and realistic models that can advance science and engineering. This edited volume serves as an up-to-date reference for researchers and application developers on the state of the art in scientific computing. It also serves as an excellent overview and introduction, especially for graduate and senior-level undergraduate students interested in computational modeling and simulation and related computer science and applied mathematics aspects.Contents List of Figures; List of Tables; Preface; Chapter 1: Frontiers of Scientific Computing: An Overview; Part I: Performance Modeling, Analysis and Optimization. Chapter 2: Performance Analysis: From Art to Science; Chapter 3: Approaches to Architecture-Aware Parallel Scientific

Computation; Chapter 4: Achieving High Performance on the BlueGene/L Supercomputer; Chapter 5: Performance Evaluation and Modeling of Ultra-Scale Systems; Part II: Parallel Algorithms and Enabling Technologies. Chapter 6: Partitioning and Load Balancing; Chapter 7: Combinatorial Parallel and Scientific Computing; Chapter 8: Parallel Adaptive Mesh Refinement; Chapter 9: Parallel Sparse Solvers, Preconditioners, and Their Applications; Chapter 10: A Survey of Parallelization Techniques for Multigrid Solvers; Chapter 11: Fault Tolerance in Large-Scale Scientific Computing; Part III: Tools and Frameworks for Parallel Applications. Chapter 12: Parallel Tools and Environments: A Survey; Chapter 13: Parallel Linear Algebra Software; Chapter 14: High-Performance Component Software Systems; Chapter 15: Integrating Component-Based Scientific Computing Software; Part IV: Applications of Parallel Computing. Chapter 16: Parallel Algorithms for PDE-Constrained Optimization; Chapter 17: Massively Parallel Mixed-Integer Programming; Chapter 18: Parallel Methods and Software for Multicomponent Simulations; Chapter 19: Parallel Computational Biology; Chapter 20: Opportunities and Challenges for Parallel Computing in Science and Engineering; Index. User Interfaces (UI) of applications, since about 2010, are usually implemented by dedicated frontend programs, following a Rich-Client architecture and are based on the Web technologies HTML, CSS and JavaScript. This

approach provides great flexibility and power, but comes with an inherent great overall complexity of UIs, running on a continuously changing technology stack. This is because since over twenty years Web technologies still progress at an extremely high invention rate and unfortunately at the same time still regularly reinvent part of their self. This situation is harmless for small UIs, consisting of just a handful dialogs and having to last for just about one or two years. However, it becomes a major hurdle for large UIs, consisting of a few hundred dialogs and having to last for five or more years. This is especially the case for the complex UIs of industrial Business Information Systems. The main scientific contribution of this dissertation is the Hierarchical User Interface Component Architecture (HUICA), a scalable software architecture for Rich-Client based User Interfaces. It is primarily based on the important architecture principle Separation of Concerns (SoC), the derived idea of Hierarchical Composition, the invented design pattern Model-View-Controller/Component-Tree (MVC/CT) and the existing concepts Presentation Model and Data Binding.

Not a new version - included warning for self signed X509 certificates - see section 5.2 This IBM® Redbooks® publication describes the concepts, architecture, and implementation of the IBM XIV® Storage System. The XIV Storage System is a scalable enterprise storage system that is based on a grid array of hardware components. It can attach to both Fibre Channel

Protocol (FCP) and IP network Small Computer System Interface (iSCSI) capable hosts. This system is a good fit for clients who want to be able to grow capacity without managing multiple tiers of storage. The XIV Storage System is suited for mixed or random access workloads, including online transaction processing, video streamings, images, email, and emerging workload areas, such as Web 2.0 and cloud storage. The focus of this edition is on the XIV Gen3 running Version 11.5.x of the XIV system software, which brings enhanced value for the XIV Storage System in cloud environments. It offers multitenancy support, VMware vCloud Suite integration, more discrete performance classes, and RESTful API enhancements that expand cloud automation integration. Version 11.5 introduces support for three-site mirroring to provide high availability and disaster recovery. It also enables capacity planning through the Hyper-Scale Manager, mobile push notifications for real-time alerts, and enhanced security. Version 11.5.1 supports 6TB drives and VMware vSphere Virtual Volumes (VVOL). In the first few chapters of this book, we describe many of the unique and powerful concepts that form the basis of the XIV Storage System logical and physical architecture. We explain how the system eliminates direct dependencies between the hardware elements and the software that governs the system. In subsequent chapters, we explain the planning and preparation tasks that are required to deploy the system in your

environment by using the intuitive yet powerful XIV Storage Manager GUI or the XIV command-line interface. We also describe the performance characteristics of the XIV Storage System and present options for alerting and monitoring, including enhanced secure remote support. This book is for IT professionals who want an understanding of the XIV Storage System. It is also for readers who need detailed advice on how to configure and use the system.

