SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Thanks for a great conference.

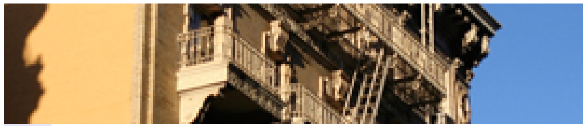
This year's Software Product Line Conference had 170 attendees from all over the world. We published a great record of the conference in two special proceedings volumes.

Volume 1, which contains session research papers, experience reports, workshop descriptions, tutorial descriptions, and doctoral symposium descriptions (ISBN 978-0-9786956-2-0)

Volume 2, which contains workshop papers, doctoral symposium papers, tool demo descriptions

Organizations Need Software Product Lines
Now More Than Ever

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Effectively using software product lines improves time to market, cost, productivity, and quality. They also enable rapid market entry and flexible response. And, using software product lines simplifies software maintenance and enhancement.

Product line approaches apply practices, processes, technology, and tools aimed at strategic, planned reuse – reuse of technical and business artifacts throughout the life cycle. The success of product line approaches depends on overall organizational goals in alignment with the implementation of appropriate product line practices and technologies. This strong interrelationship between organizational concerns and engineering practices makes working and researching the field of software product lines as interesting as it is challenging.

If your organization develops software, you need to explore the benefits software product lines deliver. If you already work with software product lines, you need to ensure that every potential advantage is reached and risk mitigated.

About SPLC 2009

The Software Product Line Conference (SPLC) is the premier forum for product line researchers, practitioners, and educators to present and discuss current and emerging trends. SPLC provides the product line community with opportunities to hear industry leaders' real-world experiences and researchers’ latest ideas, and to learn from both. Now in its 13th year, SPLC 2009 was held held August 24 – 28 in San Francisco, California.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ____ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:
* Paul Jensen, Overwatch, USA

* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

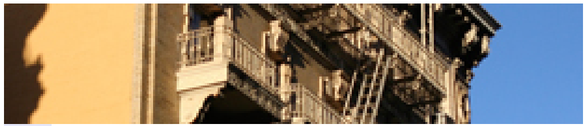
August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Program

Monday, August 24, 2009	
Workshops – 8:30 AM – 5:00 PM	Contact
DOCTORAL SYMPOSIUM »	
W1: *Third International Workshop on Dynamic Software Product Lines (DSPL)* learn more >	Mike Hinchey *Lero, the Irish Software Engineering Research Centre, Limerick, Ireland*

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

W2: *First International Workshop on Model-Driven Approaches in Software Product Line Engineering (MAPLE 2009)* learn more >	Goetz Botterweck *Lero, University of Limerick, Ireland*
W3: *Scalable Modeling Techniques for Software Product Lines (SCALE 2009)* learn more >	Tomoji Kishi *Waseda University, Japan*

AM Tutorials – 8:30 AM – 12:00 PM

T1:	*Introduction to Software Product Lines* Patrick J. Donohoe, Software Engineering Institute, USA
T2:	*Systems and Software Product Line Engineering with the SPL Lifecycle Framework* Charles Krueger, BigLever Software, USA
T3:	*Production Planning in a Software Product Line Organization* Gary Chastek & John D. McGregor, Software Engineering Institute, USA

PM Tutorials – 1:30 PM – 5:00 PM

T5:	*Introduction to Software Product Line Adoption* Linda M. Northrop & Lawrence G. Jones, Software Engineering Institute, USA
T6:	*From Product Line Requirements to Product Line Architecture - Optimizing Industrial Product Lines for New Competitive Advantage* Juha Savolainen, Nokia Research Center, Helsinki, Finland Mike Mannion, Glasgow Caledonian University, Glasgow, Scotland

Tuesday, August 25, 2009

Workshops – 8:30 AM – 5:00 PM	Contact

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: [ ] Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John D. McGregor, Clemson University, USA

W5: *Service-Oriented Architectures and Software Product Lines (SOAPL)—Enhancing Variation* learn more >	Robert W. Krut *Software Engineering Institute, USA*
W6: *Consolidating Community Consensus in Product Line Practice* learn more >	Paul Clements, *Software Engineering Institute, USA*

AM Tutorials – 8:30 AM – 12:00 PM

T4:	*Introducing and Optimizing Software Product Lines Using the FEF* Klaus Schmid, University of Hildesheim, Germany
T8:	*Evolutionary Product Line Requirements Engineering* Isabel John, Fraunhofer IESE, Germany Karina Villela, Fraunhofer IESE, Germany
T9:	*Transforming Legacy Systems into Software Product Lines* Danilo Beuche, pure-systems GmbH, Germany
T10:	*Leveraging Model Driven Engineering in Software Product Line Architectures* Bruce Trask, MDE Systems, Inc Angel Roman, MDE Systems, Inc

PM Tutorials – 1:30 PM – 5:00 PM

T7:	*Inner Source Product Line Development* Frank van der Linden, Philips Medical Systems, The Netherlands
T11:	*Building Reusable Testing Assets for a Software Product Line* John D. McGregor, Software Engineering Institute, USA
T12:	*Pragmatic Strategies for Variability Management in Product Lines in Small- to Medium-Size Companies* Stan Jarzabek, National University of Singapore, Singapore
T13:	*Using Domain-Specific Languages for Product Line Engineering* Markus Voelter, itemis AG, Germany

Industry Track:
* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

Wednesday, August 26, 2009

9:00 AM – 9:30 AM – Introduction

9:30 AM – 10:30 AM – Keynote Address
Richard Gabriel, IBM
Science Is Not Enough: On the Creation of Software

10:30 AM – 11:00 AM – Break

11:00 AM – 12:30 PM – Parallel Session

Research Papers: Configuration

Dealing with Fine-Grained Configurations in Model-Driven SPLs
Hugo Arboleda, Rubby Casallas, Jean-Claude Royer

Automated Reasoning for Multi-Step Software Product Line Configuration Problems
Jules White, David Benavides, Brian Dougherty, Douglas C. Schmidt

Issues in Mapping Change-Based Product Line Architectures to Configuration Management Systems
Nicolás López, Rubby Casallas, André van der Hoek

Experience Report: Industry 1

Supporting Usability in Product Line Architectures
Pia Stoll, Len Bass, Elspeth Golden, Bonnie E. John

Building a Comprehensive Software Product Line Cost Model
Andrew J. Nolan

Building Automotive Product Lines around Managed Interfaces
Walter J. Slegers

12:30 PM – 2:00 PM – Lunch

2:00 PM – 3:30 PM – Parallel Session

Research Papers: Scoping

A Decade of Scoping - A Survey
Isabel John, Michael Eisenbarth

Inferring Information from Feature Diagrams to Product Line Economic Models
David Fernandez-Amoros, Ruben Heradio-Gil, Jose Antonio Cerrada-Somolinos

Default Values for Improved Product Line Management
Juha Savolainen, Jan Bosch, Juha Kuusela, Tomi Männistö

Working Session 1:

Future Directions – The View from the Lab

Nobuaki Kozuka
Yuzo Ishida
NOMURA RESEARCH Institute, Ltd.

Ralf Carbon
Fraunhofer IESE

Linda M. Northrop
Software Engineering Institute

3:30 PM – 4:00 PM – Break

4:00 PM – 5:30 PM – Parallel Session

Research Papers: Variability

Strategies for Variability Transformation at Runtime
Carlos Cetina, Øystein Haugen, Xiaorui Zhang, Franck Fleurey, Vincente Pelechano

Modeling PLA Variation of Privacy-Enhancing Personalized Systems
Yang Wang, Scott A. Hendrickson, André van der Hoek, Richard N. Taylor, Alfred Kobsa

Variability Management in Software Product Lines: A Systematic Review
Lianping Chen, Muhammad Ali Babar, NourAli

Tool Demos 1

A Tool to Support Usability in Product Line Architectures
Elspeth Golden, Bonnie E. John, Len Bass

PLUM (Product Line Unified Modeller)
Jabier Martínez, Cristina López, Aitor Aldazabal, Jason Mansell and Marta del Hierro

The Fraunhofer Decision Modeler
Daniel Pech, Isabel John

7:00 PM – 10:00 PM – Reception

Thursday, August 27, 2009

9:30 AM – 10:30 AM – Keynote Address

Jacob G. Refstrup, HP

Adapting to Change: Architecture, Processes and Tools: A Closer Look at HP's Experience in Evolving the Owen Software Product Line

10:30 AM – 11:00 AM – Break

11:00 AM – 12:30 PM – Parallel Session

Research Papers: Experiences and Evolution

Gathering Current Knowledge About Quality Evaluation in Software Product Lines
Sonia Montagud, Silvia Abrãhao

Running a Software Product Line – Standing Still Is Going Backwards
Hans Peter Jepsen, Danilo Beuche

From Software Product Lines to Software Ecosystems
Jan Bosch

Experience Report: Industry 2

Adopting Software Product Line Principles to Manage Software Variants in a Complex Avionics System
Walter Hipp, Frank Dordowsky

Experiences with Software Product Line Engineering in Product-Development-Oriented Organizations
Yasuaki Takebe

Variability Management in Small Development Organizations
Daniel Pech, Jens Knodel, Ralf Carbon, Clemens Schitter, Dirk Hein

12:30 PM – 2:00 PM – Lunch

2:00 PM – 3:30 PM – Parallel Session

Research Papers: Derivation

Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects
Pádraig O'Leary, Rick Rabiser, Ita Richardson, Steffen Thiel

Context Awareness for Dynamic Service-Oriented Product Lines
Carols Parra, Xavier Blanc, Laurence Duchien

Product-Line-Based Requirements Customization for Web Service Compositions
Hongyu Sun, Robyn R. Lutz, Samik Basu

Working Session 2:

Future Directions –
The View from the Trenches

3:30 PM – 4:00 PM – Break

4:00 PM – 6:00 PM – Parallel Sessions

Research Papers: Industrial Product Lines

Towards a Product Line Approach for Office Devices Facilitating Customization of Office Devices at Ricoh Co. Ltd.
Ralf Carbon, Sebastian Adam, Takayuki Uchida

Verifying Architectural Design Rules of the Flight Software Product Line
Dharmalingam Ganesan, Mikael Lindvall, Chris Ackermann, David McComas, Maureen Bartholomew

An Industrial Case of Exploiting Product Line Architectures in Agile Software Development
Muhammad Ali Babar, Tuomas Ihme, Minna Pikkarainen

Tool Demos 2

Modeling and Building Software Product Lines with pure::variants
Danilo Beuche

IBM Rational: Moving Beyond Application Lifecycle Management to Product Line Lifecycle Management
Marty Bakal, John Carrillo, Ken Jackson

The BigLever Software Gears Software Product Line Lifecycle Framework
Charles W. Krueger

XVCL Reuse Method and XVCL Workbench
Stan Jarzabek

Friday, August 28, 2009

9:30 AM – 10:30 AM – Keynote Address

Kyo Chul Kang, Ph. D., Pohang University of Science and Technology (POSTECH) in Korea

FODA: Twenty Years of Perspective on Feature Models

10:30 AM – 11:00 AM – Break

11:00 AM – 12:30 PM – Parallel Session

Research Papers:
Feature Models

On the Impact of the Optional Feature Problem: Analysis and Case Studies
Christian Kaestner, Sven Apel, Syed Saif ur Rahman, Marko Rosenmueller, Don Batory, Gunter Saake

Supplier Independent Feature Modelling
Herman Hartmann, Tim Trew, Aart Matsinger

Relating Requirements and Feature Configurations: A Systematic Approach
Thein Than Tun, Quentin Boucher, Andreas Classen, Arnaud Hubaux, Patrick Heymans

GoldFish Panel:

How to Maximize Business Return of Software Product Line Development

12:30 PM – 2:00 PM – Lunch

2:00 PM – 3:30 PM – Parallel Session

Research Papers: Feature Modeling

A Framework For Constructing Semantically Composable Feature Models from Natural Language Requirements
Nathan Weston, Ruzanna Chitchyan, Awais Rashid

Formal Modelling and Analysis of Feature Configuration Workflows
Arnaud Hubaux, Andreas Classen, Patrick Heymans

SAT-Based Analysis of Feature Models Is Easy
Marcilio Mendonca, Andrzej Wasowski, Krzysztof Czarnecki

Panel:

Quality Assurance in Software Product Lines

Robyn R. Lutz
Jet Propulsion Lab, NASA
Iowa State University

Len J. Bass
Software Engineering Institute

3:30 PM – 4:00 PM – Break

5:00 PM – 6:00 PM – Ice Cream Social

4:00 PM – 5:30 PM – Hall of Fame

Chair: David Weiss

A hall of fame serves as a way to recognize distinguished members of a community in a field of endeavor. Those elected to membership in a hall of fame represent the highest achievement in their field, serving as models of what can be achieved and how. Each Software Product Line Conference culminates with a session in which members of the audience nominate systems for induction into the Software Product Line Hall of Fame.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

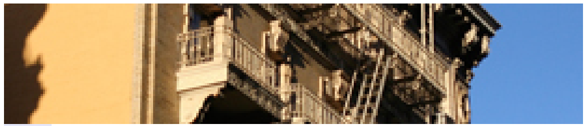
Five Reasons Why You Can’t Afford to Miss SPLC 2009

If you attend only one software development event this year, make sure it’s the 13th International Software Product Line Conference (SPLC 2009), August 24-28 in San Francisco California. From an outstanding technical program and celebrated keynote speakers who are experts in their field to high-quality tutorials, workshops, and working sessions, you will learn best practices for software product line adoption and get a chance to exchange knowledge with software product line practitioners from all over the world.

Here are five reasons why you can’t afford to miss SPLC 2009:

1. **Get a chance to learn from Kyo Kang.**
   Kang—a professor at the Pohang University of Science and Technology and expert in software reuse and product line engineering, requirements engineering, and computer-aided software engineering—will mark the 20th anniversary of Feature-

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature modeling at SPLC»
- Take advantage of expert-led tutorials at

Oriented Domain Analysis (FODA) with his keynote on the technique.

