

# Rulespect: Language-Independent Rule-Based AOP Model for Adaptable Context-Sensitive Web Services

Syed Saif ur Rahman, Ateeq Khan, and Gunter Saake

Department of Technical and Business Information Systems, Faculty of Computer Science,  
University of Magdeburg, Magdeburg, Germany  
{srahman, ateeq, saake}@ovgu.de

**Abstract.** Business domain has always been competitive. Adaptability to changes in business processes is mandatory to keep business organizations ahead in competition. Information system is a fundamental tool to manage business processes. Information system should be adaptable to accommodate changes in business processes and should be context-sensitive to give more personalized perspective to information system user. Web services and aspect-oriented programming possess good potential for adaptable context-sensitive information system development. However, main obstacle in the widespread adoption of aspect-oriented programming paradigm in information system domain is verbose and alien syntax of implementation languages that make it difficult to use. To solve this problem we propose *Rulespect*, a language-independent rule-based aspect-oriented programming model for adaptable context-sensitive web services.

## 1 Introduction

Enterprise businesses are managed by enterprise business information systems (ISs). The integration of an enterprise business with an IS raises the need for adaptation of changes to be performed at an IS level concurrently as they occur in an enterprise business. Such adaptation in an IS is incorporated as workflows [22], or rules that surround executing business processes (BPs) and activities.

Advent of web services [2], service-oriented computing (SOC) [30], workflow management systems [22], and rules execution engines enabled high interoperability and adaptability for BPs at an IS level. In this paper, we will discuss the IS developed using SOC where each BP or activity can be implemented as a web service. In such an IS, to adapt the change in BPs new web services can be instantiated or existing web services can be modified. Aspect-oriented programming (AOP) [20] can be used in conjunction with web services to develop highly adaptable IS. However, main obstacle in the widespread adoption of AOP paradigm in IS domain is verbose and alien syntax of implementation languages that make it difficult to use [8, 21]. Rule-based AOP model is presented in this paper with a proof-of-concept, in which rules are used in dynamic join-point model. Rulespect aims to simplify AOP use in IS development. Rule is a fundamental concept for business domain. Rulespect enables IS user to use familiar construct of rule to adapt IS. We also discuss how rulespect mechanism can benefit IS developer to bring context-awareness at web service level.

This paper is structured as the related concepts are explained in section 2. Section 3 and 4 gives related work and motivation. Section 5 explains the rulespect AOP model,

II section 6 and 7 give the implementation details of model using an example. Section 8 concludes the paper with some hints for future work.

## 2 Related Concepts

### 2.1 Aspect-oriented Programming

Traditional programming languages implementations typically lead to code tangling, scattering, and replication [20, 36, 4], which reduces code readability and maintainability. Code scattering refers to the code belonging to one concern scattered across multiple modules; code tangling refers to the code belonging to multiple concerns mixed in one module; code replication refers to multiple code fragments in one program that are equal or similar. Modularity and clean separation of concerns helps in improving code readability and maintainability [36].

Concerns are goals or objectives of any given piece of code or software. Concerns can be classified as core concerns or cross-cutting concerns as show in Figure 1 [4]. Crosscutting concerns can be further classified as either static/dynamic [25] or homogeneous/heterogeneous [11]. Static cross-cutting concerns affect the static structure of a program by adding new classes and interfaces, by injecting new methods or fields, and by declaring new super-classes and interfaces [25]. Dynamic cross-cutting concerns crosscut the dynamic computation of a program and thus can be defined in terms of events and actions [40]. Homogeneous cross-cutting concerns affect multiple join points and apply one piece of code, i.e., the same extension to all join points [11]. In contrast, heterogeneous cross-cutting concerns apply different pieces of code to different join points [11]. Examples of concerns could include security, persistence, transaction processing, etc. Aspect-oriented programming [20] is methodology that emerged

