OSGi Service Layer Enhancements

Nico Goeminne, Gregory De Jans, Jan Hollez,
Bart Dhoedt, Filip De Turck, Frank Gielen
Ghent University - IBBT - IMEC
Department of Information Technology
Gaston Crommenlaan 8, bus 201
9050 Gent, Belgium

Abstract – *In recent years software development design shifted from the art of crafting a home tailored solution to the art of component composition. These components are offered in various formats, such as software libraries (Java Archives, .NET Assemblies) or web services and are provided by many different vendors. In these multi-vendor environments there is a genuine need for integration and interoperability. Integration and interoperability is a first step, once this is achieved components can seamlessly use services from different providers, and that is when service policies come into play. A policy mechanism allows fine grained control over the service usage. The OSGi Service Platform is a service container which allows seamless integration of components and services but its service layer lacks a well defined mechanism for dynamic service policy management. Two approaches are presented for enhancing the service layer with policies. The first approach extends the platform while the second one adapts the plug-in components. Finally they are compared and evaluated against multiple requirements; usability, performance, transparency and backward compatibility.*

Keywords: OSGi, Service Policies

1 Introduction

Integration and interoperability are the most important factors to make a multi-vendor component model successful. New design philosophies and concepts are built around these values such as the Service Oriented Architecture (SOA) and the Enterprise Service Bus (ESB). Within the service oriented architecture a service is an entity that performs some functionality and which can be shared among multiple components.
Whenever services are exposed or shared, there is a need for service policy management. The top level of that mechanism is the policy decision logic, which is the place where rules are imposed on service use. These rules can be defined in various formats and implemented using different languages and libraries or by rule engines. Once the rules are defined they need to be enforced within the lower layer. This paper presents the components needed in OSGi Service Platform [1] [2] to support the lower layer of the policy mechanism.
The OSGi Service Platform technology allows integration of components and services from different vendors or service providers and is focused on home networks but can be used in a broader environment. The OSGi Specifications (R4) are gaining momentum being a core technology for the eclipse IDE and several JSRs [3]. The unit of deployment is a component called a bundle. A bundle is a Java archive(jar) file, and the code inside can be activated by the framework through the bundle's activator class. A bundle may contain multiple services, which are plain old java objects that are registered within the platform's service registry. Each of those services can be used by other bundles, thus creating some kind of dependency among each other.
There are several approaches to help the bundle developer manage those dependencies. For example use the ServiceTracker, Service Binder [4] [5] [6], or Declarative Services [7] [8] to reduce the impact of service dependencies. Releasing a service and in particular a java object may prove to be more difficult than one would think as pointed out by [9], but solutions are in development [10].
Bundles can compete for the service usage, and when two bundles wish to use the same service a policy mechanism needs to be in place that handles granting or revoking actions based on priority rules. This work enables policies within the service layer of the OSGi Service Platform.
Paragraph 2 outlines a use cases which shows the need for service policy management and introduces two models, the Framework Extensions model and the Bundle Adaptation model that could be used to support service policies within the OSGi Service Platform. The Framework Extensions model adds interfaces and behaviour definitions to the OSGi R3 specifications. The Bundle Adaptation model implements the same behaviour outside the OSGi core framework. It requires some modifications to bundles who wish to participate. Paragraphs 3 and 4 describe the models in detail. Paragraph 5 describes how to build a policy enforcement component using the models. Their performance is analyzed in paragraph 6 and the remaining conclusions are in paragraph 7.

2 Service Policy Management

The following use case clearly show the need for some kind of service policy management.
A use case: Appliance Control. When both a power saving service and a home surveillance service use a lighting service, some rules should be in place to govern the priorities. We do not want the power saving service turning off the lights

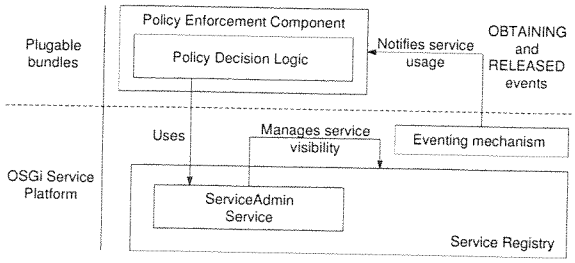


Fig. 1. Global decomposition and operation of model 1: Framework extensions. The Eventing mechanism, the Service Admin and the Policy Enforcement Component work together extending the OSGi Service Platform with service policy capabilities

when the home surveillance service detects some suspicious activities and tries to turn the lights on.

The current OSGi Specifications are not sufficient to support the use case. They do not allow fine grained service management and only support a flat view on the Service Registry also pointed out by [11]. A service exported by a bundle can be used by all bundles. The Permissions Admin Specification (R3) and the Conditional Permission Admin Specification (R4), provide means of managing access to a service, but do not define a model of behaviour. What should happen when the usage of a service is prohibited for a specific bundle? Furthermore their management capabilities do not correspond with the dynamic nature of the Service Platform. In order to support fine grained service management two models are proposed and implemented.

Model 1: Framework Extensions. In this model bundles are unaffected, yet the OSGi framework is slightly extended. Great care should be taken to make the extensions as 'natural' as possible, meaning the extensions follow the design philosophy of the service platform.

Model 2: Bundle Adaptations. In this model the OSGi Service Platform is not affected, allowing the model to be implemented as a set of bundles that are backward compatible with any OSGi R3 platform. Yet in this model the bundles that wish to support policies are adapted.

3 Model 1: Framework Extensions

The model as shown in Fig. 1 contains three separate components, their roles, implications and implementations are discussed below.

Eventing mechanism. The subsystem gives notifications when a service is being obtained or released. The subsystem can only be implemented as a direct hook into the OSGi framework.

ServiceAdmin Service. The ServiceAdmin service is a system service that offers an interface to manage the visibility of a service toward a bundle. The service can only be implemented as a direct hook into the OSGi framework.

Policy Enforcement Component (PEC). The PEC processes the information provided by the eventing mechanism and makes decisions based on that information to adjust the service's visibility towards the bundles. The PEC is a standalone bundle and does not need framework modification, it just uses the newly provided capabilities and is common for both models as described in paragraph 5.