2. **Network in excellent company.**
   Organizations including Ericsson, Hitachi, Honeywell, IBM, Lockheed Martin, Mitsubishi, Nokia, Philips, Raytheon, Rolls Royce, Siemens, and Toshiba have attended previous SPLCs.
3. **Tell your software product line story.**
   Through interactive tutorials, workshops, and working sessions that cover topics across the experience spectrum, SPLC 2009 provides many opportunities to share your experience with others and learn from their successes and failures. And by the time you leave, you'll be equipped with the latest software product line research to apply to your job and share with coworkers.
4. **Come to beautiful San Francisco.**
   San Francisco is a great location that offers some of the world's greatest restaurants, scenery, and cultural attractions. It offers unique landmarks such as the Golden Gate Bridge, cable cars, Alcatraz, and the largest Chinatown in the United States. It is home to stellar theatre, opera, symphony, and ballet companies, and its creative chefs, abundance of fresh ingredients, and diverse cultural influences create unforgettable dining experiences.
5. **Save $200 if you register by August 9.**
   In addition to that early-bird savings, you can save with a special conference hotel rate. And if you're an SEI Member or a student, you can save even more. Register today.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

SPLC August 24-28 »

- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: [ ] Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Call for Participation and Important Submission and Acceptance Information

Workshop Paper Submission Guidelines

Accepted workshop papers will be published electronically in the conference proceedings. Papers must not exceed 10 pages in the IEEE Computer Society Conference Format for 8.5x11-inch Proceedings Manuscripts.

To submit your camera-ready accepted paper, please follow these steps:

- Visit the SPLC 2009 submission site hosted by EasyChair.org at: http://www.easychair.org/conferences/?conf=splc2009
- Create an EasyChair account by completing the request form. A confirmation email

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

SPLC NEWS

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

will be sent to your email address.

- Confirm your email address by clicking on the link in the email and entering your secret word. Create your account.
- Visit http://www.easychair.org/conferences/?conf=splc2009, enter your user name and password, and submit your proposal.

For Reference Only - All Deadlines Passed

View the call for participation in the doctoral symposium »

View the call for industry track reports »

View the call for workshop proposals »

View the call for tutorial proposals »

View the call for tool and demonstration proposals »

We are seeking contributions from diverse perspectives along two dimensions:

1. ranging from practice to research

- Practice perspectives capture the identified needs or the selected solutions relative to a unique organizational context and its associated constraints.
- Research perspectives drive product line technologies forward by improving practices, underlying technologies, or tool support.

2. ranging from retrospectives to visions

- Retrospectives summarize existing work or experiences and derive lessons learned for product line researchers or practitioners.
- Visions motivate and outline work to be done in the field of software product lines ranging from postulating product line engineering methods to pointing out open

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH hypotheses that must be validated by the product line community

Within these two perspectives we expect the following questions to be of particular interest to the product line community:

- How can safety (or any other quality attribute) be managed systematically in a product line context?
- How can product lines be engineered in a complex organizational network of original equipment manufacturers (OEMs) and suppliers including those of commercial off-the-shelf (COTS) or open source components?
- How can a product line approach be centered on a given reference architecture in a certain domain or market segment (e.g., AUTOSAR for the automotive industry)?
- How can agile approaches be combined with product line practices?
- How can service-oriented architecture (SOA) be combined with product line practices?

We invite you to present your perspectives to the product line community and discuss them at SPLC 2009. Please submit your contributions as research or experience papers, tutorials, workshop proposals, demonstrations, or poster presentations. Additionally, we strongly encourage young researchers to participate in the Doctoral Symposium.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

Program Chair: John McGregor, Clemson University, USA

Industry Track:
* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

SPLC 2009 Speakers

Richard P. Gabriel

Science is Not Enough:
On the Creation of Software

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Richard P. Gabriel is a Distinguished Engineer at IBM Research and is looking into the architecture, design, and implementation of extraordinarily large, self-sustaining systems. He is the award-winning author of four books and a poetry chapbook. He lives in California.

Jacob G. Refstrup

Adapting to Change: Architecture, Processes, and Tools: A Closer Look at HP's Experience in Evolving the Owen Software Product Line

Jacob G. Refstrup is a distinguished technologist at Hewlett-Packard's Imaging and Printing Group. He is the lead architect for the Owen software product line architecture, which is used across multiple inkjet product families. He has spent the last 10 years contributing to the Owen architecture, processes, tools, and code base. He has a master's degree in software engineering from Imperial College, London.

Kyo Chul Kang, Ph. D.

modeling at SPLC! REGISTER NOW»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ☐ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

FODA: Twenty Years of Perspective on Feature Models

After receiving his Ph.D. from the University of Michigan in 1982, Dr. Kang worked as a visiting professor at the University of Michigan and as a member of the technical staff at Bell Communications Research and AT&T Bell Laboratories. He joined the Carnegie Mellon Software Engineering Institute as a senior member of the technical staff in 1987. He is currently a professor at the Pohang University of Science and Technology (POSTECH) in Korea. He served as director of the Software Engineering Center at Korea Information Technology Promotion Agency (KIPA) from 2001 to 2003. Also, he served as general chair for the 8th International Conference on Software Reuse (ICSR) held in Madrid, Spain in 2004 and as general chair for the 11th International Product Line Conference (SPLC 2007) held in Kyoto, Japan in 2007.

While at the University of Michigan, he was involved in the development of PSL/PSA, a requirements engineering tool system, and a Meta modeling technique. Since then, his research has focused on software reuse. While on leave to KIPA, he promoted the use of the SEI's Capability Maturity Model (CMM) in Korea. His current research areas include software reuse and product line engineering, requirements engineering, and computer-aided software engineering.

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

Travel

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Travel & Venue

Hotel Accommodations
San Francisco Airport Marriott
1800 Old Bayshore Highway
Burlingame, California 94010 USA
T: 1-650-692-9100
F: 1-650-692-8016

The San Francisco Airport Marriott is located just minutes from the San Francisco International Airport and 15 miles from downtown San Francisco.

Reservations

$159 per night plus sales and occupancy taxes

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Register online or call the hotel at 1-650-692-9100 and mention SPLC 2009.

Travel

If you plan to arrive by airplane, plan to take advantage of the hotel's free shuttle from the San Francisco International Airport. To get to the shuttle, go to the airport's second floor—the departure level—and follow the shuttle signs. (Please note that the Marriott and Hyatt share a shuttle, so both names will appear on the vehicle.) And once you're at the hotel, getting to downtown San Francisco is a short and easy trip via the Bay Area Rapid Transit (BART) system.

Registration Desk Hours

All the technical sessions will take place at the conference hotel—the San Francisco Airport Marriott. The registration desk will be staffed during the following hours:

Monday, August 24, 2009	7:30 AM – 5:00 PM
Tuesday, August 25, 2009	7:30 AM – 5:00 PM
Wednesday, August 26, 2009	8:00 AM – 3:00 PM
Thursday, August 27, 2009	8:00 AM – 3:30 PM
Friday, August 28, 2009	8:00 AM – 1:30 PM

Receptions

Plan to socialize with your fellow attendees at a special reception planned at the hotel from 7-10PM on Wednesday, August 26. Please join us for this kickoff event to enjoy some local San Francisco fare and prepare for a great conference experience. We are also planning an ice cream social from 5-6PM on Friday, August 28, to coincide with the conference's final event, the Product Line Hall of Fame.

Be sure to review the final program details and come ready to network with your peers and learn about the latest software product line research and practices.

About San Francisco

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: Join Now

SPLC 2009 ON Linkedin

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

San Francisco is often called "Everybody’s Favorite City," a title earned by its scenic beauty, cultural attractions, diverse communities, and world-class cuisine. Measuring 49 square miles, this very walkable city is dotted with landmarks like the Golden Gate Bridge, cable cars, Alcatraz, and the largest Chinatown in the United States. A stroll of the City’s streets can lead to Union Square, the Italian-flavored North Beach, Fisherman’s Wharf, the Castro, Japantown, and the Mission District, with intriguing neighborhoods to explore at every turn.

The City is home to world-class theatre, opera, symphony and ballet companies and often boasts premieres of Broadway-bound plays and culture-changing performing arts. San Francisco is one of America’s greatest dining cities. The diverse cultural influences, proximity of the freshest ingredients and competitive creativity of the chefs result in unforgettable dining experiences throughout the city.

For more information about events, activities, and transportation in San Francisco, visit www.onlyinsanfrancisco.com.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

SPLC 2009 Sponsors

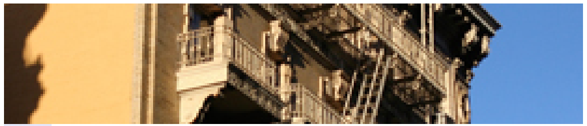
Platinum Sponsors

Software Engineering Institute

Software Engineering Institute
Carnegie Mellon

Since 1984, the Carnegie Mellon® Software Engineering Institute (SEI) has served the nation as a federally funded research and development center. The SEI staff has advanced software engineering principles and practices and has served as a national resource in software engineering, computer security, and process improvement. As part of Carnegie Mellon University, which is well known for its highly rated programs in computer science and engineering, the SEI operates at the leading edge of technical innovation.

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

visit the SEI website »

Hitachi

Hitachi, Ltd., headquartered in Tokyo, Japan, is a leading global electronics company with approximately 390,000 employees worldwide. The company offers a wide range of systems, products, and services in market sectors including information systems, electronic devices, power and industrial systems, consumer products, materials, logistics, and financial services.

visit hitachi's website »

Lero

Lero is the Irish Software Engineering Research Centre. It is a collaborative organisation, embracing the software engineering research activities in the University of Limerick (UL – lead partner), Dublin City University (DCU), Trinity College Dublin (TCD) and University College Dublin (UCD). Lero focuses on specific domains, especially those where reliability is crucial, including automotive, medical devices, telecommunications and financial services. We develop models, methods and tools that make it cheaper, faster or easier to produce the crucial software. Lero's researchers carry out world class research informed by the requirements of its chosen industrial domains. Our researchers concentrate on problem areas that have potential real-world application and much of the research is carried out in collaboration with industry partners.

visit Lero's website »

modeling at SPLC! REGISTER NOW»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

Gold Sponsors

BigLever Software, Inc.™

BigLever Software, Inc.™ is the industry's leading provider of software product line engineering framework, tools and services. BigLever's patent-pending Gears solution—recipient of the 18th Annual Jolt Productivity Award —dramatically simplifies the creation, evolution and maintenance of embedded or standalone software for a product line portfolio. The Gears tool and software product line (SPL) lifecycle framework shifts the development focus from a multitude of products to a single software production line capable of automatically producing all of the products in a portfolio.

With Gears, software development organizations can reduce development costs and bring new product line features and products to market faster, enabling the business to more reliably target and hit strategic market windows.

visit BigLever’s website »

Silver Sponsors

visit IBM’s website »

Fraunhofer

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

Fraunhofer Center for Experimental Software Engineering Maryland (FC-MD) advances real-world software practices through innovative research into software-engineering technologies and processes. The Center's expertise includes topics such as software architecture, verification & validation, process improvement and measurement.

visit Fraunhofer’s website »

pure-systems GmbH

pure-systems GmbH provides pragmatic, vendor-independent and open-standards-based tools and solutions to help engineers and organisations efficiently manage variants and automate product lines. The company, based in Magdeburg, Germany, was founded in 2001. pure-systems solutions have been deployed with customers in a variety of industries including automotive electronics, medical devices, industry automation, aerospace and transportation, consumer electronics and banking. pure::variants allows organisations to manage complex and variant-rich software systems and products, enabling effective variant management for core assets at all stages of the development process at an affordable cost. For product line development using Model-Driven-Architecture (MDA) and Development (MDD) pure::variants supports UML and SysML at model-element level and provides integrations with IBM Rational Rhapsody, Sparx Enterprise Architect and MATLAB Simulink®. Requirements Management support in pure::variants allows for requirements derivation and extraction of variant-specific information from IBM Rational DOORS, Borland Caliber, MKS Requirements and XML-based documents. Change, Test and Defect Management allowing for impact analysis and decision support is supported through integrations with popular tools like Bugzilla, JIRA and IBM Rational ClearQuest. pure-systems customers achieve significant savings in overall product development cost and shorten their time to market for new products.

visit pure-systems’ website »

Sponsor Opportunities

For more information or to reserve your contribution opportunity now, please contact Dr. Dirk Muthig at +49 (631) 6800 1302 or dirk.muthig@lhsystems.com.

	Platinum US$5,000	Gold US$3,500	Silver US$2,000
Visibility in Conference Bag	two letter-sized pages *(you provide)*	one letter-sized page *(you provide)*	-
Visibility in Conference Program	one-page advertisement *(you provide)* + contributor's name + logo	contributor's name + logo	contributor's name
Visibility in Conference Proceedings	contributor's name + logo	contributor's name	contributor's name
Visibility in Call for Participation	contributor's logo	-	-
Visibility on SPLC 2009	contributor's name + logo	contributor's name + logo	logo
Visibility in Email Distributions	contributor's name	contributor's name	contributor's name
Opening and Closing Session	contributor's name mentioned	-	-

Banners	one banner displayed at conference site *(you provide)*	-	-
Posters	one poster in main conference room + four posters in other conference rooms *(you provide)*	one ANSI D poster in main conference room *(you provide)*	visibility on one poster with all Silver Level contributors in main conference room
Free Attendance	three free conference registrations *(does not include tutorials and workshops)*	two free conference registrations *(does not include tutorials and workshops)*	one free conference registration *(does not include tutorials and workshops)*

NOTE: The contributor must provide all authorized materials by the stipulated deadline and adhere to any conference format requirements.

*****Amounts include value-added tax (VAT).

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Register Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

SPLC 2009 Online Registration

→ REGISTER NOW »

- Update a previous registration »
- Print out the PDF registration form for mailing or faxing »

Registration Dates for SPLC 2009

Early-bird registration for the conference runs until **August 9, 2009.** To qualify for early

SAVE $200 ON YOUR CONFERENCE FEE WHEN YOU REGISTER BY AUGUST 9 »

Registration is now open for the 13th International Software Product Line Conference to be held in San Francisco, CA on August 24-28, 2009. Register by August 9 to save $200 on the conference fee.