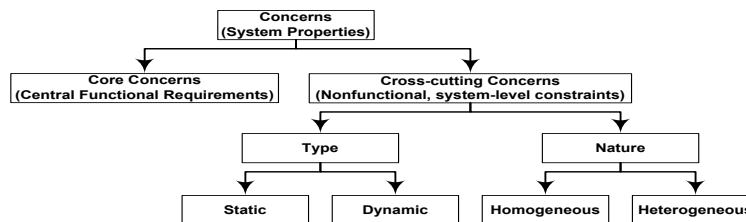


Fig. 1. Concern classification.

with the aim to separate cross cutting concerns and forms the basis for Aspect-Oriented Software Development <sup>1</sup> (AOSD) [37]. AOP suggest implementing orthogonal features as aspects to prevent code tangling and scattering [5] ensuring code scalability and maintenance. A feature can be defined as ”A distinguishable characteristic of a concept that is relevant to some stake-holder” [17].

<sup>1</sup> AOSD, <http://www.aosd.net>

III

An aspect weaver uses join-point specification to bring together aspects with components/classes [6]. Join points are points in the execution of a program [19], e.g., call to a method during execution. Pointcut (a predicate on attributes of join points) notation is introduced to enable selection of related method execution points, e.g., execution, call, etc. A pointcut may refer to multiple join points [19]. AOP supports dynamic cross-cutting using point-cut specification. AOP is able to implement non-hierarchy-conform refinements and can avoid excessive method shadowing by using wildcards in pointcut expressions [5]. An advice is used to specify the behavioral effect at join points identified by a pointcut [19]. It can be executed before, after, or at both points (i.e. around). Pointcuts and advice together dynamically affect the program flow. Integrating aspects into the execution of the base functionality is called weaving [6]. With support for dynamic weaving [7], aspects can be activated/deactivated at runtime. AspectJ [19] and AspectC++ [35] are most famous AOP extensions for Java and C++.

## 2.2 Web Services

Web service is a standard defined by W3C<sup>2</sup> to support interoperability between interacting systems over internet. According to the concepts defined in W3C standards, systems interacting via web services communicate using XML messages based on SOAP (Simple Object Access Protocol) standard<sup>3</sup>. Web service is based on message passing interactions. Interacting systems expose the functionalities using the WSDL<sup>4</sup> (Web Service Description Language) according to W3C recommendation [1]. Web service possesses good potential in BP domain because of simplicity and interoperability. Many questions, regarding their performance, security, or interoperability, are yet not answered [23, 38, 31]. Web services has become an important IS development technology and numerous organizations now want to use them to develop highly interoperable IS [1].

## 2.3 Workflow and Rules

Workflow is a sequence of operations, activities, or business processes that are executed in some predictable order according to defined rules [32]. In software engineering terminology, workflows allow separation of business logic from program functionalities resulting in system scalability, maintainability, and adaptability [32]. Workflows are governed by rules. We can define rules in the same format as we define if-else statement in programming languages or more precisely in Event-Condition-Action (ECA) format using declarative approach.

---

<sup>2</sup> The World Wide Web Consortium (W3C), <http://www.w3.org/>

<sup>3</sup> Simple Object Access Protocol (v.1.2). Published on Web (April 2007). <http://www.w3.org/TR/soap12-part1/> Accessed: 29-06-2009

<sup>4</sup> Web Service Definition Language (v.1.1). Published on Web (March 2001). <http://www.w3.org/TR/wsdl> Accessed: 29-06-2009

Product	Weaving	Mechanism for aspect definition
Aspect#	Dynamic	Built-in Language
AspectDNG	Static (CIL Manipulation)	X Path, RegularExpression
EOS	Static (Preprocessor)	Built-in Language
Encase	Dynamic	Simple Object Model
Gripper-LOOM.NET	Static (Preprocessor)	Simple Object Model
Rapier-LOOM.NET	Dynamic	Simple Object Model
Nkalore Compiler	Static (Preprocessor)	Built-in Language
Seaser S2 Container .NET	Dynamic	Simple Object Model
SetPoint	Dynamic	Syntax, Semantic
Spring .NET AOP	Dynamic	Simple Object Model
AspectJ	Static (Preprocessor)	Built-in Language
AspectC++	Static (Preprocessor)	Built-in Language
CaesarJ	Dynamic	Built-in Language
EAOP	Dynamic	Simple Object Model
JAC	Dynamic	Built-in Language
PROSE	Dynamic	Simple Object Model
JBoss AOP	Dynamic	Simple Object Model
AspecWerkz	Dynamic	Simple Object Model
Jiazzi	Static (Preprocessor)	Built-in Language
JASCO	Dynamic	Built-in Language