3.1 Service Event Extensions

The OSGi specification (R3-R4), currently offers three kinds of service events. A bundle may wish to register a ServiceListener and act on those events.

ServiceEvent.REGISTERED. When a bundle offers a service to the platform, it registers the service in the platform's service registry. A registered service event is issued.

ServiceEvent.MODIFIED. When the properties of the service are changed by the owning bundle, a modified service event is sent.

ServiceEvent.UNREGISTERING. An unregistering service event is generated when a service is about to be removed from the service registry.

Yet two other major service related 'actions', the obtaining and the releasing of a service, have no corresponding event, although they are indicated by the getService and the ungetService API method calls. When investigating the service usage one must always use the request response pattern (active polling) instead of the event driven model. Therefore the OSGi eventing mechanism should be extended with two new event types:

ServiceEvent.OBTAINING. Before a service object is delivered to the requesting bundle, a service event should be sent to all interested listeners, indicating which service (by means of the service reference) is requested by which bundle.

ServiceEvent.RELEASED. After a bundle released a certain service object, all interested listeners should be notified. Again the service event should denote which bundle is releasing the service.

It should be noted that the OSGi spec had foreseen future additions to the service event types (used in R4). The class org.osgi.framework.ServiceEvent was adjusted to handle the two new event types.

An obvious choice for listening to these new service events would be the existing ServiceListener interface. That approach has three disadvantages. First, there is no control over which listener will be notified first. In some cases one wishes to create some kind of manager that reacts upon an obtaining request. They would prefer to get notified before other bundles are notified.

As a second disadvantage, each time a service is requested or released all listeners are notified. This means a big performance loss, since services are obtained and released a lot, especially at peak moments during bootstrap or shutdown and to a lesser extent at bundle deployment time. Besides those moments the service platform is rather stable. The performance impact of having many listeners is analysed in paragraph 6.

The third disadvantage: bundles that erroneously rely on the fact that there are only three service event types are broken.

To solve all three disadvantages a new interface that extends ServiceListener was defined; the SynchronousServiceListener

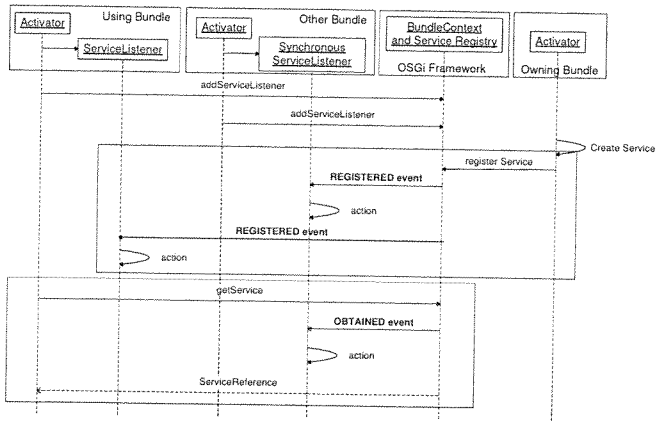


Fig. 2. Sequence diagram showing the actions following registration or obtaining a service. Note that the SynchronizedServiceListener is notified first and the plain old service listener is not notified in case of the obtaining event.

(cf. SynchronousBundleListener). All notifications are handled by the inherited serviceChanged method. The service platform delivers both the existing and the newly added event types to the SynchronousServiceListener, whereas ServiceListeners only receive the old service events and never receives the OBTAINING or RELEASED events. This solves the performance and the legacy listener problem in one effort. Furthermore all events are delivered to the SynchronousServiceListeners before they are delivered to the ServiceListeners. Both listeners can be added to the framework the same way using the bundle context; no new API method is required and the same filter rules can be applied to both synchronous and non-synchronous service listeners. The difference in operation is shown in Fig. 2. Now, the three disadvantages are resolved.

An extra advantage, using the new service event types one can observe and profile the service usage of a bundle or of a service, making it easier to debug. For example one could build a debug tool, where authorized service usage (per bundle) is logged and unauthorized or unpredicted usage is reported. Furthermore one could build watches on services.

3.2 Service Registry Extensions

In order to support service policies, we need more control over which bundle may use which service. The security facilities within the OSGi platform offer some control, but are rather static. In fact once a service usage is granted it is hard to return on that decision, because security checks are only done when the service is first requested. Denying access afterwards comes only in effect when the service is released and requested a second time. The model clearly lacks essential functionality if one wishes to revoke a service from a using bundle.

In this proposal, a bundle gets a filtered view on the service registry. A management interface called the ServiceAdmin service is available for fine-tuning that view and is listed below.

```
public interface ServiceAdmin {
```

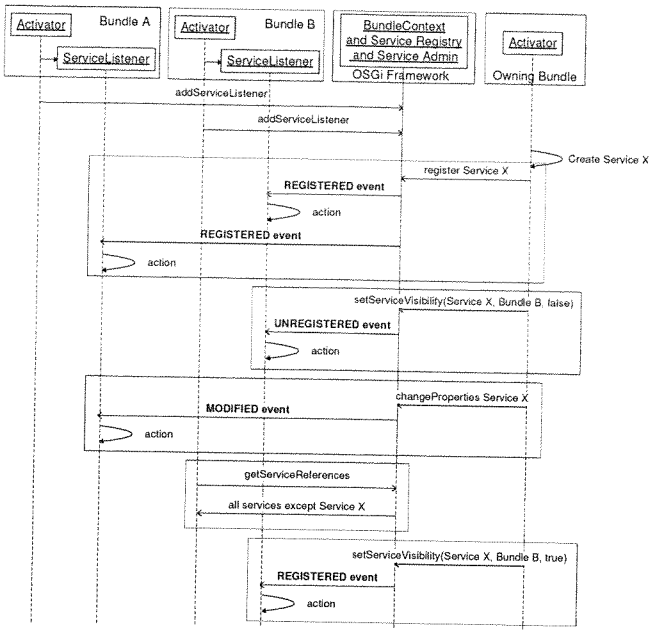


Fig. 3. Sequence diagram showing the actions and consequences when using the ServiceAdmin service

```
  public void setServiceVisibility(
    ServiceReference serviceReference,
    Bundle bundle, boolean visible);

  public ServiceReference []
    getInVisibleServices (Bundle bundle);

  public boolean isVisible(
    ServiceReference serviceReference,
    Bundle bundle);
}
```

A service can be made invisible for a bundle by using the setServiceVisibility method. The service visibility status towards a bundle can be analyzed by the two other methods. Bundles that are blocked from seeing certain services will not see them when invoking a getServiceReference on the BundleContext, and ServiceListeners registered by that Bundle will not be notified. As far as the blocked service concerns the owning bundle has unregistered the service (cf. Fig. 3).