Model-based performance prediction systematically deals with the evaluation of software performance to avoid for example bottlenecks, estimate execution environment sizing, or identify scalability limitations for new usage scenarios. Such performance predictions require up-to-date software performance models. This book describes a new integrated reverse engineering approach for the reconstruction of parameterised software performance models (software component architecture and behaviour).

Component Deployment

Component-Based Software Engineering

Component Models and Systems for Grid Applications

9th International Symposium, CBSE 2006, Västeras, Sweden, June 29 - July 1, 2006, Proceedings

Critical Code

Support Constant Change

On behalf of the Organizing Committee I am pleased to present the proceedings of the

2005 Symposium on Component-Based Software Engineering (CBSE). CBSE is concerned with the development of software-intensive systems from reusable parts (components), the development of reusable parts, and system maintenance and improvement by means of component replacement and c- tomization. CBSE 2005, "Software Components at Work," was the eighth in a series of events that promote a science and technology foundation for achieving predictable quality in software systems through the use of software component technology and its associated software engineering practices. We were fortunate to have a dedicated Program Committee comprised of 30 internationally recognized researchers and industrial practitioners. We received 91 submissions andeach paper wasreviewedby at least three ProgramComm- tee members (four for papers with an author on the Program Committee). The entirereviewingprocesswassupportedbyCyberChairPro,theWeb-basedpaper submission andreviewsystemdevelopedandsupportedbyRichardvandeStadt of Borbala Online Conference Services. After a two-day virtual Program C- mittee meeting, 21 submissions were accepted as long papers and 2

submissions were accepted as short papers. Architect and design data-intensive applications and, in the process, learn how to collect, process, store, govern, and expose data for a variety of use cases Key Features Integrate the data-intensive approach into your application architecture Create a robust application layout with effective messaging and data querying architecture Enable smooth data flow and make the data of your application intensive and fast Book Description Are you an architect or a developer who looks at your own applications gingerly while browsing through Facebook and applauding it silently for its data-intensive, yet fluent and efficient, behaviour? This book is your gateway to build smart data-intensive systems by incorporating the core data-intensive architectural principles, patterns, and techniques directly into your application architecture. This book starts by taking you through the primary design challenges involved with architecting data-intensive applications. You will learn how to implement data curation and data dissemination, depending on the volume of your data. You will then implement your application architecture one step at a time. You will get to grips with implementing the

correct message delivery protocols and creating a data layer that doesn't fail when running high traffic. This book will show you how you can divide your application into layers, each of which adheres to the single responsibility principle. By the end of this book, you will learn to streamline your thoughts and make the right choice in terms of technologies and architectural principles based on the problem at hand. What you will learn Understand how to envision a data-intensive system Identify and compare the non-functional requirements of a data collection component Understand patterns involving data processing, as well as technologies that help to speed up the development of data processing systems Understand how to implement Data Governance policies at design time using various Open Source Tools Recognize the anti-patterns to avoid while designing a data store for applications Understand the different data dissemination technologies available to query the data in an efficient manner Implement a simple data governance policy that can be extended using Apache Falcon Who this book is for This book is for developers and data architects who have to code, test, deploy, and/or maintain large-

scale, high data volume applications. It is also useful for system architects who need to understand various non-functional aspects revolving around Data Intensive Systems. This book explores various aspects of software creation and development as well as data and information processing. It covers relevant topics such as business analysis, business rules, requirements engineering, software development processes, software defect prediction, information management systems, and knowledge management solutions. Lastly, the book presents lessons learned in information and data management processes and procedures.

This is the refereed proceedings of the 9th International Symposium on Component-Based Software Engineering, CBSE 2006, held in Västerås, Sweden in June/July 2006. The 22 revised full papers and 9 revised short papers presented cover issues concerned with the development of software-intensive systems from reusable parts, the development of reusable parts, and system maintenance and improvement by means of component replacement and customization.

Concepts, Methodologies, Tools, and Applications

A Component Architecture for High-

Performance Scientific Computing
Parallel Processing for Scientific Computing
Implementation and Development
11th International Conference on Software Reuse, ICSR 2009, Falls Church, VA, USA, September 27-30, 2009. Proceedings
8th International Symposium, CBSE 2005, St. Louis, MO, USA, May 14-15, 2005

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.