**Table 1.** AOP tools comparison.

### 3 Related Work

Cibrán and Verheecke promoted modularizing web services management with AOP [10, 39]. Extensive research in aspect-oriented component engineering for web services design and development is carried out by Grundy et al. [13, 14]. Charfi et. al. has proposed aspect-oriented BPEL (AO4BPEL), an extension to the BPEL language. AO4BPEL advocate an aspect-oriented approach for web service composition. This extension brings modular and dynamic adaptability to BPEL [9]. An aspect-oriented web service composition solution is developed by Ortiz et al. [27–29]. Arsanjani et al. in [24] have identified the suitability of AOSD to modularize the heterogeneous concerns involved in web services. AWED [26] is one dynamic aspect model which supports distributed nature of services. But it does not support a high-level description of join points based on system states. As a result, specification of cross-cutting concerns of composite services is difficult in this model. Keidl et al. [18] presented a context framework to facilitate development and deployment of context-aware adaptable web services. Singh et al. [34] presented an architecture for aspect-oriented web services (AOWS) to address the problems in the area of web service description, discovery, and integration. Irmert et al. [16] presented an approach to integrate dynamic AOP into a service-oriented architecture platform to make existing services adaptable at runtime. As we have already listed above, there are many research results in web services domain that advocate use of AOP in conjunction with web services, however, all of them use separate technology to implement AOP constructs. In rulespect, we advocate use of existing IS development technologies like rules to implement AOP constructs.

One of the main drawbacks associated with AOP is its difficulty of use because of verbose syntax [21]. To tackle this issue there have been many attempts by researchers. New languages have been built to declare and configure aspects. One example of such domain specific language is LENDL [3]. XPath and RegularExpression have been used

as join point language in AspectDNG<sup>5</sup>. New construct like classpects have been proposed [33]. SetPoint attempted to use the semantic pointcuts [3]. Table 1 shows the existing AOP implementations and their comparison regarding the mechanisms they use for aspect definition. Event-based Aspect-Oriented Programming (EAOP) [12] is the model that is close to the theme of Rulespect, but it also lacks use of rules. Furthermore, none of the above-mentioned approaches support AOP implementation at web service level, instead they implement AOP for software artifacts instantiated by web services, i.e., components instantiated by web service. In contrast, rulespect enables AOP implementation at web service level, i.e., join point for web service method call. Rulespect model is unique in its concept and implementation from current state of the art.

## 4 Motivation

In the introduction we already have provided a strong motivation over the need for adaptable context-sensitive web services for adaptable IS. These web services contain many concerns such as logging, security, persistence, transaction processing, etc. which spread over multiple services cross-cutting them. These cross-cutting concerns should be handled in modular fashion. Related work proposes the use of AOP technology for developing adaptable context-sensitive web services and efficiently handling cross-cutting concerns among them. It also gave a detailed review of existing AOP tools. We argue that all tools have possessed difficulty of use because of different languages and object models. All tools require special syntax for AOP artifacts definition. These special syntaxes are verbose [21] and domain specific. From IS user point of view, all these syntaxes are alien. If commonly used programming syntax can be used for defining these artifacts, problem of difficulty in use of AOP can be resolved.