The concept of filtering has already been used in the OSGi platform R3, when a bundle does not have the right permission. Or in release R4, where due to the support of multiple packages, service requests by interface name may cause returning a non class compatible service, which is thus filtered out. Where the standard OSGi frameworks just do filtering, our adaptation sends events, notifying bundles that the service they are using has been unregistered. That event is only delivered to the one blocked bundle. In fact that bundle thinks the service is no longer available, and thus releases the service, while other bundles do not receive the unregistered event, and are still using the service. When the service gets unblocked for our blocked bundle, a registered event is sent towards the blocked bundle, which thinks the service is newly available and can start using it. As mentioned before while being in blocked

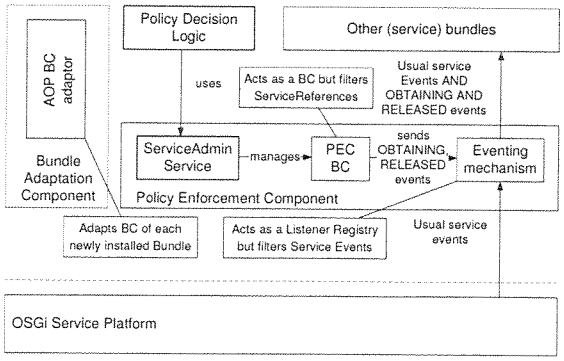


Fig. 4. Global decomposition and operation of model 2: bundle adaptations. All components that were placed inside the OSGi Service Platform are now placed in separate bundles

state, the bundle does not receive any event notification of the service (As far as the blocked bundle is concerned the service does not exist).

4 Model 2: Bundle Adaptations

The functional requirements for this model are exactly the same as for model 1. Interested bundles should still be notified of the service usage behaviours, as well as they should be able to manage the service visibility. Therefore the three main components, the Eventing mechanism, the Service Admin service and the Policy Enforcement Component stay exactly the same. Two non functional requirements are added, firstly the model should not require any OSGi framework extensions (should run on every OSGi framework) and secondly, the model should support legacy bundles (bundles and their developers are unaware of the policy management component). The model is shown in Fig. 4. When shifting these components out of the OSGi framework some problems arise.

Eventing mechanism. Two problems are manifested, firstly how can this subcomponent discover the exact time a service is obtained or released? And secondly, how can it filter out events for invisible services?

ServiceAdmin Service. Again there are two problems to deal with. How can it send the unregistered event for a service towards a bundle and thus making the service invisible for that bundle? And how can it filter out invisible services when a bundle issues the getServiceReferences method on the bundle context?

Policy Enforcement Component. The PEC is already a standalone bundle and is common for both models as described in paragraph 5.

A solution to all of those problems can be found by wrapping the bundle context and providing the bundle with a special bundle context. The bundle context is the bundle's interface towards the framework. When a bundle requests or releases a service it will invoke the getService or ungetService on the bundle context. The wrapping bundle context intercepts those calls and this solves the first problem.

Service listeners are registered with the OSGi framework by invoking the registerServiceListener method on the bundle context. At that time the wrapping bundle context can choose to add the listener to the eventing mechanism instead of adding it to the framework. The eventing mechanism now has full control over all service listeners, which solves the second problem. It listens to the framework and filters out service events before delivering the events to the service listeners. As a surplus it can send specialized events towards a certain service listener, which solves the first problem of the Service Admin. Furthermore the wrapping bundle context can filter out invisible services when a bundle invokes the getServiceReferences method on the bundle context, which solves the last problem.

By wrapping the bundle context all framework extensions are eliminated, but at a price. The policy enforcement framework now has to manage and maintain all service listeners and the bundles need to be adapted so that they are provided with the wrapping bundle context.

5 A Policy Enforcement Component

The policy enforcement component is a separate bundle and is common for both models. The proposed models provide a sufficient toolset to implement any kind of service policy management component. In fact, the PEC's decision logic could be provided and implemented by third parties using different technologies, e.g. hard coded rules, XML configuration, rule based, etc.

A simple PEC implementation for the use case could look like the code below.

```
public class SynchronousServiceListenerImpl implements
  SynchronousServiceListener {

  private Bundle surveillance, powersaving;
  private ServiceAdmin admin;

  public SynchronousServiceListenerImpl(Bundle surveillance,
      Bundle powersaving, ServiceAdmin admin) {
    this.surveillance = surveillance;
    this.powersaving = powersaving;
    this.admin = admin;
  }

  public void serviceChanged(ServiceEvent event) {
    ServiceReference ref = event.getServiceReference();
    switch(event.getType()){
      case ServiceEvent.OBTAINING:
        if (surveillance.getBundleId() ==
              event.getBundle().getBundleId()){
          admin.setServiceVisibility(ref,powersaving,false);
        }
        break;
      case ServiceEvent.RELEASED:
        if (surveillance.getBundleId() ==
              event.getBundle().getBundleId()){
          admin.setServiceVisibility(ref,powersaving,true);
        }
        break;
    }
  }
}
```

The listener uses the ServiceAdmin service to control the visibility of the lighting service towards the powersaving bundle. When the lighting service is obtained by the surveillance bundle the visibility for the powersaving bundle is turned off.

The overall operation is shown in Fig. 5. Furthermore an OSGi filter makes sure the listener only receives events related to the lighting service.