This book constitutes the refereed proceedings of the 10th International Conference on Information Systems Security, ICISS 2014, held in Hyderabad, India, in December 2014. The 20 revised full papers and 5 short papers presented together

with 3 invited papers were carefully reviewed and selected from 129 submissions. The papers address the following topics: security inferences; security policies; security user interfaces; security attacks; malware detection; forensics; and location based security services.

Over the last decade, the role of computational simulations in all aspects of aerospace design has steadily increased. However, despite the many advances, the time required for computations is far too long. This book examines new ideas and methodologies that may, in the next twenty years, revolutionize scientific computing. The book specifically looks at trends in algorithm research, human computer interface, network-based computing, surface modeling and grid generation and computer hardware and architecture. The book provides a good overview of the current state-of-the-art and provides guidelines for future research directions. The book is intended for computational scientists active in the field and program managers making strategic research decisions.

The Common Component Architecture (CCA) provides a means for software developers to manage the complexity of large-scale scientific simulations and to move toward a plug-and-play environment for high-performance computing. In the scientific computing context, component models also promote collaboration using independently developed software, thereby allowing particular individuals or groups to focus on the aspects of greatest interest to them. The CCA supports parallel and distributed computing as well as local high-performance connections between components in a language-independent manner. The design places minimal requirements on components and thus facilitates the integration of existing code into the CCA environment. The CCA model imposes

minimal overhead to minimize the impact on application performance. The focus on high performance distinguishes the CCA from most other component models. The CCA is being applied within an increasing range of disciplines, including combustion research, global climate simulation, and computational chemistry.

Grid and Cooperative Computing - GCC 2005

IFIP/ACM Working Conference, CD 2002, Berlin, Germany, June 20-21, 2002, Proceedings

Towards Software Development (Volume 4)

Proceedings of the Workshop on Component Models and Systems for Grid Applications held June 26, 2004 in Saint Malo, France.

Language Interoperability for High-performance Parallel Scientific Components

4th International Conference, Kraków, Poland, June 6–9, 2004, Proceedings, Part III

This work provides system architects a methodology for the implementation of x.500 and LDAP based metadirectory provisioning systems. In addition this work assists in the business process analysis that accompanies any deployment. DOC Safe Harbor & European Commission's Directive on Data Protection (Directive 95/46/EC) issues are also addressed.

This book constitutes the refereed post-proceedings of the 13th International Conference on AI, Simulation, and Planning in High Autonomy Systems, AIS 2004, held in Jeju Island, Korea in October 2004. The 74 revised full papers presented together with 2 invited keynote papers were carefully reviewed and

selected from 170 submissions; after the conference, the papers went through another round of revision. The papers are organized in topical sections on modeling and simulation methodologies, intelligent control, computer and network security, HLA and simulator interoperation, manufacturing, agent-based modeling, DEVS modeling and simulation, parallel and distributed modeling and simulation, mobile computer networks, Web-based simulation and natural systems, modeling and simulation environments, AI and simulation, component-based modeling, watermarking and semantics, graphics, visualization and animation, and business modeling.

The International Conference on Computational Science (ICCS 2004) held in Krak´ ow, Poland, June 6–9, 2004, was a follow-up to the highly successful ICCS 2003 held at two locations, in Melbourne, Australia and St. Petersburg, Russia; ICCS 2002 in Amsterdam, The Netherlands; and ICCS 2001 in San Francisco, USA. As computational science is still evolving in its quest for subjects of inves- gation and e?cient methods, ICCS 2004 was devised as a forum for scientists from mathematics and computer science, as the basic computing disciplines and application areas, interested in advanced computational methods for physics, chemistry, life sciences, engineering, arts and humanities, as well as computer system vendors and software developers. The main objective of this conference was to discuss

problems and solutions in all areas, to identify new issues, to shape future directions of research, and to help users apply various advanced computational techniques. The event harvested recent developments in com- tational gridsandnextgenerationcomputingsystems,tool s,advancednumerical methods, data-driven systems, and novel application ?elds, such as complex - stems, ?nance, econo-physics and population evolution.