Rulespect is an AOP model that can be used to separate cross-cutting concerns from web services using rule-base approach. Rule is a fundamental concept in business IS domain and is in widespread use for adaptability. Rulespect make use of simple rules in ECA form for specifying weaving details. As most of the IS domain users are well versed with rule definition and adaptation, its use in AOP model promises widespread use of AOP technology in this domain. Most important innovation is to make use of existing IS development technologies for implementing the model. There is no need to have specialized engine to implement rulespect, design is easy to understand, and defined in platform and language independent manner. For implementation, any existing programming language and rule execution mechanism will be sufficient. The motivation of this research is two fold. First, we want to reduce the complexity of the AOP syntax by using easy to use rule-base syntax. Second, as rule-based systems and web services are common in use for IS development, we demonstrate how adaptable context-sensitive web services can benefit from this model.

---

<sup>5</sup> <http://aspectdng.tigris.org/>

## VI 5 Rulespect: Rule-based AOP Model

Rulespect, the Rule-based AOP model is motivated by the use of rules for specifying the logic for aspect weaving into the main code. It is specially designed for business domain and web services. For better understanding, rulespect tries to simplify the concepts involved in AOP with simple terms.

*Defining An Entity Of Interest:* Very first step in rulespect AOP model is definition of an entity of interest. An entity of interest means an entity that contains the main code. It could be a method in a class, a web method in a web service, or it could be an entity at coarser or finer granularity. By defining an entity of interest we means to tag an entity. In our implementation we used annotation-based approach for tagging, however, user can use other mechanism suitable for their implementation. This tagging enables the model to track the events when execution of program reaches to a point where it calls and executes the code of entity of interest. Rulespect uses these events as join point.

*Joint Point Model:* In AOP, joint point model indicates the points where new behaviors could be included in the main code. For any join point, rulespect allow defining before, after, and around advice. Rulespect allows defining rules for every combination of an entity of interest, join points, and advices. If no rule is defined, then no weaving will be done. Pointcuts defines at which point of execution we want to insert the new code (i.e. advice). An advice corresponds to the code that is executed when its associated rule conditions are satisfied.

*Rules In Joint Point Model:* Rules are based on Event, Condition, Action, and ElseAction. Rules are triggered based on an event and are executed based on conditions. Condition also serves the purpose of precise pointcut identification for selected join points. Action and ElseAction performs the advice weaving. Rulespect performs weaving dynamically. Weaver weaves the cross-cutting code (i.e. advice) to original code (i.e. entity of interest). If no condition or always true condition is defined, it means that action will always be executed. If conditions are not satisfied, this model also supports execution of alternative behavior defined using ElseAction.

*Multiple Rules For Single Pointcut:* Rulespect suggest the possibility to define multiple rules for single pointcut. Rulespect allows to enable/disable or add/remove rules dynamically. Rules can be prioritized to ensure more control over advice weaving and execution. Rulespect allows the removal of rules at any time during the execution of program, however, there also exists the possibility of disabling the unused rules.

*Workflow For Single Pointcut:* Rulespect also suggest the possibility to define a complete workflow for single pointcut. Using workflow we can define a complete set of actions and rules that should be executed for weaving advice in main code. This approach gives more flexibility and control on weaving logic.

Rulespect model is show in Figure 2 which shows that rulespect keeps clear separation between main functionality, cross-cutting functionality, and rules that contain the

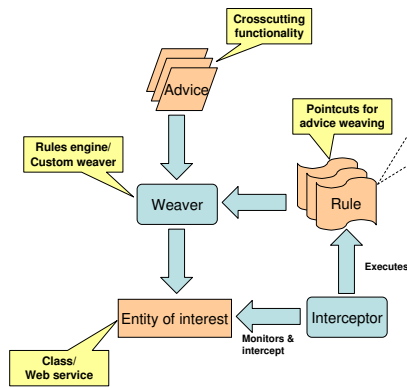


Fig. 2. Rulespect: Rule-based AOP model.