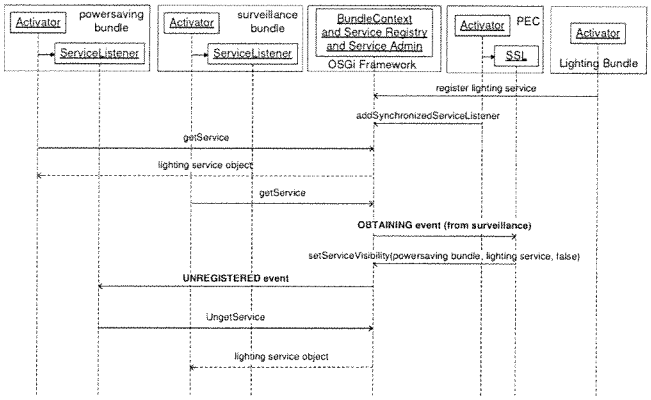


Fig. 5. Sequence diagram showing how the policy enforcement reacts when the surveillance bundle is requesting the lighting service

6 Performance

In paragraph 3 the SynchronousServiceListener was introduced as a way to reduce the performance impact of the models. Having obtaining and released events delivered to more listeners would result in a reduced overall performance as shown in Fig. 6, so delivering to a reduced set of specialized listeners performs better.

A second series of test (cf. Fig. 7, Fig. 8) were performed to analyze the impact of changing the visibility of a service. In the test setup a bundle is measuring the downtime of a service. (The time in ms it cannot use the service). A service is brought down and up by changing the visibility using the ServiceAdmin service (a cycle). The two models are compared against each other. Furthermore they are compared against the situation where the bundle owning the service, unregisters and reregisters the service by using the ServiceRegistration object and the bundle context.

As expected the standard third method, which does not allow service policies, performs worst. When the service was

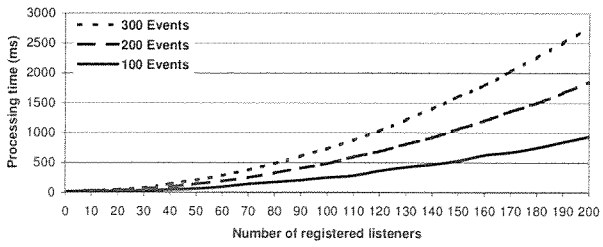


Fig. 6. Measured times needed for the delivery of 100, 200 and 300 obtaining and released events. The actual event handling is not included. The information from table 2 shows that the delivery of 100 events is a realistic amount of events during a peak moment. Furthermore delivery to all listeners is not very scalable

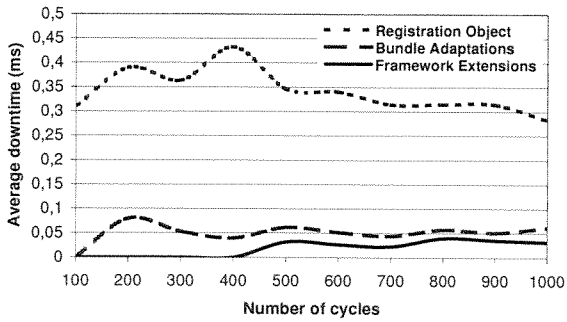


Fig. 7. Average downtime

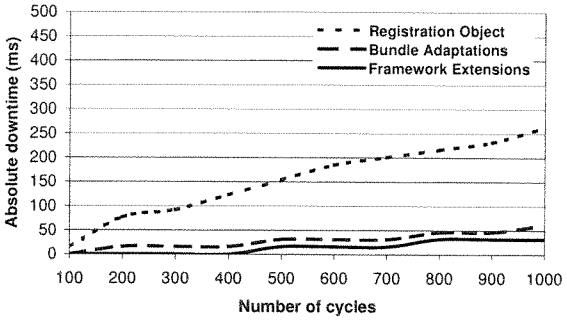


Fig. 8. Absolute downtime

brought down and up a 1000 times, the absolute downtime is more then 250 ms. The average downtime for the standard method is about 0.3 ms. The same test for the bundle adaptations model results in an absolute downtime of 62 ms and an average downtime of 0.05 ms. And finally the best results were obtained using the framework extension model where an absolute downtime of 32 ms and an average downtime of 0.03 ms.

7 Conclusions

This paper indicated the need for component and service integration frameworks in a multi-vendor environment. Furthermore, as shown in the use case, service policy management should not be neglected if one wishes to avoid inconsistent overall system behaviour. The OSGi Service Platform was chosen for its capabilities to integrate components and services from different providers. The platform was analyzed and found insufficient to support dynamic service policies. Therefore two models were presented and evaluated.

Although framework extensions model is more feasible in terms of architectural design, capabilities, performance, transparency and backward compatibility support for legacy bundles, it has one major setback; it requires modifications to the core platform. The proposed extensions to the platform are still within the design philosophy of the OSGi Service Platform and great care is being taken to avoid changes in the OSGi programming model. This approach results in extensions that

do not have any impact on the development of bundles. In fact these extensions are completely transparent to both the providing and the using bundles.
The key requirement that needed to be fulfilled in the bundle adaptations model was backward compatibility with existing platforms. The model was defined as a pluggable set of bundles and can run on any R3 compatible platform. Achieving this goal created a trade-off and resulted in slightly reduced performance, a more complex architecture and the need for bundles to be adapted. Luckily the adaptation can be automated by a tool.
Both models offer a complete set of capabilities to implement a policy management component as demonstrated in paragraph 5. Finally we propose to incorporate the framework extensions within a future release of the OSGi Service Platform.

Acknowledgment

This research has been partly funded by the IBBT-TCASE project [13] which focuses on service delivery to the end-user environment, service and business logic execution and common service capabilities.