This year’s program includes keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the bird savings, your payment must be received by that date along with your registration form. Pre-registration closes at 5 p.m. EST on August 19, 2009. After that, all SPLC attendees may register onsite.

SPLC 2009 Conference Registration Fees

All fees are in U.S. dollars.	Through August 9, 2009			After August 9, 2009		
	Regular	SEI Member	Student	Regular	SEI Member	Student
SPLC Conference only	$700	$595	$400	$900	$765	$600
Conference plus one tutorial OR one workshop	$800	$695	$500	$1000	$865	$700
Each tutorial	$200	$200	$100	$300	$300	$200
Each workshop	$200	$200	$100	$300	$300	$200
Doctoral Symposium	$200	$200	$100	$300	$300	$200

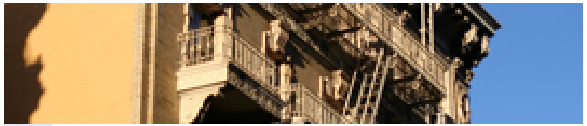
Electronically

To register electronically, simply complete the online registration form. The online registration site works with Internet Explorer and Netscape V6 or above. If you encounter problems registering online, please contact Mandy Mann at Registration Systems Lab by phone +1 (407) 971-4451 or email.

Via Mail or Fax

entire program here »

Register NOW »



SPLC NEWS

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature modeling at SPLC! REGISTER NOW»
- Don't miss out on the early-bird conference rate! REGISTER FOR SPLC NOW»
- SPLC conference rate on San Francisco Airport Marriott held over to August 7 »
- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- SAVE $200 when you register by August 9! »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

1. Simply download or print the Adobe Acrobat PDF file containing the registration form here »
2. Submit the completed form after filling out the form completely and specifying the payment method and any applicable credit card information.
3. Mail or fax the form and any payment papers (such as a check or purchase order).

   Mail them to:
   SPLC 2009
   c/o Registration Systems Lab
   779 East Chapman Road
   Oviedo, FL 32765 USA
   Fax them to **+1 (407) 366-4138**

PLEASE NOTE: By registering, you grant Carnegie Mellon University and/or anyone acting on their behalf ("Carnegie Mellon") permission to photograph, film, or otherwise record and use your name, likeness, image, voice and comments and to publish, reproduce, exhibit, distribute, broadcast, edit and/or digitize the resulting images and materials in publications, advertising materials, or in any other form, and for any purpose without compensation.

Registration Terms and Conditions

Payment Methods Note: We must receive your full payment prior to the conference, or you will be expected to pay onsite in order to attend the conference. Acceptable methods of payment include the following:

- **Credit cards -** We accept MasterCard, Visa, and American Express.
- **Company or personal checks -** Your check must be mailed with your registration form. Keep in mind that to qualify for early-bird savings, we must receive your check by August 9, 2009. Please make your check payable to SEI/CMU and be sure to write the name of your organization on it.
- **Completed purchase orders -** Purchase orders are accepted only until August 19, 2009 and should be signed by the designated fiscal officer in your organization.

Cancellation Requests - Refund requests received in writing and postmarked by August 9, 2009 will be processed minus a $50 administrative fee. NO REFUNDS WILL BE GIVEN AFTER AUGUST 9, 2009. If you do not cancel and do not attend, you will be charged the

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: [   ] Join Now

SPLC 2009 ON Linked in

SAVE WITH SEI MEMBERSHIP

SAVE 15%

Become an SEI Member and Save on Registration for SPLC 2009

You can save 15% on the already reduced early-bird registration price for SPLC 2009 by becoming a Member of the Software Engineering Institute.

SEI Members save on registrations for SEI-sponsored conferences and courses and enjoy opportunities to connect with others in their fields.

Not a member? **Join now »**

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:
* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan

full registration fee. Substitute attendees are welcome at no extra charge; however, we request written notification prior to the conference for preparation of registration materials. For refunds, please allow two to four weeks for processing after the conference.

If you have any problems with this registration system, please send us an email.

We look forward to seeing you at SPLC 2009.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009 Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Tutorials – Monday, August 24, 2009

Morning Session

T1

Introduction to Software Product Lines
Patrick Donohoe, Software Engineering Institute, USA

Abstract:
Software product lines have emerged as a new software development paradigm of great importance. A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Organizations developing a portfolio of products as a software product line are

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

experiencing order-of-magnitude improvements in cost, time to market, staff productivity, and quality of the deployed products.

This tutorial introduces the essential activities and underlying practice areas of software product line development. It is aimed at those in an organization who are in a position to influence the decision to adopt a product line approach and those in a position to carry out that decision. Anyone who can act as a technology change agent will benefit from this tutorial. The tutorial reviews the basic concepts of software product lines, discusses the costs and benefits of product line adoption, introduces the SEI *Framework for Software Product Line Practice*, and describes approaches to applying the practices of the framework.

Presenter Biography:
Patrick Donohoe is a senior member of the technical staff at the Software Engineering Institute, working in the Research, Technology, and System Solutions Program. His current interests are analysis modeling and production planning for software product lines. He has participated in several SEI Product Line Technical Probes and architecture evaluations and is also an instructor in the SEI's Software Product Line Curriculum.

T2

Systems and Software Product Line Engineering
with the SPL Lifecycle Framework
Charles Krueger, BigLever Software, USA

Mainstream forces are driving software product line (SPL) approaches to take a more holistic perspective that is deeply integrated into the systems and software engineering lifecycle. These forces illustrate that SPL challenges will not be solved at any one stage in the product engineering lifecycle, nor will they be solved in independent and disparate silos in each of the different stages of the lifecycle. We describe a response to these forces—the SPL Lifecycle Framework. The motivation for this technology framework is to ease the integration of tools, assets, and processes across the full systems and software development lifecycle.

In this tutorial, we explore how the SPL Lifecycle Framework provides all product line engineers—including systems analysts, requirements engineers, architects, modelers, developers, build engineers, document writers, configuration managers, test engineers,

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: □ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

project managers, product marketers, and so forth—with a common set of SPL concepts and constructs for all of their tools and assets, at every stage of the lifecycle and how to assure that product line development traceability and processes flow cleanly from one stage of the lifecycle to another.

The SPL Lifecycle Framework has been adopted by IBM Rational Software as the standard SPL solution for the Rational toolset, so we will illustrate how the framework is used in conjunction with these and other widely used industry tools that many organizations already have in-house, as well as how to integrate homegrown tools into the framework. We will describe observations and firsthand experiences on how the SPL Lifecycle Framework has enabled mainstream organizations, such as Lockheed Martin, with some of the largest, most sophisticated and complex, safety-critical systems ever built, to transition legacy and new systems and software assets to the SPL approach.

The target audience for this tutorial is (1) practitioners from industry settings who are interested in the most efficient, effective, and proven methods for transitioning to and sustaining software product line practice, and (2) members of the research community who are interested in the new methods emerging from proven industry successes. The tutorial is suitable for all levels, ranging from SPL novices who want to learn how the SPL Lifecycle Framework approach is an improvement over early generation SPL approaches, to experienced SPL practitioners who want to learn about the latest advances in commercial SPL practices.

Presenter Biography:
Charles Krueger, PhD, is the founder and CEO of BigLever Software, the leading provider of SPL framework, tools, and services. He moderates the SoftwareProductLines.com website and is a thought leader in the SPL field, with 20 years of experience and over 40 articles, columns, book chapters, and conference sessions to his credit. He has proven expertise in leading commercial software product line development teams and helping companies establish some of the industry's most highly acclaimed SPL practices. These companies include nominees and inductees in the SPLC Hall of Fame including Salion, LSI Logic, and HomeAway. He received his PhD in computer science from Carnegie Mellon University.

T3

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

Production Planning in a Software Product Line Organization

Gary Chastek & John D. McGregor, Software Engineering Institute, USA

Most software product line organizations recognize the need for two roles: core asset developers and product builders. These roles may both be assumed by the same individual, or each may be assumed by persons who are in different administrative units, in different geographic locations, or of vastly different skill levels. For example, a corporation may have one lab assigned to produce core assets and other labs around the world to use those assets to produce products. The greater the separation among these people, the greater the need for communication and coordination regarding product production.

Production planning is used in many industries to coordinate the efforts of external suppliers who supply parts and to structure the assembly line where products are produced. The need for coordination in a software product line organization is even greater than in hard-goods manufacturing, because product production is less constrained by physical properties or industrial standards. Our research has shown that organizations that fail to plan production are more likely to fail than those that do plan. The goal of this tutorial is to provide participants with techniques for conducting production planning.

We will cover the complete product line life cycle from adoption until a first generation of products is developed. We use a business strategy development tool, Porter’s Five Forces model, to guide strategy development. We will use the Software Process Engineering Meta-model and an instantiation of it—the Eclipse Process Framework—for method development and documentation. For the production plan, we will use a document template that has been used with numerous clients.

Presenter Biographies:

Gary J. Chastek is a senior member of the technical staff at the Software Engineering Institute, working in the Research, Technology, and System Solutions Program. He has presented tutorials and led workshops at SPLC and OOPSLA. Chastek’s current research interests include production planning, variability management, and the use of aspect-oriented development in a software product line.

John D. McGregor is an associate professor of computer science at Clemson University, a founding partner of Luminary Software, and a Visiting Scientist at the Software Engineering Institute. He is co-author of two books on software engineering, including *A Practical Guide to Testing Object-Oriented Software Engineering*. McGregor teaches graduate software engineering courses and courses in the SEI's Software Product Line Curriculum and has presented numerous tutorials at a variety of conferences. He also consults with numerous software development organizations.

Afternoon Session

T5

Introduction to Software Product Line Adoption
Linda Northrop & Larry Jones, Software Engineering Institute, USA

Through a software product line approach, organizations have achieved significant reductions in cost and time to market and, at the same time, increased the quality of families of their software systems. However, to date, there are considerable barriers to organizational adoption of product line practices. Phased adoption is attractive as a risk reduction and fiscally viable proposition. This tutorial describes a phased, pattern-based approach to software product line adoption. This tutorial will acquaint participants with product line adoption barriers and two ways to overcome them:

1. a phased, pattern-based adoption approach

2. explicit linkage with other improvement efforts

The objectives of the tutorial are to acquaint participants with

- issues surrounding software product line adoption
- a phased, pattern-based adoption approach
- adoption planning artifacts
- explicit linkage of software product line adoption with other improvement efforts

The tutorial begins with a discussion of software product line adoption issues, including benefits, barriers, risks, and the technical and organizational factors that influence adoption. We then present the Adoption Factory pattern, a roadmap for phased product line adoption. The tutorial covers the Adoption Factory pattern in detail. Examples of product line adoption plans following the pattern are used to illustrate its utility. The tutorial also describes strategies for creating synergy within an organization between product line adoption and ongoing CMMI or other improvement initiatives.

Participants should have experience in designing and developing software-intensive systems, have some familiarity with modern software engineering concepts and management practices, and be familiar with product line concepts. The tutorial is aimed at those in an organization who are in a position to influence the decision to adopt a product line approach and those in a position to carry out that decision. This includes technical managers at all levels, as well as those on the software development staff. Anyone who can act as a technology change agent will benefit from this tutorial.

Presenter Biographies:

Linda Northrop is director of the Research, Technology, and System Solutions Program at the Software Engineering Institute where she leads the work in architecture-centric engineering, software product lines, systems of systems, and ultra-large-scale (ULS) systems. She is coauthor of the book *Software Product Lines: Practices and Patterns* and led the research group on ULS systems that resulted in the book, *Ultra-Large-Scale Systems: The Software Challenge of the Future.* Before joining the SEI, she was associated with both the United States Air Force Academy and the State University of New York as professor of computer science, and with both Eastman Kodak and IBM as a software engineer.

Lawrence G. Jones is a senior member of the technical staff at the Software Engineering Institute, working in the Research, Technology and System Solutions Program. He has over 39 years experience in software development, management and education including service in the U.S. Air Force. He is the former Chair of the Computer Science Department at the Air Force Academy, current Chair of the ABET Accreditation Council, Past Chair of the ABET Computing Accreditation Commission, a Senior Member of the IEEE and the ACM, and Secretary/Treasurer of the Computing Sciences Accreditation Board.

T6

From Product Line Requirements to Product Line Architecture – Optimizing Industrial Product Lines for New Competitive Advantage
Juha Savolainen, Nokia Research Center, Helsinki, Finland
Michael Mannion, Glasgow Caledonian University, Glasgow, Scotland

Product lines have been used in Nokia for more than a decade. In the consumer products, a commercial challenge is to offer personalization of products and services for individual customers at a mass-production price. Product line development is a compromise between customer requirements, existing product line architectural constraints, and commercial needs. Managing variability is the key to a successful product line development. As a product line evolves, selections of requirements for new products are often constrained by the design of the existing product line architecture and the cost of making these changes.

In this tutorial, we describe how to evaluate the current state of the product line, alleviate identified problems, and select the right techniques for managing the evolution. We discuss techniques, experiences, and open issues about managing the transitions back and forth between product line requirements and architectural components as products evolve. We present a set of rules to map variability in requirements to the architecture. We describe architectural views to design, document, and analyze variability and dependencies. We examine the challenges of these techniques and present results of using them for real-world applications.

Attendees should have a reasonable understanding of product line engineering and the problems of developing medium to large computer-based systems. The audience does not need to know about the mobile phone domain used for the case study. Sufficient explanation will be provided to enable understanding of our key development ideas.

Presenter Biographies:
Juha Savolainen is a principal member of the research staff at Nokia Research Center, Helsinki, Finland. He has extensive experience in working closely with the developers of numerous product lines, helping them to manage and realize variability. He is a frequent speaker, teaching courses on requirements engineering and software architecture. His main research interests include requirements engineering, software architectures, and product line development. He has published more than 25 papers.

Michael Mannion is professor of computing and pro vice-chancellor (international) at Glasgow Caledonian University, Glasgow, Scotland, UK. He has several years of software engineering industrial experience and is a former chairman of the British Computer Society Special Interest Group in Software Reuse. His research interests include product line engineering, software engineering, and engineering education. He has published more than 50 papers.