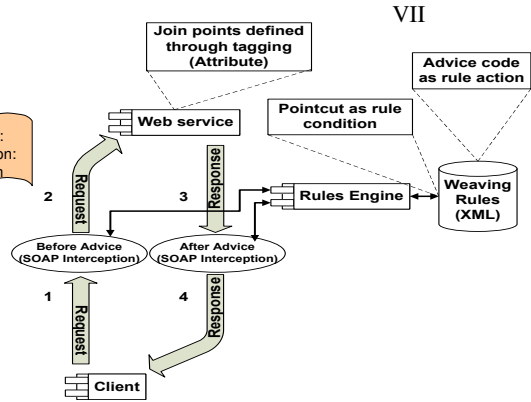


Fig. 3. General Rulespect implementation in .NET.

weaving details. An important feature of rulespect that separate it from other AOP implementation is its applicability for web services. Existing AOP implementations do not support aspect weaving for web methods (i.e., methods exposed by web service). In rulespect, aspect weaving for web method is also possible. Secondly, web services are based on message passing mechanism. In rulespect, we kept a general step of defining the entity of interest. For web services, it means interception of SOAP messages whereas for OOP classes it could be the interception of method or variable call. Use of multiple rules enables the weaving to be performed intelligently, i.e., we can have set of rules which can decide how weaving can be performed based on heuristics. Use of workflow enables the possibility of well controlled execution of behavior that might be needed for the execution of weaving code, e.g., context-based aspects.

## 6 The Environment At a Glance

Currently several platforms for web service development exist, the most commonly used platforms are J2EE and .NET [15]. .NET platform is used to demonstrate the proof of concept in this research. .NET framework<sup>6</sup> is Microsoft proprietary managed code programming model for application development. It supports syntax for multiple programming languages including C#, VB.NET, J#, and C++ all generating Common Intermediate Language (CIL) on compilation. .NET platform provides excellent support for software extensibility, adaptability, maintainability, and customizability through various techniques including reflection, proxy, intermediate language, and on-fly code generation.

In .NET environment, we used workflows to define the rules. In demonstration workflows are implemented using Windows Workflow Foundation (WF)<sup>7</sup>. WF is Microsoft proprietary programming model to develop and deploy workflows in applica-

<sup>6</sup> Microsoft .NET, [www.microsoft.com/net](http://www.microsoft.com/net) Accessed: 29-06-2009

<sup>7</sup> Windows Workflow Foundation (WF), <http://msdn.microsoft.com/en-us/netframework/aa663328.aspx> Accessed: 29-06-2009

VIII  
tions. It supports authoring, executing, and managing workflows. WF consists of an activity model, workflow designer, workflow runtime, and rules engine. WF stores rules in XML<sup>8</sup> format. Implementation of rulespect in .NET environment is shown in Figure 3.

## 7 Implementation

To demonstrate the practical implementation of rulespect, a travel agency case study is developed. In the case study, travel agency business is realized using .NET web service, i.e., travel agency web service. Main operation of travel agency is trip booking. Trip booking is implemented using web method of web service. This implementation demonstrates, how cross-cutting concern of logging is modularized using rulespect, how easy the syntax for implementation is, and how much control it provides over weaving mechanism. Application logs each time trip booking web method is called and every time it is returned, tracing number of trips booked and the time web service took to satisfy the client request. In the example, trip booking web method is the entity of interest. Trip booking web method is tagged as join point. This tagging as join point is implemented using annotation of attribute-based programming feature of .NET which is shown below:

```
[ TraceExtension ( ) ]  
[ WebMethod ]  
public string TripBooking ( )  
{ ... }
```

In the code above, TraceExtension is the tagging that will enable monitoring the TripBooking web method for join point and will do the interception for all incoming requests and outgoing responses. As we have tagged a web method, here interception means the interception of SOAP messages, however, for traditional plain classes it means interception of method call or variable call, etc. Every time the client request for trip booking, request is intercepted by tagging implementation. Interception of request is a before advice. After complete execution of trip booking, response is sent back to client. Tagging implementation also enable interception of response. Interception of response is an after advice. Similarly, interception of both, request and response leads to an around advice. In this implementation both before and after advice events are captured to trigger the workflows. As our model specifies the rule as event-condition-action, for every rule, join point for TripBooking web method will serve as an event for the rules. Now as event is triggered, workflow is executed, and so are the rules. However, use of workflow is not mandatory. User can directly execute rules on join point event. This implementation makes use of the workflow for possible exploitation of workflow-based rule execution, which is not covered in the presented implementation. There is a rule defined for logging with condition which checks if logging is enabled for the service or not.