References

[1] The Open Services Gateway Initiative, OSGi Service Platform Release 3, IOS Press, Amsterdam, The Netherlands, March 2003. http://www.osgi.org/
[2] The OSGi Alliance, OSGi Service Platform Core Specification Release 4, October 2005. http://www.osgi.org/
[3] The Java Community Process,
JSR 277: JavaTM Module System,
JSR 291: Dynamic Component Support for JavaTM SE,
http://www.jcp.org/
[4] Oscar - An OSGi framework implementation http://oscar.objectweb.org/
[5] Apache Felix Project http://incubator.apache.org/felix/
[6] Humberto Cervantes, Richard S. Hall, Service Binder,
http://gravity.sourceforge.net/servicebinder
[7] The OSGi Alliance, OSGi Service Platform Service Compendium Release 4, October 2005. http://www.osgi.org/
[8] Humberto. Cervantes and Richard .S. Hall. Automating Service Dependency Management in a Service-Oriented Component Model, Proceedings of the Sixth Component-Based Software Engineering Workshop, May 2003, pp. 91-96.
[9] Almut Herzog, Nahid Shahmehri, Problems Running Untrusted Services as Java Threads, In Certification and Security in Inter-Organizational E-Services. E. Nardelli, M. Talamo (eds). Pages: 19-32. Springer Verlag. 2005.
[10] The Java Community Process,
JSR 121: Application Isolation API Specification,
JSR 278: Resource Management API for Java ME,
JSR 284: Resource Consumption Management API,
http://www.jcp.org/
[11] Richard .S. Hall and Humberto. Cervantes. An OSGi Implementation and Experience Report, Proceedings of the IEEE Consumer Communications and Networking Conference, January 2004.
[12] The Knopflerfish Project,
http://www.knopflerfish.org/
[13] IBBT, The Interdisciplinary institute for BroadBand Technology,
http://www.ibbt.be/

SERP'06

The 2006 International Conference on Software Engineering Research &Practice

Foreword Author's Index

Session: SOFTWARE TESTING AND QUALITY ASSURANCE

A Framework for Automatic Testing of Industrial Controller Code
Dag Kristiansen, Karl–Petter Lindegaard
Agile Test–based Modeling
Bernhard Rumpe
Statistical Analysis and Enhancement of Random Testing Methods also under Constrained Resources
Johannes Mayer, Christoph Schneckenburger
DPTModel: The Defect Prevention and Traceability – Driven Model for Software Engineering
Jay Xiong, Jonathan Xiong
Selecting Effective Test Messages
Len Gebase, Roch Bertucat, Robert Snelick
Distributed Tool for Performance Testing
nenad stankovic
Test–bed for Verification and Validation Activities in Developing an Operations Support System
Dae–Woo Kim, Hyun–Min Lim, Sang–Kon Lee
Generation of Test Scenarios from Use Cases
Stephane Some
Restricted Adaptive Random Testing by Random Partitioning
Johannes Mayer
DPTMethodology: The Defect Prevention and Traceability – Driven Methodology for Software Engineering
Jay Xiong, Jonathan Xiong
The DPTSystem: The Defect Prevention and Traceability – Driven System for Software Engineering
Jay Xiong, Jonathan Xiong
Dynamically Generating Conformance Tests for Messaging Systems
Robert Snelick, Len Gebase, Sydney Henrard
Adapting Structural Testing to Functional Programming
Manfred Widera
Critical Systems and Software Risk to Public Safety: Issues and Research Directions

Shreedevi Inamdar, Hisham Haddad

Software Quality and Testing

Hassan Pournaghshband, Asaleh Sharifi, Shahriar Movafaghi

A Method for Generating a Minimal Functional Set of Test–Cases for Software–Intensive Systems

Joerg Gericke, Matthias Wiemann

Looking at Comparisons of Regression and Analogy–based Software Project Cost Prediction

Carolyn Mair, Martin Shepperd

An Efficient Slicing Approach for Test Case Generation

Durvasula V L N Somayajulu, Ajay Kumar Bothra, Prashant Kumar, Pratyush Pratyush

Impact of Using Test–Driven Development: A Case Study

Sumanth Yenduri, Louise Perkins

Multi Dimension Quality Model of MAS

Punam Bedi, Vibha Gaur

Session: SOFTWARE REUSE

Reusing Families Design

Virginia de Paula

Reuse and Component Based Development (CBD)

Rizwan Jameel

Retrieval of Most Relevant Reusable Component Using Genetic Algorithms

Rajesh Bhatia, Mayank Dave, RC Joshi

A Reuse–Oriented Process Component Representation Framework

Xiaohong Yang, Jing Lu, Ruzhi Xu, Guangfeng Pan, Jin Liu

Effective Reuse Procedure for Open Source Software

Doo Yeon Kim, Jong Bae Kim, Sung Yul Rhew

Reuse – A Management View

Danny Ho

Study of Information Retrieval Systems and Software Reuse Libraries

Usa Rungratchakanon, Hisham Haddad

Session: SOFTWARE METHODOLOGIES, PROCESS, AND MODEL ORIENTED DESIGN

An Object–Oriented Framework for Predicting Student Competency Level in an Incoming Class

Suresh Kalathur

An Experience Report of Applying the Personal Software Process Methodology

Wen–Hsiang Shen, Nien–Lin Hsueh, Peng–Hua Chu

Automatic Code Generation: Model–Code Semantic Consistency

Andrew Kornecki, Sona Johri

A Graph-Based Representation of Object-Oriented Designs
Wei Li, Huaming Zhang, Raed Shatnawi
Modeling Timed Automata Theory in PVS
Qingguo Xu, Huaikou Miao
Model-Based XML Editor Generation
Jong-Myung Choi, Soo-Lyul Oh, Dong-Soon Ahn, Jong-Hwa Kim, Kyung-Woo Park, Han-Suk Choi, Hea-Sang Shin
A Feature Oriented Approach to Mapping from Domain Requirements to Product Line Architecture
Chongxiang Zhu, Yuqin Lee, Wenyun Zhao, Jingzhou Zhang
Adapter Pattern in Component and Service Levels vs. Class and Object Levels
Kai Qian, Larry Wang, Subramanian Ananthram
Success Factors of Agile Software Development
Subhas Misra, Vinod Kumar, Uma Kumar
SEM2XPDL: Towards SPEM Model Enactment
Feng Yuan, Mingshu Li, Zhigang Wan
Software Process Improvement In Bangladesh
Bernard Wong, Sazzad Hasan
Analysis of Object-Oriented Numerical Libraries
Kostas Zotos, George Stephanides

Session: SOFTWARE REQUIREMENT ANALYSIS

Requirements Engineering for E-Voting Systems
Kevin Daimi, Katherine Snyder, Robert James
Automatic Comprehension of Textual User Requirements and their Static and Dynamic Modeling
Olga Ormandjieva, Magda Ilieva
A Multi-Role Collaborative Method and Platform for Developing Software Requirements
Chin-Yi Tsai, Chua-Huang Huang
A Course Design and Implementation Experience on Agile Software Development Methodologies
Hongxing Lu, Xiaohong (Sophie) Wang
Software Development with Automatic Code Generation: Observations from Novice Developer Viewpoint
Farahzad Behi, Andrew Kornecki
The Factors of Software Systems that Contribute to Requirements Elicitation
Allison Scogin