Tutorials – Tuesday, August 25, 2009

Morning Session

T4

Introducing and Optimizing Software Product Lines Using the FEF
Klaus Schmid, University of Hildesheim, Germany

This tutorial addresses in particular the needs of people who either want to introduce product line engineering in their organizations or want to reevaluate and assess their product line maturity.

The tutorial first provides a general introduction to the principles and success factors of software product line engineering (PLE). It introduces PLE as a modern form of software reuse and describes the history and success factors that make it different from earlier work in the area of software reuse.

According to our point of view, the basic principles of PLE are

- *business orientation:* Product line engineering must be embedded in a business and strategic context.
- *variability management:* Enabling the management of variation is key to efficient and effective reuse.
- *architecture-driven development: E* ffective reuse of implementations requires architectural measures.
- *two-lifecycle approach:* An explicit distinction between development for reuse and with reuse on the process and organization level is very important for the integrity of the product line.

Each of these principles is described, and its relation and importance to product line engineering is discussed.

On this basis, the Families Evaluation Framework (FEF) is described. This is a framework for assessing the status of a software product line organization along four dimensions: business, architecture, process, and organization. While the process dimension relies on an extension of the CMMI, the other three dimensions have been developed from scratch.

The FEF is also illustrated using examples of very large and very small organizations.

Presenter Biography:
Klaus Schmid, PhD, leads the software engineering group at the University of Hildesheim. He has worked in various product line projects; both in research projects like ESAPS, Café, and Families and in industrial projects where he helped to introduce product line engineering or to optimize specific practices. He has authored numerous papers in various areas of product line engineering, as diverse as product line economics and scoping, variability management, product line evolution, and variant generation.

T8

Evolutionary Product Line Requirements Engineering
Isabel John, Fraunhofer IESE, Germany
Karina Villela, Fraunhofer IESE, Germany

Product line engineering has a widespread use in industry now. Therefore, there is a great need for customizable, adaptable, as well as mature methods. Scoping and product line analysis are a unique and integral part of product line engineering. In these phases, we determine where to reuse and what to reuse, establishing the basis for all technical, managerial, and investment decisions in the product line to come. Furthermore, these early phases are highly context dependent. In this tutorial, we will give an introduction on how to analyze an environment with the purpose of planning a product line and its future evolution. We focus on product line requirements engineering methods, comprising product line scoping, product line analysis, and planning for evolution.

With these topics, we completely cover the early phases of product line engineering, enabling practitioners to start with product lines on a solid basis. The intended audience is practitioners who want to learn how to carry out these early phases successfully, as well as researchers who want to know about an integrated approach for product line analysis and planning for future evolution.

1. Introduction
   - overview on product line scoping, analysis, and modeling in an architecture-centric product line approach
   - importance of a thoroughly planned product line as a key factor for successful product line engineering and evolution
   - key principles of product line requirements engineering

2. Scoping
   - introduction to the PuLSE-Eco approach for scoping, which includes an overview on the activities and key principles of scoping
   - explanation on how to build up key artifacts (e.g., a product–feature matrix)

3. Product Line Analysis
   - introduction to the PuLSE-CDA approach for product line analysis, which adds product line specifics to existing requirements engineering approaches and notations
   - overview of other approaches for product line analysis, such as FAST and FODA

4. Product Line Evolution
   - overview of a model of software evolution that defines key concepts for systematic reasoning on product line requirements volatility
   - introduction to PLEvo-Scoping, a method based on such concepts, encompassing its activities and a complete example of its usage
   - integration with existing scoping approaches

This tutorial is based on our experience with product line engineering in many industrial projects. It crystallizes the essence of the experience gained in those industrial projects and combines it with our latest research in the area of evolution in product line requirements.

Presenter Biographies:
Isabel John is a researcher and project leader at Fraunhofer IESE. She works in several research and industrial projects in the context of software product lines, scoping, and requirements engineering. Her work focuses on product line analysis and scoping. She has given several presentations and tutorials on product line engineering at software engineering conferences and in industrial contexts. She received her Diploma degree in Computer Science from the Technical University of Kaiserslautern.

Karina Villela has recently become a researcher at Fraunhofer IESE. As part of her

Alexander von Humboldt Fellowship at this institute, she defined a method for proactively managing the evolving scope of a product line, which was applied in different application domains. Since then, she has been working on product line requirements engineering with the goal of improving the ability of product lines to evolve over time. She received her M. Sc. and PhD degrees in computer science from the Federal University of Rio de Janeiro, in Brazil.

T9

Transforming Legacy Systems into Software Product Lines
Danilo Beuche, pure-systems GmbH, Germany

Not every software product line starts from scratch. Often, an organization faces the problem that after a while its software system is deployed in several variants and the need arises to migrate to systematic variability and variant management using a software product line approach.

The tutorial will discuss issues coming up during this migration process mainly on the technical level but will also discuss some of the organisational questions. The goal of the tutorial is to give attendees an initial idea how a transition into a software product line development process could be done with respect to the technical transition.

The tutorial starts with a brief introduction to software product line concepts, discussing terms such as *problem* and *solution space*, *feature models,* and *versions vs. variants.*

Further tutorial topics will be how to choose adequate problem space modelling and the mining of problem space variability from existing artefacts such as requirements documents and software architecture. Also, part of the discussion will be on the need for separation of problem space from solution space and ways to realize it. A substantial part will be dedicated to variability detection and refactoring in the solution space of legacy systems.

Presenter Biography:
Danilo Beuche, PhD, is CEO of the pure-systems GmbH. Pure-systems is a software company specialized in services and tool development for the application of product line technologies in embedded software systems. In 1995, he started to work in the field of embedded operating systems and software families and received his PhD in this area. His

work on tool support for feature-based software development finally led to the founding of pure-systems in 2001. At pure-systems, he also works as consultant in the area of product line development, mainly for clients from the automotive industry. He has been tutorial presenter, speaker, workshop organizer, and panelist at conferences such as AOSD, ISORC, SPLC, and OOPSLA. In addition, he is the author of articles in scientific journals and software developer magazines.

T10

Leveraging Model Driven Engineering in Software Product Line Architectures
Bruce Trask, MDE Systems, Inc
Angel Roman, MDE Systems, Inc

Model driven engineering (MDE) is a recent innovation in the software industry that has proven to work synergistically with software product line architectures (SPLAs). MDE can provide the tools necessary to fully harness the power of software product lines. The major players in the software industry—including commercial companies such as IBM and Microsoft, standards bodies such as the Object Management Group, and leading Universities such as the ISIS group at Vanderbilt University—are embracing this MDE/PLA combination fully. IBM is spearheading the Eclipse Foundation, including its MDE tools like EMF, GEF, and GMF. Microsoft has launched its Software Factories foray into the MDE space with its Domain Specific Language Toolkit. Top software groups such as the ISIS group at Vanderbilt are using these MDE techniques in combination with SPLAs for very complex systems. The Object Management Group is working on standardizing the various facets of MDE. All of these groups are capitalizing on the perfect storm of critical innovations today that allows such an approach to finally be viable. Further emphasizing the timeliness of this technology is the complexity ceiling the software industry finds itself facing, wherein the platform technologies have increased far in advance of the language tools necessary to deal with them.

The process of d SPLAs can be a complex task. However, the use of MDE techniques can facilitate their development by introducing domain-specific languages, domain-specific graphical editors, and domain-specific generators. Together, these are considered the sacred triad of MDE. Key to understanding MDE and how it fits into SPLAs is to know exactly what each part of the trinity means, how it relates to the other parts, and what the various implementations are for each. This tutorial will walk through the development of an entire

MDE tool as applied to a particular software product line of mobile applications (e.g., for the new Google Android phone platform).

The goal of this tutorial is to educate attendees on what MDE technologies are, exactly how they relate synergistically to SPLAs, and how to actually apply them using an existing Eclipse implementation.

The benefits of the technology are so far reaching that the intended audience spans technical managers, developers, and CTOs. In general, the target audience includes researchers and practitioners who are working on problems related to the design and implementation of SPLAs and would like to understand the benefits of applying MDE techniques towards SPLAs.

Presenter Biographies:
Bruce Trask has been working on complex distributed real-time embedded systems for over 20 years, specializing in software product lines and MDE as applied to these systems in the last 7 years. He has been teaching C++, object orientation, design patterns, UML, CORBA, and framework courses for over 10 years. He has led multiple study groups in the New York/New Jersey/Connecticut area on various topics ranging from design patterns to middleware. He is a regular speaker/presenter at software industry conferences and has delivered tutorials at the OMG. Bruce Trask is the CEO of MDE Systems.

Angel Roman is the chief software architect of MDE Systems and an expert on the Eclipse development environment and its application frameworks. He has presented at various industry conferences on topics such as software defined radios and MDE technologies.

Afternoon Session

T7

Inner Source Product Line Development
Frank van der Linden, Philips Medical Systems, The Netherlands

Open source has shown to be an effective way to do distributed development. This tutorial shows how to profit from the open source model in product line development.

Inner source is a way to exploit the advantages of distributed development in the open source way but in a wish to avoid problems with planning, ownership, and control. Several companies have adopted an inner source development model. In inner source development, a set of teams collaborates in a cooperative eco-system. Similar to open source development, inner source development applies an open, concurrent, model of collaboration. It implies distributed ownership and control of code, early and frequent releasing, and many continuous feedback channels. It makes use of organization mechanisms already in place; for example, for escalation of conflicts or setting up roadmaps. Inner source enables flexibility in (starting, stopping, and changing of) collaborations and in timing and setting priorities of development teams across organizational (and geographical) boundaries.

The tutorial is mainly structured along the basic dimensions of BAPO and the FEF: business, architecture, process, and organization.

The tutorial starts with an overview of BAPO and the FEF. This is a rehearsal of the basic ideas presented in the tutorial T4: “Introducing and Optimizing Software Product Lines Using the FEF.” Next, the idea of inner source is presented, followed with a detailed explanation of inner source in each of the four BAPO dimensions.

Presenter Biography:
Frank van der Linden, PhD, works at Philips Healthcare CTO Office. He received his Ph. D. in pure Mathematics in 1984 at the University of Amsterdam. He was then employed by Philips Research and since 1999 by Philips Medical Systems. During this time, his main interest was in software product lines. He was the project leader of four relevant ITEA projects: ESAPS, CAFÉ, FAMILIES, and COSI. During these projects, he organized as program chair a series of SWAPF & PFE workshops. He then served as general chair of the SPLC 2005 and as program co-chair of the SPLC 2006.

T11

Building Reusable Testing Assets for a Software Product Line
John D. McGregor, Software Engineering Institute, USA

Testing consumes a significant percentage of the resources required to produce software-intensive products. The exact impact on the project is often hard to evaluate, because testing activities are distributed over the entire scope of the development effort. In this tutorial, we take a comprehensive end-to-end view of the testing activities and roles that should be present in a software product line organization.

The Software Engineering Institute (SEI) identifies three areas of responsibility in a product line organization. We relate to testing those described below.

Organizational managers have responsibility for establishing the test strategy for the organization in general and the product line in particular. These activities are directly related to the business goals and scope of the product line.

Technical managers have responsibility for planning the numerous test activities needed to implement the test strategy. These activities are planned in concert with the development activities to coordinate milestones and resources.

Software engineers have responsibility for implementing the planned activities. They select the specific test cases necessary to achieve specific test coverage levels and implement any software needed to apply the test cases to the software under test.

The close relationship between developing software and testing it results in the test activities being crafted with knowledge of the chosen development process. The method engineer arranges the testing activities so that they are timely and have the appropriate perspective for their position in the development process. This tutorial considers test techniques and test process models.

Presenter Biographies:
John D. McGregor is an associate professor of computer science at Clemson University, a founding partner of Luminary Software, and a Visiting Scientist at the Software Engineering Institute. He is co-author of two books on software engineering, including *A Practical Guide to Testing Object-Oriented Software Engineering*. McGregor teaches graduate software engineering courses and courses in the SEI's Software Product Line Curriculum and has presented numerous tutorials at a variety of conferences. He also consults with numerous software development organizations.

T12

Pragmatic Strategies for Variability Management in Product Lines in Small- to Medium-Size Companies

Stan Jarzabek, National University of Singapore, Singapore

If you deploy multiple product variants for a variety of customers, you are already in the software product line (SPL) business. Most SPLs in small- to medium-size companies evolve from a single successful product. Each new product variant is often developed by ad hoc reuse—copy and modify—of source code files implementing existing products. As the SPL practice matures, a common practice is to stabilize a product component architecture and to use preprocessing, parameter configuration files, Ant, or annotations (Java/JEE) to handle the impact of variant features at the detailed level of code. If you use these techniques, you may be aware of problems that usually emerge in time: Features get complicated and inclusion of one feature into a custom product must be properly coordinated with modifications of yet other features; core reusable components become heavily instrumented with variation points and complex to work with.

If the above picture reflects your experience, you may find this tutorial useful. We'll review techniques commonly employed for SPL variability management and their strengths and pitfalls. In the second part of the tutorial, we'll examine the XML-based Variant Configuration Language (XVCL) variation mechanism that exercises the total control over SPL variability, from architecture, to component configuration, to any detail of code (e.g., variations at the source statement, expression, or keyword level). XVCL streamlines and automates customizations involved in implementation of selected variant features into custom products, from component reconfiguration to detailed customizations of component code. The approach replaces the need for multiple variation mechanisms and avoids the problems of digging out feature customization and reuse information from SCM repositories. It complements conventional architecture-centric, component-based design for reuse and works with any conventional programming language and/or platform such as JEE, .NET, Ruby on Rails, or PHP.

In the tutorial, we discuss industrial case studies of product lines with XVCL.

Presenter Biographies:

Stan Jarzabek is an associate professor at the Department of Computer Science, School of Computing, National University of Singapore. He spent 12 years of his professional career in industry and 20 years in academia. Stan is interested in all aspects of software design, in particular, techniques for design of adaptable, easy-to-change (high-variability) software, clone detection, and program analysis. He is an author of the book *Effective Software Maintenance and Evolution: Reuse-based Approach* and has published over 90 papers in international journals and conference proceedings. (His recent paper received the ACM Distinguished Paper Award). Stan works with industries, and XVCL has been applied to manage web portal product lines at ST Electronics (Info-Software Systems) Pte Ltd. and to create mobile phone role-playing games and customer relation management systems.