If logging is enabled, then execution is logged otherwise it is not logged. This is the simple rule to demonstrate the basic rulespect implementation. Rulespect enable much better control through provision of multiple rules as shown in Figure 4. For example,

<sup>8</sup> Extensible Markup Language (XML), <http://www.w3.org/XML/>

Name	P.	Ree...	Ac...	Rule Preview
Authentication	2	Always	True	IF !this.authentication THEN LoggingAspect.Logging.Log("Unauthenticated access for " + this.mes:
LoggingRule	1	Always	True	IF this.loggingenabled == True THEN this.log(this.message.MethodInfo.Name + " Called")

Fig. 4. Multiple rules (i.e. RuleSet) example.

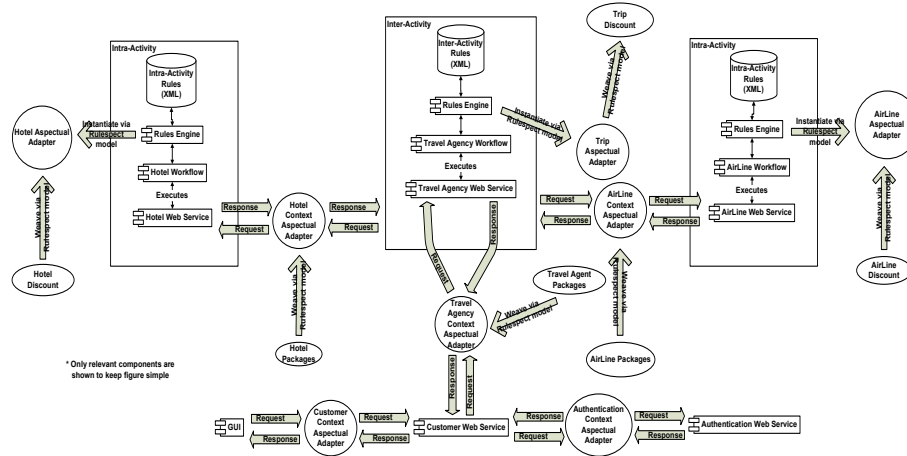


Fig. 5. Rulespect implementation scenario.

a rule for unauthenticated user can also be included. A condition will check if the user is authenticated. If user is not authenticated, still logging can be enforced, even it is disabled in web service. This provision is achieved through possibility of multiple rule definition and prioritization of rules. Rule-based pointcut in rulespect allows picking out join point not only based on signature, rather also based on the behavior of application. To demonstrate the capability of rulespect for large projects, we extended our existing web service based ECA-centric architecture defined in [32] using rulespect as shown in Figure 5. In our old implementation for AOP we used Spring.Aop<sup>9</sup>, a framework for AOP support in .NET. Using rulespect model, we were able to implement the AOP constructs by using simple language constructs like interception of method invocation and rules engine. This enabled us to reduce the technologies involved to implement the architecture, furthermore it enabled the business user to adapt the IS using the familiar rule modification approach. Rulespect also enabled us to change our existing web services to context-sensitive web services. Using rulespect based approach, we were able to separate context related implementation from web service as aspect. We termed these aspects as contextual aspects. Based on defined rules, contextual aspects were weaved into web services implementation to give a web service user more personalized experience. As shown in Figure 5, we have rulespect based context adapter

<sup>9</sup> SPRING .Net Application Framework, <http://www.springframework.net/>

X  
for every web service. These web service adapters manipulate request and response to identify and process context. For example, in the extended architecture, travel agency web service manipulates the customer context to identify the location of customer, then it uses this information to provide customer with most relevant trips and packages. Similarly, airline web service uses travel agent context information to identify the location of travel agent. Based on travel agent location, it will provide it with the most relevant flight information and packages. Furthermore, it will also use the context information to identify the type of travel agent. Based on the type it will identify what special features to offer that might not be available to customer if it approaches airline web service directly. Using rulespect we are able to separate the context manipulation functionality from core functionality of web service.