Session: SOFTWARE ARCHITECTURE, DESIGN PATTERNS, AND FRAMEWORKS

EJB Performance Measurement Framework

Denis Gefter, Robert Chun

Analyzing Communication Patterns in Software Engineering Projects

H. Keith Edwards, Robert R. Puckett, Art Jolly

A SOA-Based IA Asset Management Architecture Using XML in E-Government

Namho Yoo, Hyeong-Ah Choi

OSGi Service Layer Enhancements

Nico Goeminne, Gregory De Jans, Jan Hollez, Bart Dhoedt, Filip De Turck, Frank Gielen

Using Webservice Choreography and Orchestration Perspectives to Model and Evaluate B2B Interactions

Andreas Schönberger, Guido Wirtz

Updating Software Architectures: A Style-Based Approach

Dalila Tamzalit, Mourad Oussalah, Olivier Le Goaer, Abdelhak-Djamel Seriai

Towards a Layered Architectural Design of a Persistence Framework

Sai Peck Lee, Tong Ming Lim, Ho-Jin Choi

Pattern-Oriented Design for Multi-Agent System: A Process Framework

Radziah Mohamad, Safaai Deris, Hany Ammar

The Role of Model-Oriented Software Architecture in Safety Engineering

Hassan Reza, Emanuel Grant

Session: DISTRIBUTED AND REAL TIME SYSTEMS

Application Platforms for Embedded Systems: Suitability of J2ME and .NET Compact Framework

Koen Victor, Yves Vandewoude, Yolande Berbers

Practical Technologies for Implementing Distributed Applications as Evolvable Software Systems (ESS)

Kendall Conrad, Vincent Schmidt

Comparison of Object Oriented Technology Automatic Codes Generating Tools for Safety Critical Real-time Software

Farahzad Behi, Daniel Penny III

Experiences in Distributed Software Development with Wiki

Khalid Al-asmari, Liguo Yu

Interlocutor System

Edson Barros, Roseli Lopes

Compositional Abstraction for Concurrent Programs

Junyan Qian, Baowen Xu

Transformation of the Ravenscar Profile Based Ada Real-time Application to the Verification-ready Statecharts : Reverse Engineering and Statemate approach

Chang Jin Kim, Jin-Young Choi

Session: SOFTWARE MAINTENANCE

An Effort Estimation by UML Points in Early Stage of Software Development

SangEun Kim, William Lively, Dick Simmons

Predicting Error Probability in the Eclipse Project
Raed Shatnawi, Wei Li, Huaming Zhang
Are the Changes Induced by the Defect Reports in the Open Source Software Maintenance?
Timo Koponen, Heli Lintula
A Model of Maintainability – Suggestion for Future Research
Mira Kajko-Mattsson, Gerardo Canfora, Dan Chiorean, Arie van Deursen, Tuomas Ihme, Meir M Lehman, Rupert Reiger, Torsten Engel, Josef Wernke
An Entropy-Based Approach to Assessing Object-Oriented Software Maintainability and Degradation -- A Method and Case Study
Hector Olague, Letha Etzkorn, Glenn Cox
A Software Traceability Validation For Change Impact Analysis of Object Oriented Software
Suhaimi Ibrahim, Norbik Idris, Malcolm Munro, Aziz Deraman
A Comparison of the Efficiencies of Code Inspections in Software Development and Maintenance
Liguo Yu, Robert Batzinger, Srini Ramaswamy

Session: SOFTWARE METRICS, CONFIGURATION AND PROJECT MANAGEMENT

Virus Removal Cost (VRC) Metric
Kuangnan Chang, Bobby Adkins
Towards an Extendable Software System for Information Integration
Paul Whitney, Christian Posse, Xingye Lei
A Workbench for Learning Enterprise Patterns
Paulo Sousa
Web Metrics: The way of improvement of quality of Non web-based systems
Shazia Arshad, Muhammad Shoaib, Abad Shah
Effect of Human Behavior in SDLC
Ashmeet Kaur, Ritu Soni
On the Role of Software Metrics in Applying Design Patterns
Niloofar Khedri, Masoud Rahgozar, Mahmoud Reza Hashemi
A Qualitative Study on PATT – A Project Assessment and Tracking Tool
Fabio Marzullo, Geraldo Xexéo
Computations with Large Numbers
Weihu Hong, Mingshen Wu

Session: UML, MDA, ...

On the Effectiveness of Source Code Transformations for Binary Obfuscation
Matias Madou, Bertrand Anckaert, Bruno De Bus, Koen De Bosschere, Jan Cappaert, Bart Preneel
Model Driven Development with Interactive Use Cases and UML Models

Paul Nguyen, Robert Chun

Medical Informatics and Medical Databases Approach in Modeling Healthcare Education System with Unified Modeling Language (UML)

Anil Khatri, Azene Zenebe, David Anyiwo

Model Transformation Based on Meta Templates

Hongming Liu, Lizhang Qin, Xiaoping Jia, Adam Steele

Using UML to Develop Verifiable Reactive Systems

S. Fatemeh Alavizadeh, Marjan Sirjani

Developing Medical Information System with MDA and Web Services

Simone A. B. Melo, Denivaldo Lopes, Zair Abdelouahab

UML Analysis Using State Diagrams

Mohammad Alanazi, Jason Belt, David Gustafson

Session: COMPONENT ORIENTED SOFTWARE DEVELOPMENT

Plugin-Based Systems with Self-Organized Hierarchical Presentation

Boto Bako, Andreas Borchert, Norbert Heidenbluth, Johannes Mayer

Algorithms for Optimally Tracing Time Critical Programs

Sergej Alekseev

Assessment of Component-Based Systems with Distributed Object Technologies

Jiang Guo, Yuehong Liao, Xichun Pei

A Java Instrumentation-based Analysis Approach for the Dynamic Behaviors of J2EE Applications

Yuehong Liao, Jiang Guo, Xichun Pei

SoCoEMo-COTS: A Software Economic Model for Commercial Off-the-shelf (COTS) Based Software Development