T13

Using Domain-Specific Languages for Product Line Engineering
Markus Voelter, itemis AG, Germany

Domain-specific languages, together with code generation or interpreters (a.k.a. model-driven development), are becoming more and more important. Since there is a certain overhead involved in building languages and processors, this approach is especially useful in environments where a specific set of languages and generators can be reused many times. Product lines are such an environment. Consequently, the use of domain-specific languages (DSLs) for software product line engineering (SPLE) is becoming more relevant.

However, exploiting DSLs in the context of product lines involves more than just defining and using languages. This tutorial explains the differences as well as commonalities between model-driven development (MDD) and SPLE and shows how the two approaches can be combined.

In this tutorial, we will first recap/introduce feature modeling and model-driven development. We then build a simple textual DSL and a code generator based on Eclipse openArchitectureWare (oAW). Based on this language, we'll discuss the kinds of variability expressible via DSLs versus those expressible via feature modeling, leading to a discussion about ways to combine the two. In the next demo slot, we'll do just that: We'll annotate a model with feature dependencies. When generating code, the elements whose features are not selected will be removed, and hence no code will be generated. Finally, we'll discuss and demo the integration feature dependencies into code generators to configure the kind of code generated from the model.

Presenter Biography:
Markus Völter works as an independent researcher, consultant, and coach for itemis AG in

Stuttgart, Germany. His focus is on software architecture, model-driven software development, and domain-specific languages, as well as on product line engineering. He coaches projects small to large, in business, science and embedded systems, trying to bridge the gaps between these worlds. Markus also regularly writes (articles, patterns, books) and speaks (trainings, conferences) on those subjects. Contact him via voelter@acm.org or www.voelter.de.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

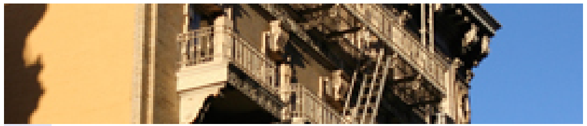
Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Become an SEI Member and Save on Registration for SPLC 2009

The Software Product Line Conference (SPLC) is the premier forum for product line researchers, practitioners, and educators to present and discuss current and emerging trends. SPLC provides the product line community with opportunities to hear industry leaders' real-world experiences and researchers' latest ideas, and to learn from both. Now in its 13th year, SPLC 2009 will be held August 24 – 28 in San Francisco, California.

You can save 15% on the already reduced early-bird registration price for SPLC 2009 by becoming a Member of the Software Engineering Institute. SEI Members save on registrations for SEI-sponsored conferences and courses and enjoy opportunities to connect with others in their fields.

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

BECOME AN SEI MEMBER NOW AND SAVE »

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ______ Join Now

SPLC 2009 on LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

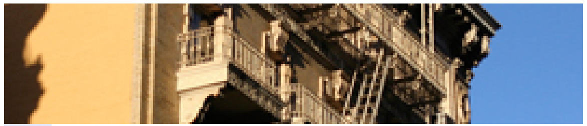
Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

SPLC 2009 Program Committee

Program Chair
John D. McGregor, *Clemson University, USA*

Program Committee
Muhammad Ali Babar, *Lero, University of Limerick, Ireland*
David Benavides, *University of Seville*
Jan Bosch, *Intuit, USA*
Manfred Broy, *TU Munich, Germany*
Paul Clements, *Software Engineering Institute, USA*
Krzysztof Czarnecki, *University of Waterloo, Canada*
Stuart Faulk, *University of Orgeon, USA*
Xavier Franch, *Universitat Politècnica de Catalunya, Spain*
Birgit Geppert, *Avaya Labs, USA*

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Stefania Gnesi, *ISTI-CNR, Italy*

Oystein Haugen, *SINTEF and University of Oslo, Norway*

Patrick Heymans, *University of Namur - FUNDP, Belgium*

Isabel John, *Fraunhofer IESE, Germany*

Kyo Kang, *University Pohang, Korea*

Tomoji Kishi, *JAIST, Japan*

Peter Knauber, *HS Mannheim, Germany*

Philipp Kutter, *Montages, Switzerland*

Patricia Lago, *Vrije University Amsterdam, The Netherlands*

Robyn Lutz, *Iowa State University & Jet Propulsion Lab, USA*

Andreas Metzger, *University of Duisburg-Essen, Germany*

Maurizio Moriso, Politecnico di Torino, Italy

Eila Niemelä, *VTT Technical Research Centre of Finland, Finland*

Liam O'Brien, *NICTA, Australia*

Rob van Ommering, *Philips, The Netherlands*

Robert Nord, *Software Engineering Institute, USA*

Daniel Paulish, *Siemens, USA*

Juha Savolainen, *Nokia, Finland*

Doug Schmidt, *Vanderbilt University, USA*

Steffen Thiel, *Lero, University of Limerick, Ireland*

Tim Trew, *NXP, The Netherlands*

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: Join Now

SPLC 2009 ON Linkedin

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

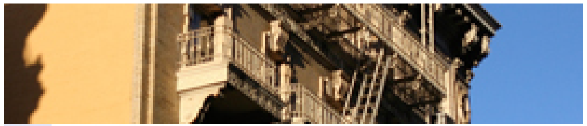
August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

SPLC 2009 Doctoral Symposium

Monday, August 24, 2009	
1:30 – 1:40 PM	Welcome Session
1:40 – 2:00 PM	Achieving True Flexibility of SOA-Based Information Systems by Adopting Practices from Product Line Engineering
2:00 – 2:20 PM	Discussion Paper 1
2:20 – 2:40 PM	Streamlining Digital Games Development Through Software Factories Automation

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

2:40 – 3:00 PM	Discussion Paper 2
3:30 – 3:50 PM	Towards a Model-Driven Product Line for SCM Systems
3:50 – 4:10 PM	Discussion Paper 3
4:10 – 4:30 PM	Rationale-Enriched Variability Management in Software Product Lines
4:30 – 4:50 PM	Discussion Paper 4
4:50 – 5:00 PM	Closing

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Third International Workshop on Dynamic Software Product Lines (DSPL 2009)

Description

In domains such as ubiquitous computing, pervasive computing, service robots, unmanned aerial vehicles, and so forth, the importance and complexity of software are increasing more than ever. These domains are characterized above all by extensive variation both in requirements and resource constraints. The software product line (SPL) approach has been receiving increased attention as a means to cope with this, specifically as software engineers and developers are faced with increasing pressure to deliver high-quality software more quickly and economically.

More importantly, modern computing and network environments demand a high degree of

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

adaptability from software systems. Computing environments, user requirements, and interface mechanisms between software and hardware devices like sensors may change dynamically during runtime. Therefore, in these kinds of dynamic environments, the application of SPL needs to be changed from a static perspective to a dynamic perspective, where systems capable of modifying their own behavior with respect to changes in its operating environment are achieved by dynamically rebinding variation points at runtime. This is the idea of dynamic software product lines (DSPL).

DSPL is an emerging and promising area of research, with clear overlaps to other areas of research in addition to SPL, notably: self-* (adapting/managing/healing ...) systems, dynamic architectures, and agent-oriented software engineering. The objective of this workshop is to solicit ideas, research directions, and results of SPL that employ and support dynamism in the manner outlined above.

Important Dates

Submission Deadline: May 24, 2009
Notifications to Authors: June 19, 2009
Camera-Ready Papers: July 1, 2009

Contact Information

Mike Hinchey, Lero, the Irish Software Engineering Research Centre, Limerick, Ireland
mike.hinchey@lero.ie

To find out more, go to the DSPL 2009 workshop homepage.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

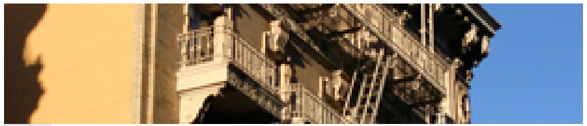
First International Workshop on Model-Driven Approaches in Software Product Line Engineering (MAPLE 2009)

Description

Many of the benefits expected from software product lines (SPLs) are based on the assumption that the additional investment in setting up a product line pays off later when products are created. However, to fully exploit this we need to optimize application engineering processes and handle SPL artifacts in a systematic and efficient manner. This workshop explores how model-driven approaches can help to achieve these goals. In particular the workshop revolves around three themes:

1) efficient product derivation – The true return on investment in product line engineering is achievable when the product lines can be used efficiently for product derivation. How can application engineering benefit from model-driven and aspect-oriented approaches?

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

2) link PLE research and industry practice – We have to overcome the gap between research and industrial practice so that both sides can learn from each other. Hence, we are particularly interested in experience reports that discuss the use of models in real-world PLE projects.

3) SPL models with a meaning – If we want to improve product derivation, we require models that are more than just vehicles for documentation and discussions on the whiteboard: models that are precise and expressive enough to be used for automation and in advanced interactive tools. However, if the existing models are documentary and ambiguous, how do we achieve more precise models?

Important Dates

Submission Deadline: ~~June 5, 2009~~ - **extended to June 10, 2009**
Notifications to Authors: June 19, 2009
Camera-Ready Papers: July 1, 2009

Contact Information

Goetz Botterweck, Lero, University of Limerick, Ireland
goetz.botterweck@lero.ie

For more information, go to the MAPLE 2009 workshop homepage.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC »

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ☐ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Workshop on Scalable Modeling Techniques for Software Product Lines (SCALE 2009)

Description

Modeling techniques play essential roles in software product line development (PLD), and various modeling techniques have been proposed so far. However, some of these techniques are not actually usable in the industries, due to the lack of scalability. Although modeling techniques are essentially for reducing scale and complexity, further development of techniques are indispensable to manage the scale and complexity we are confronting today. Especially in PLD, the problem becomes more serious, because we have to model target domains, requirements, architectures, and designs along with complicated variabilities and configurations— which is especially challenging if the product line is service based We thus need scalable modeling techniques that can handle such spatial and temporal

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

complexity and service orientation, based on careful study of actual modeling problems in industries.

The objective of this workshop is to bring together both researchers and practitioners to discuss the strengths and limitations of the current modeling techniques for supporting large-scale and service-based PLD. The workshop will provide a forum to share ideas and experiences about the current modeling approaches and to identify the future research directions related to scalable modeling techniques for PLD. We expect that this workshop will deepen mutual understanding among researchers and practitioners and promote the development of scalable modeling techniques usable in the field.

Important Dates

Submission Deadline: June 1, 2009
Notification of Acceptance: June 15, 2009
Camera-Ready Version: July 1, 2009

Contact Information

Tomoji Kishi, faculty of science and engineering, Waseda University, Japan
kishi@waseda.jp

For more information, go to the SCALE 2009 workshop homepage.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:
* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL)—Enhancing Variation

Description

The Service-Oriented Architectures and Software Product Lines (SOAPL) 2009 Workshop is the third workshop to examine the connection between service-oriented architecture and software product line approaches. While the first two workshops examined the connection between the approaches and the experience of integrating them, this workshop will examine how the two techniques benefit each other.

Workshop Presentations

2009 Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

2009): Enhancing Variation
Bob Krut and Sholom Cohen
View presentation »

Context Setting for Afternoon Discussion
Sholom Cohen
View presentation »

Service Oriented Product Line Engineering (A Negotiation Framework for Service-Oriented Product Line Development)
Jaejoon Lee
View presentation »

Managing SOA System Variation through Business Process Lines and Process Oriented Development
Nicola Boffoli, Marta Cimitile, Fabrizio Maria Maggi, Giuseppe Visaggio
View paper »
View presentation »

Towards an Approach for Service-Oriented Product Line Architectures
Flávio Mota Medeiros, Eduardo Santana de Almeida,
Silvio Romero de Lemos Meira
View paper »
View presentation »

Semantic Variability Modeling for Multi-staged Service Composition
Bardia Mohabbati, Nima Kaviani, Dragan Gasevic
View paper »
View presentation »

Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions
Dennis Smith, Grace Lewis
View paper »
View presentation »

Motivation

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON Linked in

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

Product lines and service-oriented architecture are approaches to meet a common business goal—reuse—using the same functionality in multiple contexts. In the case of product lines, reuse happens by using a “core asset” (could be a hardware or a software product) in multiple systems of a family of products (e.g., cell phone or medical records management systems). Uses of core assets vary depending on the system context. In a service-oriented system, reuse happens by using the same business functionality in multiple process, workflow, or application contexts.

Product line development is a proven approach that has been adopted by key industry players. On the other hand, although industry has accepted the benefits of a service-oriented approach for systematic reuse, the implementation methods are still new and emerging.

The SOAPL Workshop gives participants an opportunity to examine how the two techniques benefit each other.

Objectives

This workshop will build on the results of the SOAPL 2007 Workshop: Service-Oriented Architectures and Product Lines - What Is the Connection? and the SOAPL 2008 Workshop: Service-Oriented Architectures and Product Lines – Putting Them Together. This year’s workshop, SOAPL 2009, will explore how service-oriented architectures and software product lines can benefit from each other, specifically

- how service-oriented systems can benefit from software product lines' variation management approaches to identify and design services targeted to multiple service-oriented systems

- how software product lines can benefit from service-oriented architectures by employing services as a mechanism for variation within a product line

Topics and Intended Audience

Topics of interest for the workshop include both research and practitioner perspectives.

From a research perspective, how can these two reuse approaches benefit from each other?

1. Using service orientation in a product line:

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

a. implementing the software core assets of the product line as services
b. using service invocation as a variation mechanism
c. using service-oriented approaches to support end-user variations

2. Using product line concepts in a service-oriented context:
   a. using concepts of variability and commonality analysis to understand and model requirements variability that will eventually help identify the right set of services that can be used in multiple contexts/environments
   b. once the variability and commonality in services have been identified, engineering this variability into the services
   c. exploiting variation points for service versioning, which is a major problem in service-oriented systems, both at design time and runtime

From the practitioner perspective, three topics of interest are

1. What product line practices have been used to successfully govern assets that were created using a service-oriented approach?
2. What approaches have worked (and not worked) in the migration of legacy software to services for use in multiple applications?
3. What are the underlying infrastructures that support product lines based on implementations of service-oriented architecture?