## 8 Conclusions and Future Work

Rulespect provided easy-to-use rule-based syntax for specifying aspect weaving details. Rulespect model is simple and defined in platform independent manner. This paper demonstrated implementation in .NET, however, this model can be implemented using any technology that supports the implementation of the defined behavior. Since this model use rules, business IS is the most relevant domain for its use. In this paper, we demonstrated how rulespect can be used to implement AOP for web services. In future we want to extend this concept of rulespect for using AOP at web service composition, definition, and discovery level. We also want to analyze the performance of the model for large IS.

## References

1. C. Adams and S. Boeyen. Uddi and wsdl extensions for web service: a security framework. In *XMLSEC '02: Proceedings of the 2002 ACM workshop on XML security*, pages 30–35, New York, NY, USA, 2002. ACM.
2. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, Berlin, Germany, October 2003.
3. R. Altman, A. Cyment, and N. Kicillof. On the need for setpoints. In *EIWAS 2005: European Interactive Workshop on Aspects in Software*, <http://prog.vub.ac.be/events/eiwas2005/>, 2005.
4. S. Apel, D. Batory, and M. Rosenmüller. On the structure of crosscutting concerns: Using aspects or collaborations? In *GPCE Workshop on Aspect-Oriented Product Line Engineering (AOPLE)*, 2006.
5. S. Apel, T. Leich, M. Rosenmüller, and G. Saake. Combining feature-oriented and aspect-oriented programming to support software evolution. In *AMSE05, at ECOOP05*, 2005.
6. S. Apel, T. Leich, and G. Saake. Aspectual feature modules. *IEEE Transactions on Software Engineering*, 34(2):162–180, 2008.
7. C. Bockisch, M. Haupt, M. Mezini, and K. Ostermann. Virtual machine support for dynamic join points. In *AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development*, pages 83–92, New York, NY, USA, 2004. ACM.
8. J. Bonér. What are the key issues for commercial aop use: how does aspectwerkz address them? In *AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development*, pages 5–6, New York, NY, USA, 2004. ACM.

9. A. Charfi and M. Mezini. Hybrid web service composition: business processes meet business rules. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 30–38, New York, NY, USA, 2004. ACM.
10. M. Cibrán and B. Verheecke. Modularizing Web Services Management with AOP. In *First European Workshop on Object Orientation and Web Services (EOOWS)*, Darmstadt, Alemanha, 2003.
11. A. Colyer, A. Rashid, and G. Blair. On the separation of concerns in program families. Technical report, Computing Department, Lancaster University, 2004.
12. R. Douence and M. Sudholt. A model and a tool for Event-based Aspect-Oriented Programming (EAOP). Technical report, Technical report no. 02/11/INFO, Ecole des Mines de Nantes, 2002, 2002.
13. J. Grundy. Multi-perspective specification, design and implementation of software components using aspects. *International Journal of Software Engineering and Knowledge Engineering*, 10(6):713–734, 2000.
14. J. Grundy and G. Ding. Automatic validation of deployed J2EE components using aspects. In *Proceedings of the 17th IEEE International Conference on Automated Software Engineering, San Diego, California, USA*, pages 47–56, 2001.
15. J. Hanson. .NET versus J2EE Web Services: A comparison of approaches. *Web Services Architect*, 9, 2002.
16. F. Irmert, M. Meyerhöfer, and M. Weiten. Towards runtime adaptation in a soa environment. In *RAM-SE*, pages 17–26. Fakultät für Informatik, Universität Magdeburg, 2007.
17. K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute, November 1990.
18. M. Keidl and A. Kemper. Towards context-aware adaptable web services. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 55–65, New York, NY, USA, 2004. ACM.
19. G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of aspectj. In *ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming*, pages 327–353, London, UK, 2001. Springer-Verlag.
20. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proceedings of the 11th European Conference on Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242, Jyväskylä, Finland, June 1997. Springer-Verlag.
21. M. Kuhlemann and C. Kästner. Reducing the Complexity of AspectJ Mechanisms for Recurring Extensions. In *Workshop on Aspect-Oriented Product Line Engineering*, pages 14–19, 2007.
22. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, September 1999.
23. M. Litoiu. Migrating to web services ” latency and scalability. In *WSE '02: Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)*, page 13, Washington, DC, USA, 2002. IEEE Computer Society.
24. J. Martin, A. Arsanjani, P. Tarr, and B. Hailpern. Web services: Promises and compromises. *Queue*, 1(1):48–58, 2003.
25. M. Mezini and K. Ostermann. Variability management with feature-oriented programming and aspects. In *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, pages 127–136, New York, NY, USA, 2004. ACM.
26. L. D. B. Navarro, M. Südholt, W. Vanderperren, B. D. Fraine, and D. Suvée. Explicitly distributed aop using awed. In *AOSD '06: Proceedings of the 5th international conference on Aspect-oriented software development*, pages 51–62, New York, NY, USA, 2006. ACM.