Sana Ben Abdallah Ben Lamine, Lamia Labed Jilani, Henda Hajjami Ben Ghezala

Conceptual Model for Integration of COTS Components

James Tollerson, Hisham Haddad

Process Component Plug-in Approach

Jin Myung Choi, Sung Yul Rhew

Session: FORMAL METHODS AND SPECIFICATION LANGUAGES, AND LANGUAGE DESIGN

Inspection of Concurrent Systems: Combining Tables, Theorem Proving and Model Checking

Vera Pantelic, Xiao-Hui Jin, Mark Lawford, David Parnas

On a GUI-based Editor for Z Specifications and its Applications

Hiroshi Ishikawa

A Formally Verified Geometric Modelling Core

Catherine Dubois, Jean-Marc Mota

Formal Verification of a Simple Automated Negotiation Protocol
George Dimitoglou, Okan Duzyol, Lawrence Owusu
Re-Engineering BLUE Financial System Using Round-Trip Engineering and Java Language Conversion Assistant
Salem Al-Agtash, Tamer Al-Dwairy, Adnan EL-Nasan, Bruce Mull, Mamdouh Barakat, Anas Shqair
A Base for Achieving Semantics for Prolog with Cut for Correct Observables
Lingzhong Zhao, Tianlong Gu, Junyan Qian, Guoyong Cai
Comparison of the Modeling Languages Alloy and UML
Yujing He
Supporting Separation of Concerns to Automation of Code Generation
Paniti Netinant
A Software Specification Language for RNA Pseudoknots
Keum-Young Sung
The Intelligent C Language Debugger
Ming Wang, Robert Chun

Session: CASE STUDY, USABILITY ENGINEERING, AND EDUCATION

Integrating User Centered Design in a Product Development Lifecycle Process: A Case Study
Karsten Nebe, Lennart Groetzbach, Ronald Hartwig
Learner-centered Technical Review in Programming Courses
Hongxing Lu, Xiaohong (Sophie) Wang
Development of an Ant Script Builder with Thought to Usability and Best Practices
Kalyana Gundamaraju, Michael Wainer
Service Learning, Software Engineering, and Hurricane Katrina – A Case Study
Donald Schwartz, Jonathan Spencer, Adam Huffman
Podcasts: Changing the Face of e-Learning
Saby Tavales, Sotirios Skevoulis

Session: SOFTWARE RELIABILITY MODELS AND RISK ANALYSIS

Supporting Software Fault Tree Analysis Using a Key Node Metric
Donald Needham, Sean Jones
Metrics in Risk Determination for Large-Scale Distributed Systems Maintenance
Maureen Raley, Letha Etzkorn

Session: 5TH INTERNATIONAL WORKSHOP ON SYSTEM/SOFTWARE ARCHITECTURES, IWSSA'06

Ontology-Driven Middleware for Next-Generation Train Backbones

Stijn Verstichel, Sofie Van Hoecke, Matthias Strobbe, Steven Van den Berghe, Filip De Turck, Frederik Vermeulen, Piet Demeester

System Modeling for Systematic Development of Groupware Applications

Manuel Noguera, Miguel González, José Luis Garrido, María Visitación Hurtado, María Luisa Rodríguez

Organization Modelling to Support Access Control for Collaborative Systems

Francisco Luis Gutierrez, Jose Luis Isla, Patricia Paderewski, Miguel Sanchez

An NFR–Based Framework for Aligning Software Architectures with System Architectures

Nary Subramanian, Lawrence Chung

Architecture–Centric Program Transformation for Distributed Systems

Chung–Horng Lung, Jianning Liu, Xiaoli Ling, Dan Jiang

Component–Aware System Architecting: A Software Interoperability

Weimin Ma, Kendra Cooper, Lawrence Chung

Position Paper: From Enterprise Architectures to Software Architectures using Requirements Engineering

Matthias Galster, Armin Eberlein, Mahmood Moussavi

Helping to Meet the Security Needs of Enterprises: Using FDAF to Build RBAC into Software Architectures

Lirong Dai, Kendra Cooper

Modeling of Evolution to Secure Application System: from Requirements Model to Software Architecture

Michael Shin

An Enterprise Architecture Process Model

François Coallier, Roger Champagne

A Model of Access Control for Data Materials Based on Ambient Calculus

Masaki Murakami

Session: PROCEEDINGS OF PLC'06 – DATA–FLOW ANALYSIS

A Fine–Grained Analysis of the Performance and Power Benefits of Compiler Optimizations for Embedded Devices

Jason W.A. Selby, Mark Giesbrecht

Complexity of Data Flow Analysis for Non–Separable Frameworks

Bageshri Sathe, Uday Khedker

Session: PROCEEDINGS OF PLC'06 – CODE OPTIMIZATION AND COMPILER GENERATION TECHNIQUES

Experience in Testing Compiler Optimizers Using Comparison Checking

Masataka Sassa, Daijiro Sudosa

Deterministically Executing Concurrent Programs for Testing and Debugging

Steve MacDonald, Jun Chen, Diego Novillo

Compiler Generator for Creating MOF–compliant Source Code Models

Zoltán László, Tibor Sulyán

An Embedded Haskell Subset Implementation

Ian Lewis

User–Friendly Methodology for Automatic Exploration of Compiler Options: A Case Study on the Intel XScale Microarchitecture

Haiping Wu, Eunjung Park, Long Chen, Juan del Cuvillo, Guang R. Gao

A User–Friendly Methodology for Automatic Exploration of Compiler Options

Haiping Wu, Long Chen, Joseph Manzano, Guang R. Gao

Session: PROCEEDINGS OF PLC'06 – LOGIC, FUNCTIONAL, MODELING, NEW PROGRAMMING PARADIGMS

Implementation of Tag Representation in Prolog Virtual Machine

Guillaume Autran, Xining Li

XML Markup Languages Framework for Programming in 21st Century towards Managed Software Engineering