Participants in SOAPL 2009 will include research and practitioners who have experience in service-oriented architecture, software product lines, and variation management issues.

Submissions

Prospective participants must submit a five-to-eight-page position paper or experience report that pertains to the workshop topics listed. Papers should be in the IEEE Computer Society Conference Format for 8.5x11-inch Proceedings Manuscripts.

All submissions will be reviewed by members of the program committee for quality and relevance. Accepted papers will be published electronically in the conference proceedings. Three or four papers will be chosen to be presented during the workshop to foment discussion.

Important Dates

Submit your paper in PDF form to soa-workshop@sei.cmu.edu by June 1, 2009.

Notifications of paper or experience report acceptance will be sent by June 15, 2009. Camera-ready copies are due July 1, 2009.

Format and Program

The workshop will be highly interactive and focus on determining how best to integrate service-oriented architecture and product line practices. The morning session will feature invited speakers and selected presentations based on position papers. Participants in the afternoon session will be assigned to working groups that cover specific topics of interest. After the workshop, the leader of each working group will be asked to write a summary of the working group’s discussion and (especially) its conclusions.

Contact Information

For more information, contact

Robert Krut
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Phone: +1-412-268-8505
Fax: +1-412-268-5758
Email: rk@sei.cmu.edu

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

SPLC 2009

Software Engineering Institute | CarnegieMellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

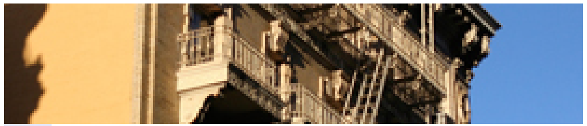
Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Workshop on Consolidating Community Consensus in Product Line Practice

Description

This workshop is intended as the first step towards capturing and codifying proven and effective product line engineering concepts, models, and practices. One way to capture and codify is by creating a standard. The goals of this workshop are to explore the possibility of creating a lightweight standard for product line engineering that defines what constitutes a software product line, to determine what minimum engineering processes must be part of a true product line engineering effort, and to list a number of proven practices. While this workshop cannot produce such a standard, it can set the process in motion and provide direction for the effort.

This year’s program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »



SPLC NEWS

- Five Reasons Why You Can’t Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Important Dates

Submission Deadline: July 1, 2009
Notifications to Authors: August 1, 2009
Camera-Ready Papers: NA

Contact Information

Paul Clements, Software Engineering Institute, USA
clements@sei.cmu.edu

For more information, go to the workshop homepage.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP's Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: ▭ Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:
* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.

Subscribe to the SPLC Conference news feed.

SPLC 2009

Software Engineering Institute | Carnegie Mellon

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA

Home Call for Participation Program Speakers Travel & Venue Sponsors SPLC.NET

Panels

Wednesday, August 26, 2009

2 PM – Working Session 1

This year's program included keynotes by leaders in the field, experience reports from industry, presentations on current research, and product line workshops, tutorials, and tool demos. View the entire program here »

- Five Reasons Why You Can't Afford to Miss SPLC 2009
- Learn from experts on Feature-Oriented Domain Analysis (FODA) and feature

Future Directions: The View from the Labs
In this session, representatives from three research labs will share their views on future directions in product line practice in short presentations. The majority of the session will be spent in audience participation discussing future directions.

Nobuaki Kozuka
Yuzo Ishida
NOMURA RESEARCH Institute, Ltd.

Ralf Carbon
Fraunhofer IESE

Linda M. Northrop
Software Engineering Institute

Thursday, August 27, 2009

2 PM – Working Session 1 *(continued)*

This is a continuation of Working Session 1.

A brief summary of the presentations and discussion from Wednesday's session will be given to initiate audience participation.

Nobuaki Kozuka
Yuzo Ishida
NOMURA RESEARCH Institute, Ltd.

Ralf Carbon
Fraunhofer IESE

Linda M. Northrop
Software Engineering Institute

Friday, August 28, 2009

2 PM – Working Session 2

modeling at SPLC»

- Take advantage of expert-led tutorials at SPLC August 24-28 »
- Kyo Kang, Originator of Feature-Oriented Domain Analysis (FODA), to Keynote at SPLC 2009 »
- SAVE 15% on registration for SPLC with your SEI Membership »
- HP’s Jacob G. Refstrup to keynote at SPLC »
- IBM's Dick Gabriel to Keynote at SPLC 2009 »
- Find out how to become a sponsor of SPLC 2009 »
- Conference to be held in San Francisco, California August 24-28, 2009 »

CONNECT WITH SPLC

Join the SPLC 2009 Conference Mailing List and stay up to date:

To be included in the SPLC 2009 Conference mailing list for latest updates, sign up here:

Email: [ ] Join Now

SPLC 2009 ON LinkedIn

COMMITTEE MEMBERS

Organizing Committee Members

General Chair: Dirk Muthig, Lufthansa Systems Passenger Services GmbH

Program Chair: John McGregor, Clemson University, USA

Industry Track:

Quality Assurance in Software Product Lines

What works and what doesn't? How do you trace quality attributes from elicitation to code in your product lines? Researchers will pose problems they've seen regarding the traceability of quality attributes, and the audience will engage in an open discussion of the successes, failures, and unmet needs they've experienced.

Robyn R. Lutz
Jet Propulsion Lab, NASA
Iowa State University

Len J. Bass
Software Engineering Institute

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

Subscribe to the SPLC Conference news feed.

* Paul Jensen, Overwatch, USA
* Kentaro Yoshimura, Hitachi, Japan
* Michael Schumpelt, ETAS, Germany

Workshops: Jaejoon Lee, Lancaster University, UK

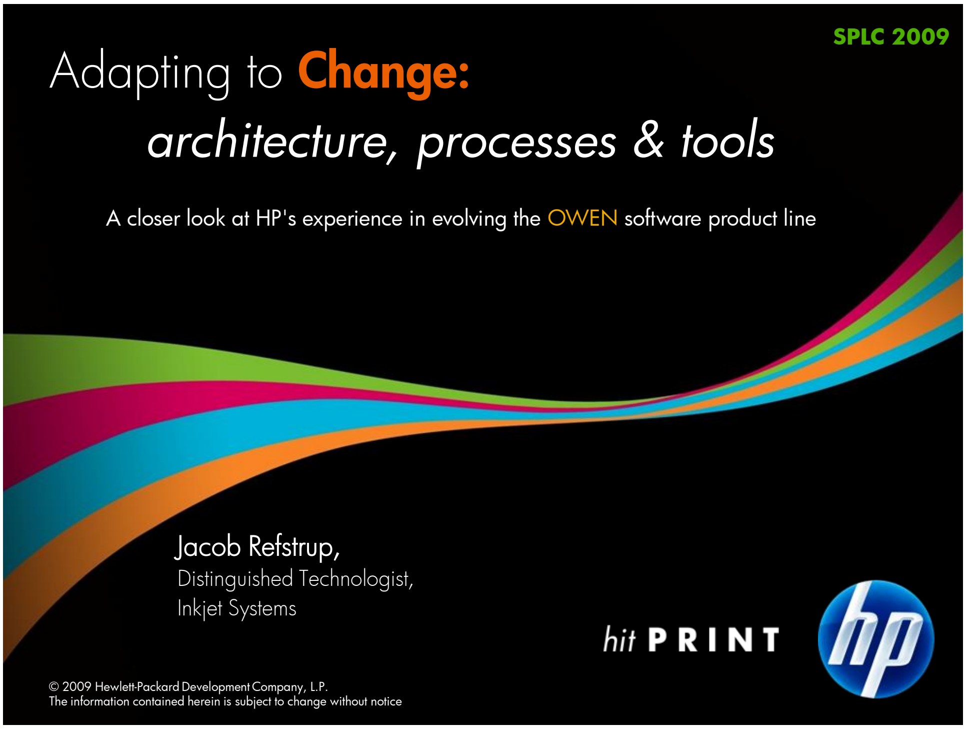
Demonstrations & Posters: Ronny Kolb, Honeywell, Switzerland

Tutorials: Gary Chastek, Software Engineering Institute, USA

Doctoral Symposium: Eduardo Santana de Almeida, C.E.S.A.R., Brazil

Publicity: Pat Donohoe, Software Engineering Institute, USA

See the list of program committee members.



SPLC 2009

Adapting to **Change:**
architecture, processes & tools

A closer look at HP's experience in evolving the OWEN software product line

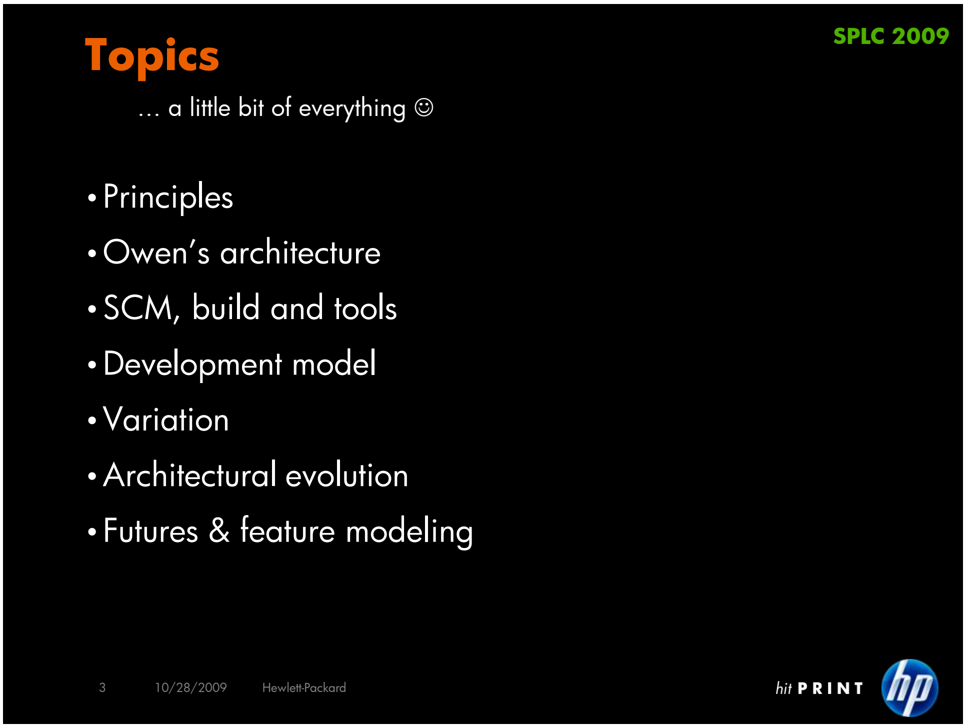
Jacob Refstrup,
Distinguished Technologist,
Inkjet Systems

SPLC 2009

What's **Owen**?

- A embedded software product line architecture
- Used in multiple Hewlett-Packard product lines
  - Deskjet, Photosmart, Officejet and Officejet Pro
- First product intro in '98
- Same fundamental architecture in place
  - Evolved architecture/design of subsystems
- Lots of tools & process changes
  - From co-op to full re-use

2 10/28/2009 Hewlett-Packard



SPLC 2009

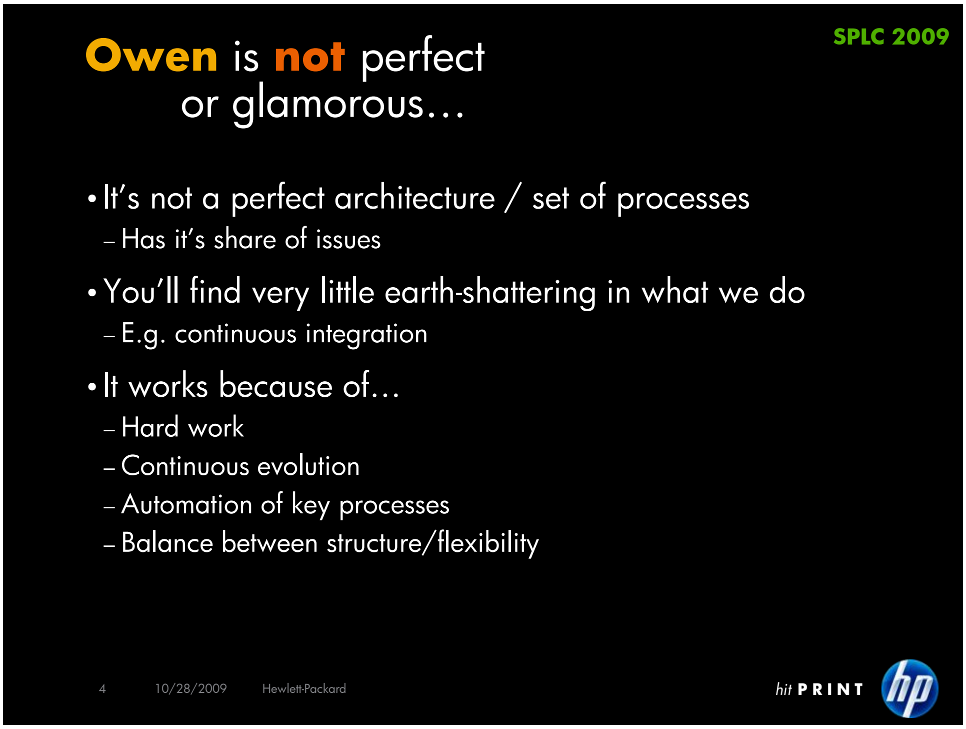
Topics

… a little bit of everything ☺

- Principles
- Owen’s architecture
- SCM, build and tools
- Development model
- Variation
- Architectural evolution
- Futures & feature modeling