27. G. Ortiz, J. Hernandez, and P. Clemente. Decoupling non-functional properties in Web Services: an aspect-oriented approach. In *Proceedings of The 2nd European Workshop on Web Services and Object Orientation (EOOWS2004) held in conjunction with the 18th European Conference on Object-Oriented Programming (ECOOP2004)*, 2004.
28. G. Ortiz, J. Hernández, and P. Clemente. Web Service orchestration and interaction patterns: an aspect-oriented approach. In *Forum Papers Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC)*, 2004.
29. G. Ortiz, J. Hernández, P. Clemente, and P. Amaya. How to model aspect-oriented web services. In *Proceedings of the 1st Model-driven Web Engineering Workshop, ICWE*, 2005.
30. M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, Nov. 2007.
31. G. Piccinelli, W. Emmerich, C. Zirpins, and K. Schutt. Web service interfaces for inter-organisational business processes an infrastructure for automated reconciliation. In *Enterprise Distributed Object Computing Conference, 2002. EDOC'02. Proceedings. Sixth International*, pages 285–292, 2002.
32. S. S. u. Rahman, N. Aoumeur, and G. Saake. An adaptive eca-centric architecture for agile service-based business processes with compliant aspectual .net environment. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 240–247, New York, NY, USA, 2008. ACM.
33. H. Rajan and K. J. Sullivan. Classpects: unifying aspect- and object-oriented language design. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 59–68, New York, NY, USA, 2005. ACM.
34. S. Singh, J. Grundy, J. Hosking, and J. Sun. An architecture for developing aspect-oriented web services. In *Third IEEE European Conference on Web Services, 2005. ECOWS 2005*, page 11, 2005.
35. O. Spinczyk, D. Lohmann, and M. Urban. Aspectc++: an aop extension for c++. *Software Developers Journal*, pages 68–74, 2005.
36. P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, Jr. N degrees of separation: multi-dimensional separation of concerns. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 107–119, New York, NY, USA, 1999. ACM.
37. A. Tesanovic, K. Sheng, and J. Hansson. Application-tailored database systems: A case of aspects in an embedded database. *ideas*, pages 291–301, 2004.
38. S. Tilley, J. Gerdes, T. Hamilton, S. Huang, H. Muller, and K. Wong. Adoption challenges in migrating to web services. In *Web Site Evolution, 2002. Proceedings. Fourth International Workshop on*, pages 21–29, 2002.
39. B. Verheecke, M. Cibrán, W. Vanderperren, D. Suvee, and V. Jonckers. AOP for dynamic configuration and management of web services. *International Journal of Web Services Research*, 1(3):25–41, 2004.
40. M. Wand, G. Kiczales, and C. Dutchyn. A semantics for advice and dynamic join points in aspect-oriented programming. *ACM Trans. Program. Lang. Syst.*, 26(5):890–910, 2004.