Intelligent readers who want to build their own embedded computer systems-- installed in everything from cell phones to cars to handheld organizers to refrigerators-- will find this book to be the most in-depth, practical, and up-to-date guide on the market. Designing Embedded Hardware carefully steers between the practical and philosophical aspects, so developers can both create their own devices and gadgets and customize and extend off-the-shelf systems. There are hundreds of books to choose from if you need to learn programming, but only a few are available if you want to learn to create hardware. Designing Embedded Hardware provides software and hardware engineers with no prior experience in embedded systems with the necessary conceptual and design building blocks to understand the architectures of embedded systems. Written to provide the depth of coverage and real-world examples developers need, Designing Embedded Hardware also provides a road-map to the pitfalls and traps to avoid in designing embedded systems. Designing Embedded

Hardware covers such essential topics as: The principles of developing computer hardware Core hardware designs Assembly language concepts Parallel I/O Analog-digital conversion Timers (internal and external) UART Serial Peripheral Interface Inter-Integrated Circuit Bus Controller Area Network (CAN) Data Converter Interface (DCI) Low-power operation This invaluable and eminently useful book gives you the practical tools and skills to develop, build, and program your own application-specific computers.

The Integrated Architecture Framework Explained

Testing and Quality Assurance for Component-based Software

Reconstruction of Software Component Architectures and Behaviour Models Using Static and Dynamic Analysis

Hierarchical User Interface Component Architecture

Исторія западныхъ славянъ

Software Architecture for Business

Providing all the latest on a topic of extreme commercial relevance, this book contains the refereed proceedings of the 10th International ACM SIGSOFT Symposium on Component-Based Software Engineering, held in Medford, MA, USA in July 2007. The 19 revised full papers presented were carefully reviewed and selected from 89 submissions. The papers feature new trends in global software services and distributed systems architectures to push the limits of established and tested component-based methods, tools and platforms.

We present a case study of performance measurement and modeling of a CCA (Common Component Architecture) component-based

application in a high performance computing environment. We explore issues peculiar to component-based HPC applications and propose a performance measurement infrastructure for HPC based loosely on recent work done for Grid environments. A prototypical implementation of the infrastructure is used to collect data for a three components in a scientific application and construct performance models for two of them. Both computational and message-passing performance are addressed.

This IBM Redbooks publication reviews the overall Tivoli Enterprise Security Architecture. It focuses on the integration of audit and compliance, access control, identity management, and federation throughout extensive e-business enterprise implementations. The available security product diversity in the marketplace challenges everyone in charge of designing single secure solutions or an overall enterprise security architecture. With Access Manager, Identity Manager, Federated Identity Manager, Security Compliance Manager, Security Operations Manager, Directory Server, and Directory Integrator, Tivoli offers a complete set of products designed to address these challenges. This book describes the major logical and physical components of each of the Tivoli products. It also depicts several e-business scenarios with different security challenges and requirements. By matching the desired Tivoli security product criteria, this publication describes the appropriate security implementations that meet the targeted requirements. This book is a valuable resource for security officers, administrators, and architects who want to understand and implement enterprise security following architectural guidelines.

The Common Component Architecture (CCA) provides a means for developers to manage the complexity of large-scale scientific software systems and to move toward a "plug and play" environment for high-performance computing. The CCA model allows for a direct connection between components within the same process to maintain performance on inter-component calls. It is neutral with respect to parallelism, allowing components to use whatever means

they desire to communicate within their parallel "cohort." We will discuss in detail the importance of performance in the design of the CCA and will analyze the performance costs associated with features of the CCA.

Performance Measurement and Modeling of Component Applications in a High Performance Computing Environment

Data-Centric Business and Applications

Computational Aerosciences in the 21st Century

An Architecture for Checkpointing and Migration of Distributed Components on the Grid

An architect's guide to building maintainable and change-tolerant applications with Java and Quarkus

Building Evolutionary Architectures

This book captures and communicates the wealth of architecture experience Capgemini has gathered as a member of The Open Group – a vendor- and technology-neutral consortium formed by major industry players – in developing, deploying, and using its “Integrated Architecture Framework” (IAF) since its origination in 1993. Today, many elements of IAF have been incorporated into the new version 9 of TOGAF, the related Open Group standard. The authors, all working on and with IAF for many years, here provide a full reference to IAF and a guide on how to apply it. In addition, they describe in detail the relations between IAF and the architecture standards TOGAF and Archimate and other development or process frameworks like ITIL, CMMI, and RUP. Their presentation is targeted at architects, project managers, and process analysts who have either considered or are already working

with IAF – they will find many roadmaps, case studies, checklists, and tips and advice for their daily work.