3 10/28/2009 Hewlett-Packard

hit **PRINT**



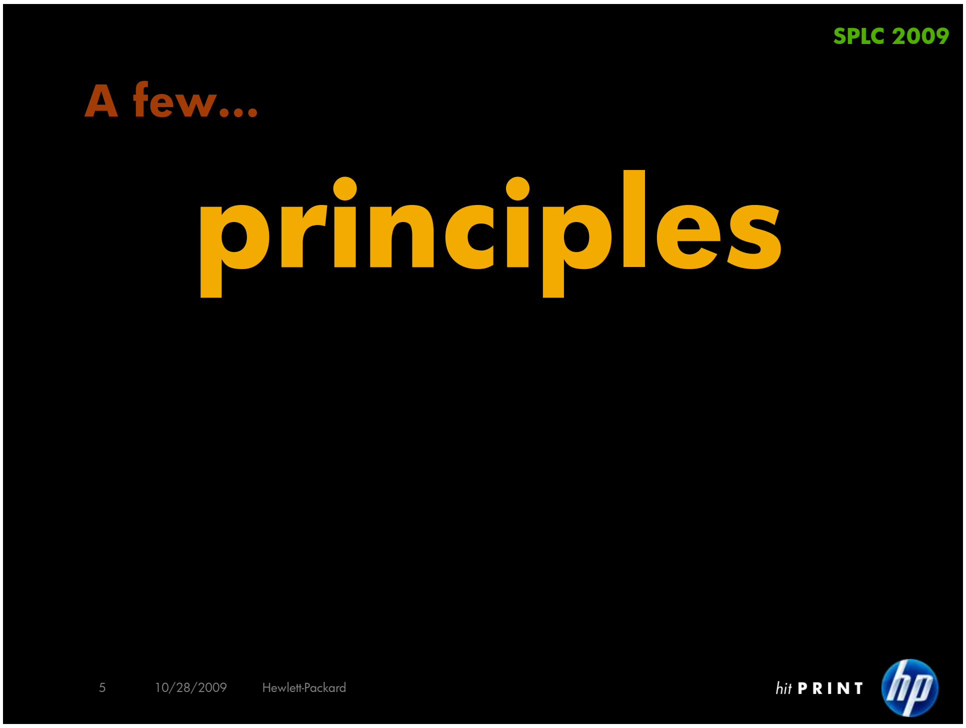
SPLC 2009

Owen is not perfect or glamorous…

- It’s not a perfect architecture / set of processes
  - Has it’s share of issues
- You’ll find very little earth-shattering in what we do
  - E.g. continuous integration
- It works because of…
  - Hard work
  - Continuous evolution
  - Automation of key processes
  - Balance between structure/flexibility

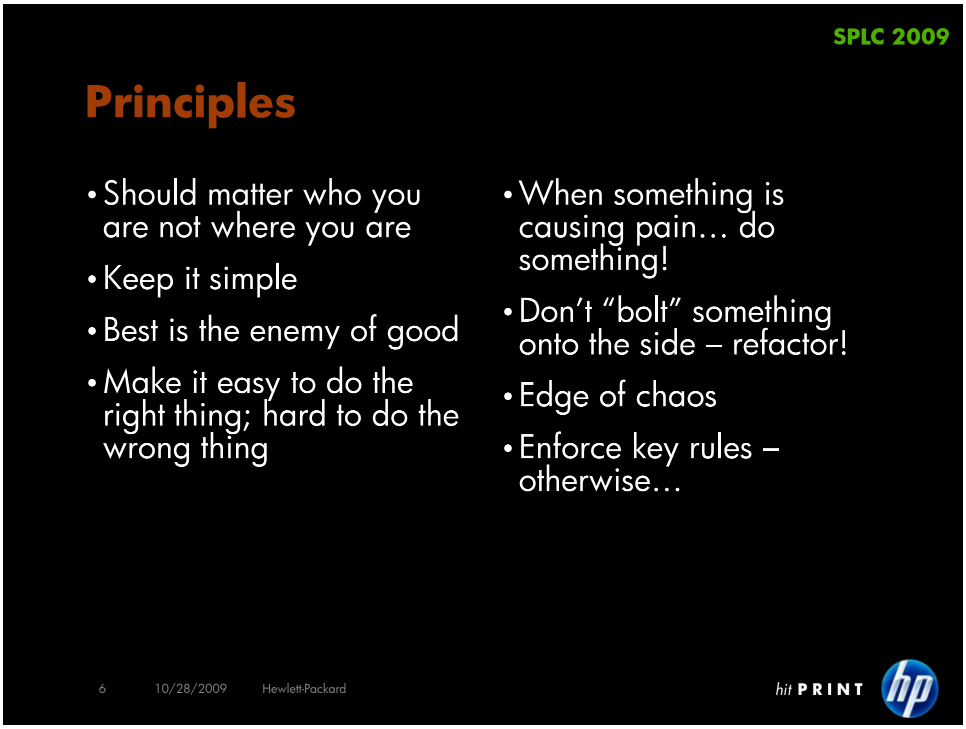
4 10/28/2009 Hewlett-Packard

hit **PRINT**



A few…

principles



Principles

- Should matter who you are not where you are
- Keep it simple
- Best is the enemy of good
- Make it easy to do the right thing; hard to do the wrong thing
- When something is causing pain… do something!
- Don’t “bolt” something onto the side – refactor!
- Edge of chaos
- Enforce key rules – otherwise…

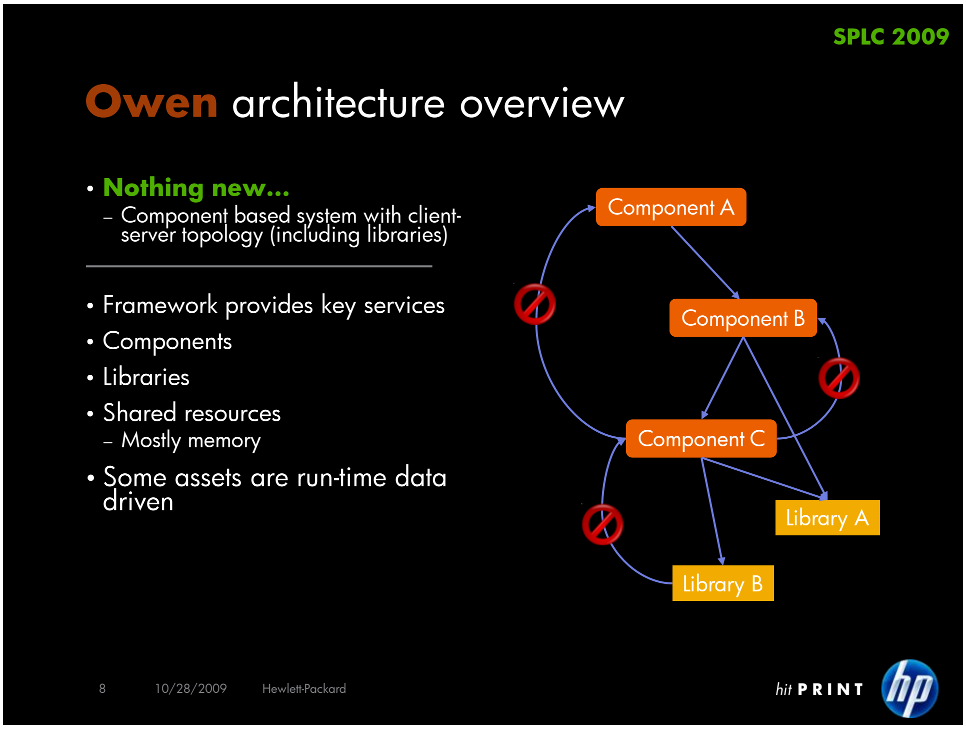
SPLC 2009

One minute tour of…

architecture

7 10/28/2009 Hewlett-Packard

hit **PRINT**



SPLC 2009

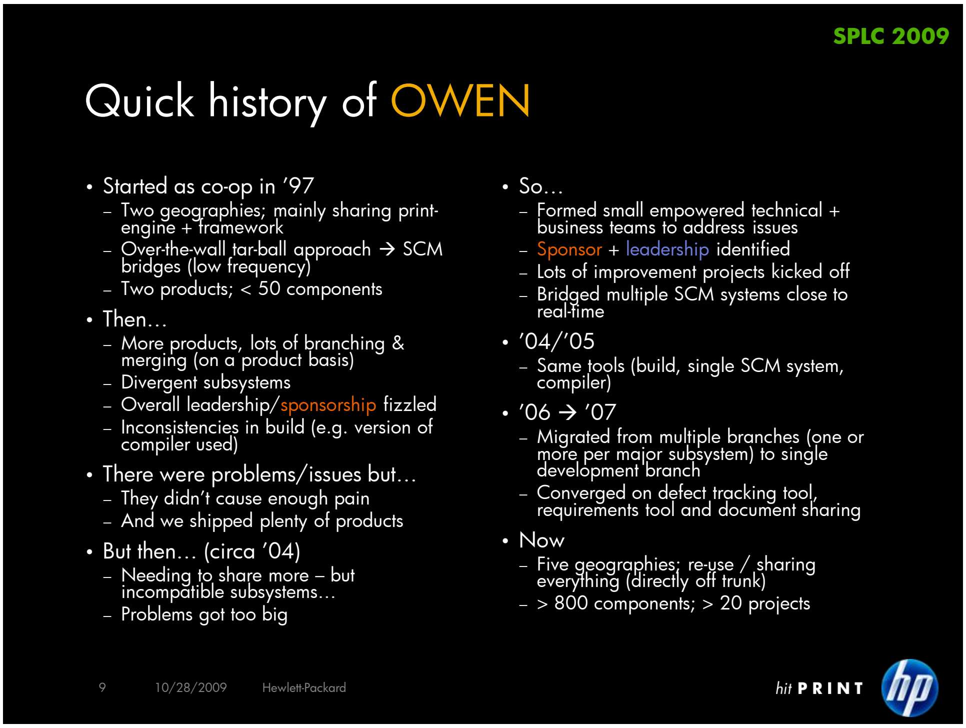
Owen architecture overview

- **Nothing new…**
  - Component based system with client-server topology (including libraries)

- Framework provides key services
- Components
- Libraries
- Shared resources
  - Mostly memory
- Some assets are run-time data driven

8 10/28/2009 Hewlett-Packard

hit **PRINT**



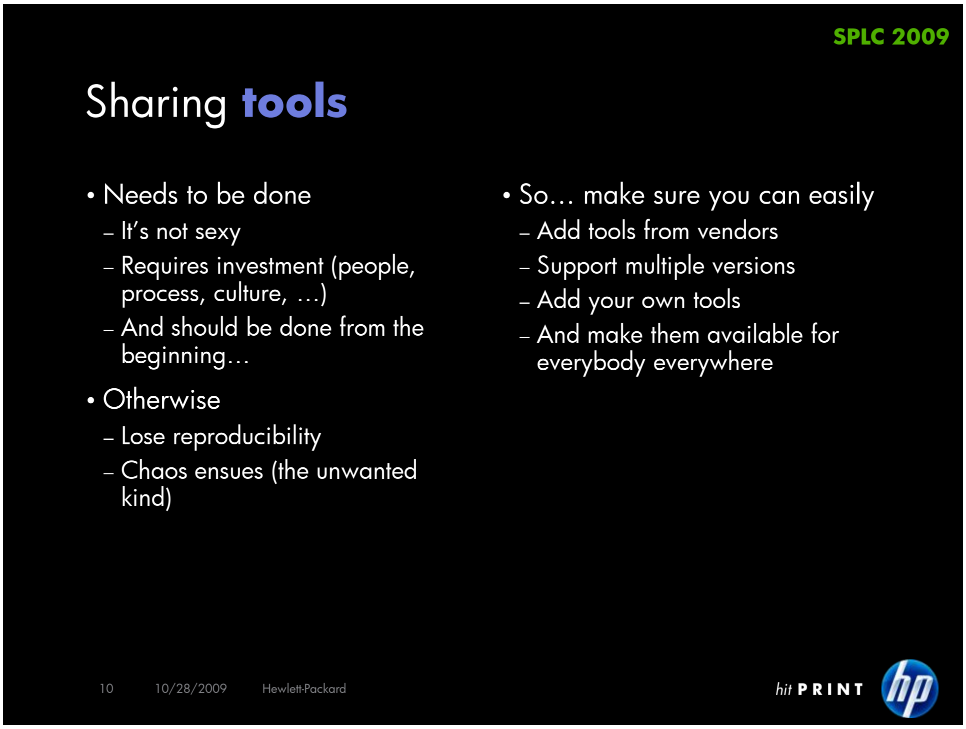
SPLC 2009

Quick history of OWEN

- Started as co-op in '97
  - Two geographies; mainly sharing print-engine + framework
  - Over-the-wall tar-ball approach → SCM bridges (low frequency)
  - Two products; < 50 components
- Then…
  - More products, lots of branching & merging (on a product basis)
  - Divergent subsystems
  - Overall leadership/sponsorship fizzled
  - Inconsistencies in build (e.g. version of compiler used)
- There were problems/issues but…
  - They didn't cause enough pain
  - And we shipped plenty of products
- But then… (circa '04)
  - Needing to share more – but incompatible subsystems…
  - Problems got too big
- So…
  - Formed small empowered technical + business teams to address issues
  - Sponsor + leadership identified
  - Lots of improvement projects kicked off
  - Bridged multiple SCM systems close to real-time
- '04/'05
  - Same tools (build, single SCM system, compiler)
- '06 → '07
  - Migrated from multiple branches (one or more per major subsystem) to single development branch
  - Converged on defect tracking tool, requirements tool and document sharing
- Now
  - Five geographies; re-use / sharing everything (directly off trunk)
  - > 800 components; > 20 projects