Khubaib Ahmed Qureshi, M Zeeshan Ali Ansari

Improved Graph–Based Lambda Lifting

Marco T. Morazan, Barbara Mucha

On Petri Nets and Predicate–Transition Nets

Andrea Röck, Ray Kresman

IncH: An Incremental Compiler for a Functional Language

James Gil de Lamadrid, Jill Zimmerman

Extensible and Adaptable System Software

Paniti Netinant

Session: PROCEEDINGS OF PLC'06 – REGISTER ALLOCATION, MEMORY MANAGEMENT, AND OO TECHNIQUES

Efficient and General On–Stack Replacement for Aggressive Program Specialization

Sunil Soman, Chandra Krintz

Java Virtual Machine: the key for accurated memory prefetching

Yolanda Becerra, Jordi Garcia, Toni Cortes, Nacho Navarro

Evaluation Issues in Generic Programming with Inheritance and Templates in C++

Emil Vassev, Joey Paquet

String Concatenation Optimization on Java Bytecode

Ye Henry Tian

Aspects of Memory Management in Java and C++

Emil Vassev, Joey Paquet

The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing

Monte Carlo Resort, Las Vegas, Nevada, USA
June 26–29, 2006

Conferences:

The 2006 International Conference on Bioinformatics & Computational Biology

The 2006 International Conference on Computer Design & International Conference on Computing in Nanotechnology

The 2006 International Conference on Computer Graphics & Virtual Reality

The 2006 International Conference on Communications in Computing

The 2006 International Conference on Scientific Computing

The 2006 International Conference on Data Mining

The 2006 International Conference on e–Learning, e–Business, Enterprise Information Systems, e–Government, & Outsourcing

The 2006 International Conference on Engineering of Reconfigurable Systems & Algorithms

The 2006 International Conference on Embedded Systems & Applications

The 2006 International Conference on Foundations of Computer Science

The 2006 International Conference on Frontiers in Education: Computer Science & Computer Engineering

The 2006 International Conference on Grid Computing & Applications

The 2006 International Conference on Artificial Intelligence

The 2006 International Conference on Internet Computing & International Conference on Computer Games Development

The 2006 International Conference on Wireless Networks

The 2006 International Conference on Information & Knowledge Engineering

The 2006 International Conference on Image Processing, Computer Vision, & Pattern Recognition

The 2006 International Conference on Machine Learning; Models, Technologies & Applications

The 2006 International Conference on Modeling, Simulation & Visualization Methods

The 2006 International Conference on Parallel & Distributed Processing Techniques & Applications & International Conference on Real-Time Computing Systems & Applications

The 2006 International Conference on Pervasive Systems & Computing

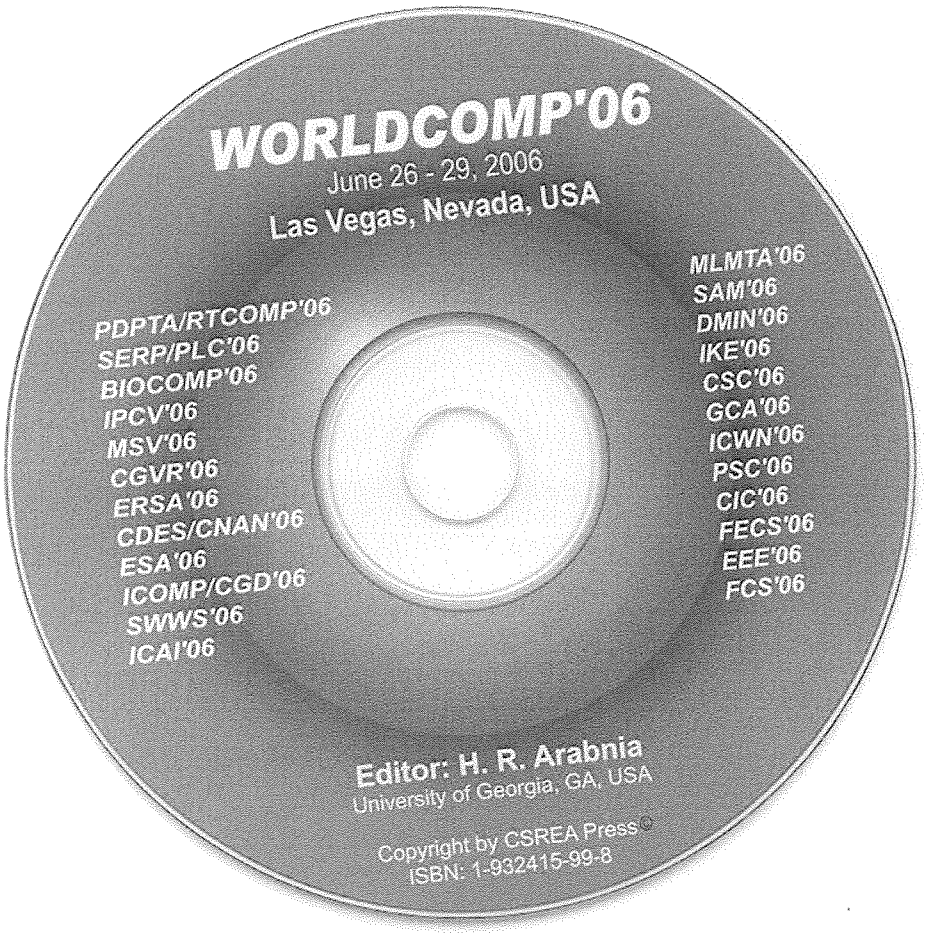
The 2006 International Conference on Security & Management

The 2006 International Conference on Software Engineering Research & Practice & International Conference on Programming Languages and Compilers

The 2006 International Conference on Semantic Web & Web Services

Editor H.R. Arabnia
University of Georgia, GA, USA

ISBN: 1-932415-99-8


WORLDCOMP'06
June 26 - 29, 2006
Las Vegas, Nevada, USA
PDPTA/RTCOMP'06
SERP/PLC'06
BIOCOMP'06
IPCV'06
MSV'06
CGVR'06
ERSA'06
CDES/CNAN'06
ESA'06
ICOMP/CGD'06
SWWS'06
ICAI'06
MLMTA'06
SAM'06
DMIN'06
IKE'06
CSC'06
GCA'06
ICWN'06
PSC'06
CIC'06
FECS'06
EEE'06
FCS'06
Editor: H. R. Arabnia
University of Georgia, GA, USA
Copyright by CSREA Press ©
ISBN: 1-932415-99-8