Video games represent a unique blend of programming, art, music, and unbridled creativity. To the general public, they are perhaps the most exciting computer applications ever undertaken. In the field of computer science, they have been the impetus for a continuous stream of innovations designed to provide gaming enthusiasts with the most realistic and enjoyable gaming experience possible. Algorithmic and Architectural Gaming Design: Implementation and Development discusses the most recent advances in the field of video game design, with particular emphasis on practical examples of game development, including design and implementation. The target audience of this book includes educators, students, practitioners, professionals, and researchers working in the area of video game design and development. Anyone actively developing video games will benefit from the practical application of fundamental computer science concepts demonstrated in this book.

A practical guide for software architects and Java developers to build cloud-native hexagonal applications using Java and Quarkus to create systems that are easier to refactor, scale, and maintain Key FeaturesLearn techniques to

decouple business and technology code in an applicationApply hexagonal architecture principles to produce more organized, coherent, and maintainable softwareMinimize technical debts and tackle complexities derived from multiple teams dealing with the same code baseBook Description Hexagonal architecture enhances developers' productivity by decoupling business code from technology code, making the software more change-tolerant, and allowing it to evolve and incorporate new technologies without the need for significant refactoring. By adhering to hexagonal principles, you can structure your software in a way that reduces the effort required to understand and maintain the code. This book starts with an in-depth analysis of hexagonal architecture's building blocks, such as entities, use cases, ports, and adapters. You'll learn how to assemble business code in the Domain hexagon, create features by using ports and use cases in the Application hexagon, and make your software compatible with different technologies by employing adapters in the Framework hexagon. Moving on, you'll get your hands dirty developing a system based on a real-world scenario applying all the hexagonal architecture's building blocks. By creating a hexagonal system, you'll also understand how you can use Java modules to reinforce dependency inversion and ensure the isolation of each hexagon

in the architecture. Finally, you'll get to grips with using Quarkus to turn your hexagonal application into a cloud-native system. By the end of this hexagonal architecture book, you'll be able to bring order and sanity to the development of complex and long-lasting applications. What you will learnFind out how to assemble business rules algorithms using the specification design patternCombine domain-driven design techniques with hexagonal principles to create powerful domain modelsEmploy adapters to make the system support different protocols such as REST, gRPC, and WebSocketCreate a module and package structure based on hexagonal principlesUse Java modules to enforce dependency inversion and ensure isolation between software componentsImplement Quarkus DI to manage the life cycle of input and output portsWho this book is for This book is for software architects and Java developers who want to improve code maintainability and enhance productivity with an architecture that allows changes in technology without compromising business logic, which is precisely what hexagonal architecture does. Intermediate knowledge of the Java programming language and familiarity with Jakarta EE will help you to get the most out of this book.

This book constitutes the refereed proceedings of the 7th International Symposium on Component-

Based Software Engineering, CBSE 2004, held in Edinburgh, UK in May 2004 as an adjunct event to ICSE 2004. The 12 revised long papers and 13 revised short papers presented together with the abstracts of 2 invited talks were carefully reviewed and selected from 82 submissions. The papers are organized in topical sections on generation and adoptation of component-based systems, tools and building frameworks, components for real-time embedded systems, extra-functional properties of components and component-based systems, and measurement and prediction models for component assemblies.

10th International Conference, ICISS 2014, Hyderabad, India, December 16-20, 2014. Proceedings

A Component Architecture for High-Performance Computing

Formal Foundations of Reuse and Domain Engineering

10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007, Proceedings

13th International Conference on AI, Simulation, and Planning in High Autonomy Systems, AIS 2004, Jeju Island, Korea, October 4-6, 2004, Revised Selected Papers

Information Systems Security

Parallel and distributed processing, although within the focus of computer science research for a long

time, is gaining more and more importance in a wide spectrum of applications. These proceedings aim to demonstrate the use of parallel and distributed processing concepts in different application fields, and attempt to spark interest in novel research directions to parallel and high-performance computing research in general. The objective of these workshops is to specifically address researchers coming from university, industry and governmental research organizations and application-oriented companies in order to close the gap between purely scientific research and the applicab- ity of the research ideas to real-life problems. Euro-Par is an annual series of international conferences dedicated to the pro- tion and advancement of all aspects of parallel and distributed computing. The 2008 event was the 14th issue of the conference. Euro-Par has for a long time been eager to attract colocated events sharing the same goal of promoting the dev- opment of parallel and distributed computing, both as an industrial technique and an academic discipline, extending the frontier of both the state of the art and the state of the practice. Since 2006, Euro-Par has been offering researchers the chance to co- cate advanced technical workshops back-to-back with the main conference.

The software development ecosystem is constantly changing, providing a constant stream of new tools, frameworks, techniques, and paradigms. Over the past few years, incremental developments in core

engineering practices for software development have created the foundations for rethinking how architecture changes over time, along with ways to protect important architectural characteristics as it evolves. This practical guide ties those parts together with a new way to think about architecture and time.

This volume of the Lecture Notes in Computer Science series contains the proceedings of the second Working Conference on Component Deployment, which took place May 20-21, 2004, at the e-Science Institute in Edinburgh, Scotland, as a collocated event of the International Conference on Software Engineering. Component deployment addresses what needs to be done afteracomponent has been developed. Component deployment includes activities such as com- nent customization, con?guration, integration, activation, de-activation and - commissioning. The emerging research community that investigates component deployment concernsitselfwiththeprinciples,methodsandtoolsfor deployment activities. The community held its ?rst working conference in Berlin, Germany, inJune2002. TheproceedingswerepublishedbySpringer-Verlagasvolume2370 of the Lecture Notes in Computer Science series. The program of this year's conference consisted of an invited talk and 16 technical paper presentations. The invited talk was given by Patrick Goldsack of Hewlett PackardResearchLaboratoriesBristol, UK. He presented the Smart- Frog component deployment

framework that HP released as Open Source. The tec hnicalpaperswerecarefullyselectedfromatotalof34sub mittedpapers.Each paper was thoroughly peer reviewed by at least three members of the program committee and consensus on acceptance was achieved by means of an electronic PC meeting. Critical Code contemplates Department of Defense (DoD) needs and priorities for software research and suggests a research agenda and related actions. Building on two prior books--Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale and Preliminary Observations on DoD Software Research Needs and Priorities--the present volume assesses the nature of the national investment in software research and, in particular, considers ways to revitalize the knowledge base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs. Critical Code discusses four sets of questions: To what extent is software capability significant for the DoD? Is it becoming more or less significant and strategic in systems development? Will the advances in software producibility needed by the DoD emerge unaided from industry at a pace sufficient to meet evolving defense requirements? What are the opportunities for the DoD to make more effective use of emerging technology to improve software capability and software producibility? In which technology areas should the DoD invest in research to advance defense software capability and producibility?

7th International Symposium, CBSE 2004, Edinburgh, UK, May 24-25, 2004, Proceedings
VHPC 2008, UNICORE 2008, HPPC 2008, SGS 2008, PROPER 2008, ROIA 2008, and DPA 2008, Las Palmas de Gran Canaria, Spain, August 25-26, 2008, Revised Selected Papers
Performance Measurement and Modeling of Component Applications in a High Performance Computing Environment - A Case Study
Proceedings of the ICASE/LaRC/NSF/ARO Workshop, conducted by the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, The National Science Foundation and the Army Research Office, April 22–24, 1998
Designing Hexagonal Architecture with Java
Second International Working Conference, CD 2004, Edinburgh, UK, May 20-21, 2004, Proceedings
As the commercial software industry burgeoned, it was clear that increasingly complex software would require a mechanism scaling across people, geography and time. The answer came in the concept of software components. Software components are stand-alone modules that have a prescribed means for composition into an application. Component concepts enable, for example, MS Word documents to appear in MS Powerpoint slides, and has led to the point-and-click user interfaces that inhabit most desktop computers today. The idea of a component in software comes from its root word: "composeable". The process of connecting components together into an application can be likened

to their electrical component analogue: hook transistors, diodes and resistors together one way, and you have a radio, another way and you have an MP3 player. The Common Component Architecture is a component model created by computational scientists from all of the DOE laboratories to establish a plug and play standard for high-performance computing. Recently the Common Component Architecture has been named on the Top 10 DOE Science Achievements in 2002 list (http://www.sc.doe.gov/sub/accomplishments/top%5F10.htm). Though computing has been synonymous with the DOE labs long before anyone dreamed of having a computer on their desktop, scientific computing high-performance scientific computing in particular has not benefitted from these advancements. This is because parallel computing, the mainstay of high performance computing, is not amenable to the component software existing in the commercial world. Parallel software requires a model that enables cooperation among thousands of individual processors, a situation not familiar to commercial software vendors. The Common Component Architecture was conceived to fill this gap.

An Overview of the Common Component Architecture

Why, What, How

прибалтійскихъ, чеховъ и поляковъ : лекціи, читанныя въ М. университетѣ и на Высшихъ женскихъ курсахъ въ Москвѣ

IBM XIV Storage System Architecture and Implementation

Computational Science – ICCS 2004