9 10/28/2009 Hewlett-Packard

hit **PRINT**



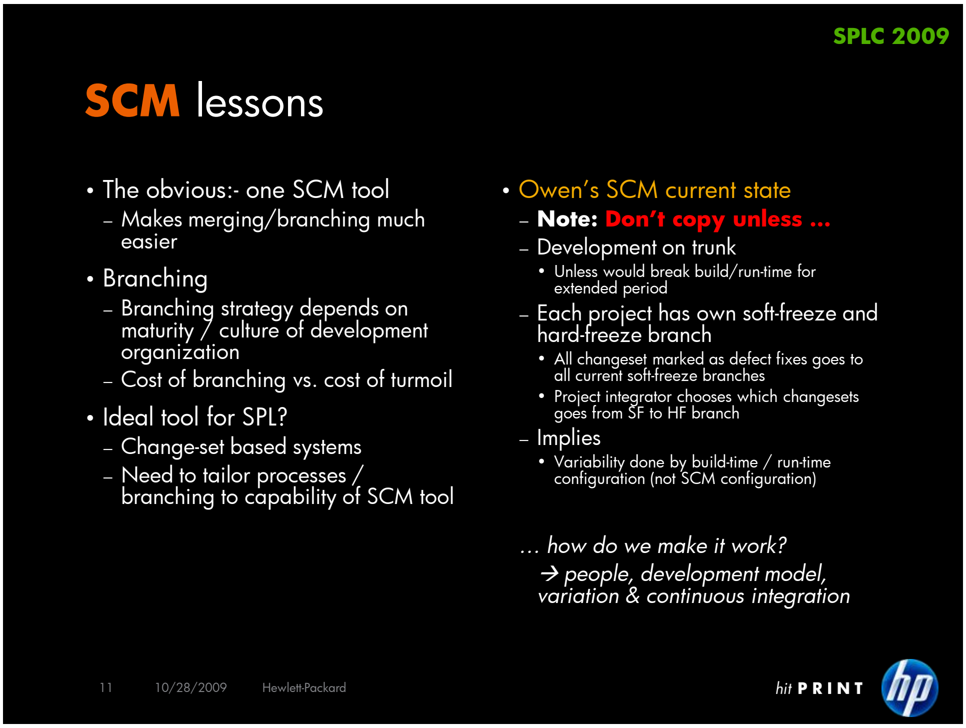
SPLC 2009

Sharing **tools**

- Needs to be done
  - It's not sexy
  - Requires investment (people, process, culture, …)
  - And should be done from the beginning…
- Otherwise
  - Lose reproducibility
  - Chaos ensues (the unwanted kind)
- So… make sure you can easily
  - Add tools from vendors
  - Support multiple versions
  - Add your own tools
  - And make them available for everybody everywhere

10 10/28/2009 Hewlett-Packard

hit **PRINT**



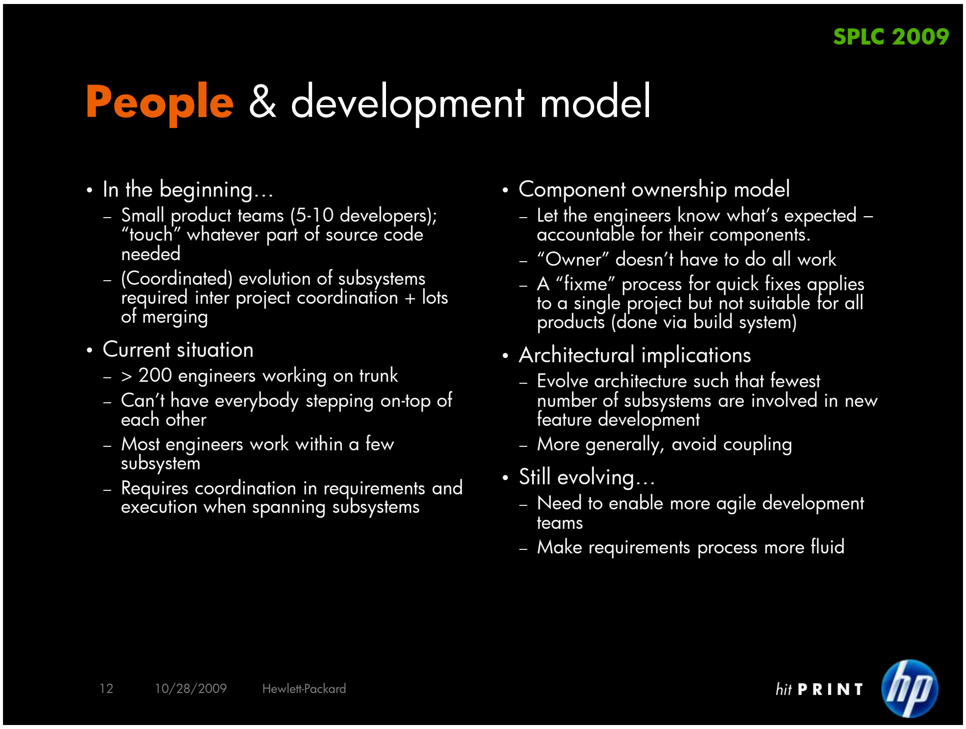
SPLC 2009

SCM lessons

- The obvious:- one SCM tool
  - Makes merging/branching much easier
- Branching
  - Branching strategy depends on maturity / culture of development organization
  - Cost of branching vs. cost of turmoil
- Ideal tool for SPL?
  - Change-set based systems
  - Need to tailor processes / branching to capability of SCM tool

- Owen's SCM current state
  - **Note: Don't copy unless …**
  - Development on trunk
    - Unless would break build/run-time for extended period
  - Each project has own soft-freeze and hard-freeze branch
    - All changeset marked as defect fixes goes to all current soft-freeze branches
    - Project integrator chooses which changesets goes from SF to HF branch
  - Implies
    - Variability done by build-time / run-time configuration (not SCM configuration)

… how do we make it work?
→ people, development model, variation & continuous integration

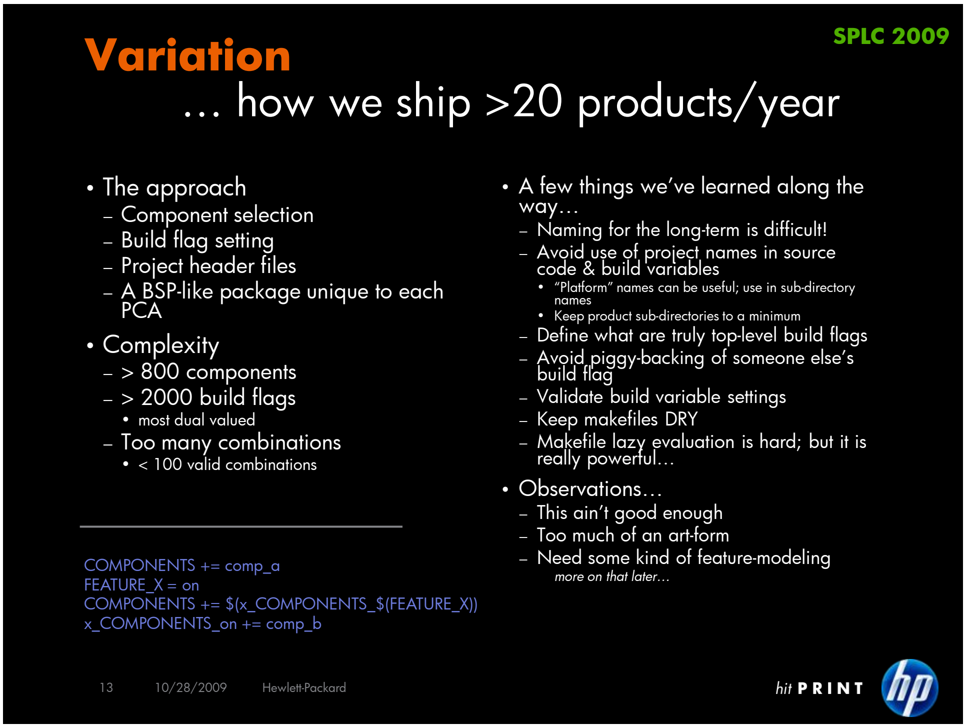


SPLC 2009

People & development model

- In the beginning…
  - Small product teams (5-10 developers); "touch" whatever part of source code needed
  - (Coordinated) evolution of subsystems required inter project coordination + lots of merging
- Current situation
  - > 200 engineers working on trunk
  - Can't have everybody stepping on-top of each other
  - Most engineers work within a few subsystem
  - Requires coordination in requirements and execution when spanning subsystems

- Component ownership model
  - Let the engineers know what's expected – accountable for their components.
  - "Owner" doesn't have to do all work
  - A "fixme" process for quick fixes applies to a single project but not suitable for all products (done via build system)
- Architectural implications
  - Evolve architecture such that fewest number of subsystems are involved in new feature development
  - More generally, avoid coupling
- Still evolving…
  - Need to enable more agile development teams
  - Make requirements process more fluid



SPLC 2009

Variation
… how we ship >20 products/year

- The approach
  - Component selection
  - Build flag setting
  - Project header files
  - A BSP-like package unique to each PCA
- Complexity
  - > 800 components
  - > 2000 build flags
    - most dual valued
  - Too many combinations
    - < 100 valid combinations

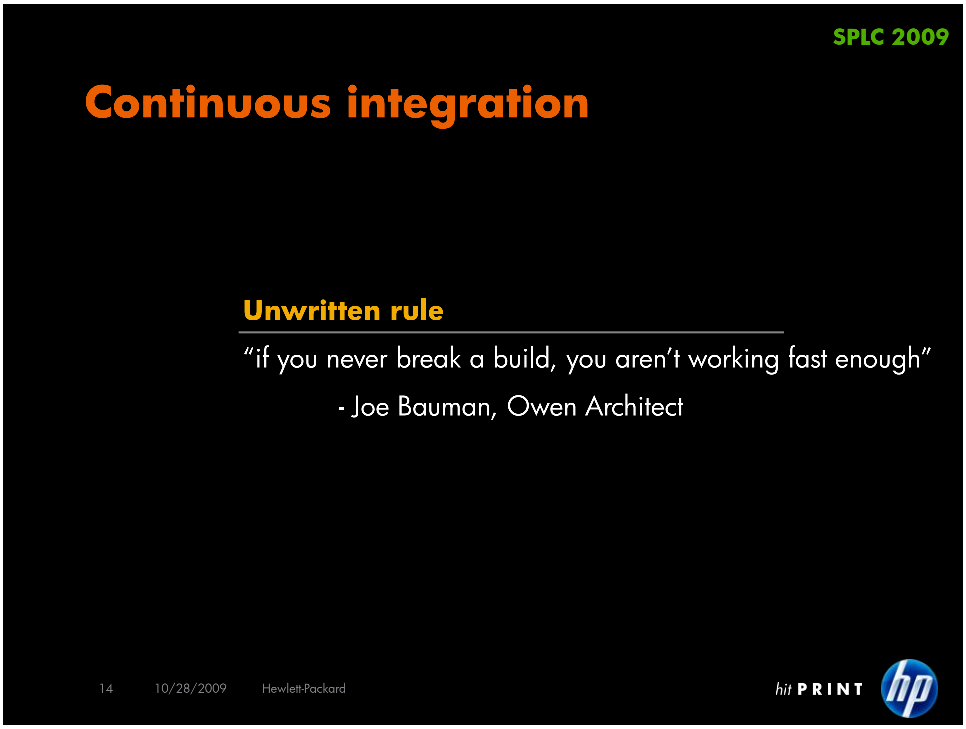
```
COMPONENTS += comp_a
FEATURE_X = on
COMPONENTS += $(x_COMPONENTS_$(FEATURE_X))
x_COMPONENTS_on += comp_b
```

- A few things we've learned along the way…
  - Naming for the long-term is difficult!
  - Avoid use of project names in source code & build variables
    - "Platform" names can be useful; use in sub-directory names
    - Keep product sub-directories to a minimum
  - Define what are truly top-level build flags
  - Avoid piggy-backing of someone else's build flag
  - Validate build variable settings
  - Keep makefiles DRY
  - Makefile lazy evaluation is hard; but it is really powerful…
- Observations…
  - This ain't good enough
  - Too much of an art-form
  - Need some kind of feature-modeling

more on that later…

13 10/28/2009 Hewlett-Packard

hit PRINT



SPLC 2009

Continuous integration

Unwritten rule

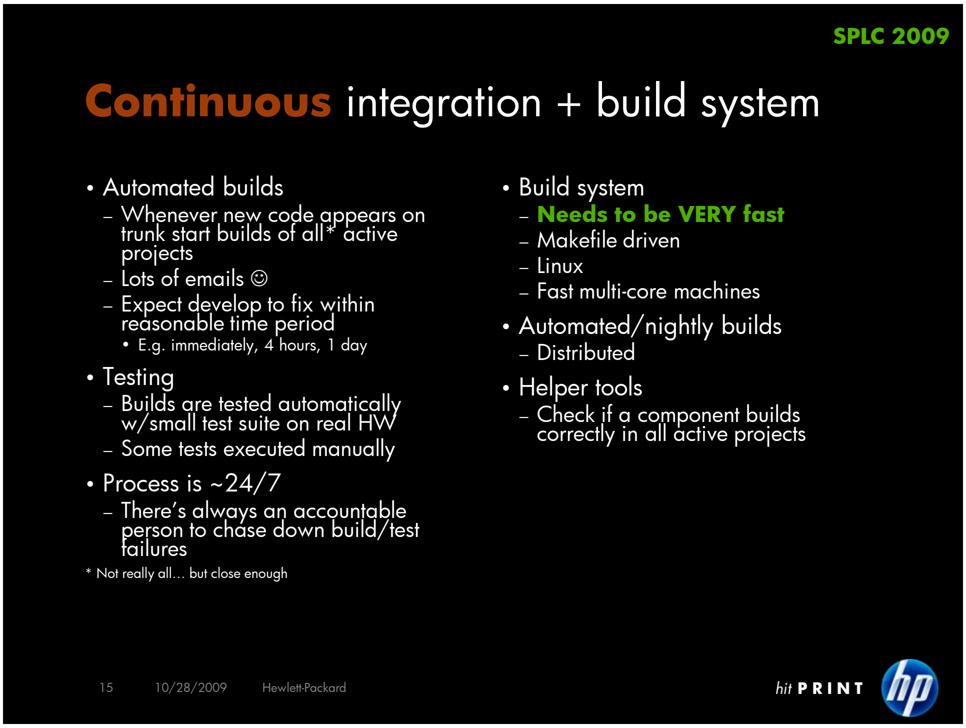
"if you never break a build, you aren't working fast enough"

- Joe Bauman, Owen Architect

14 10/28/2009 Hewlett-Packard

hit PRINT

invent



SPLC 2009

Continuous integration + build system

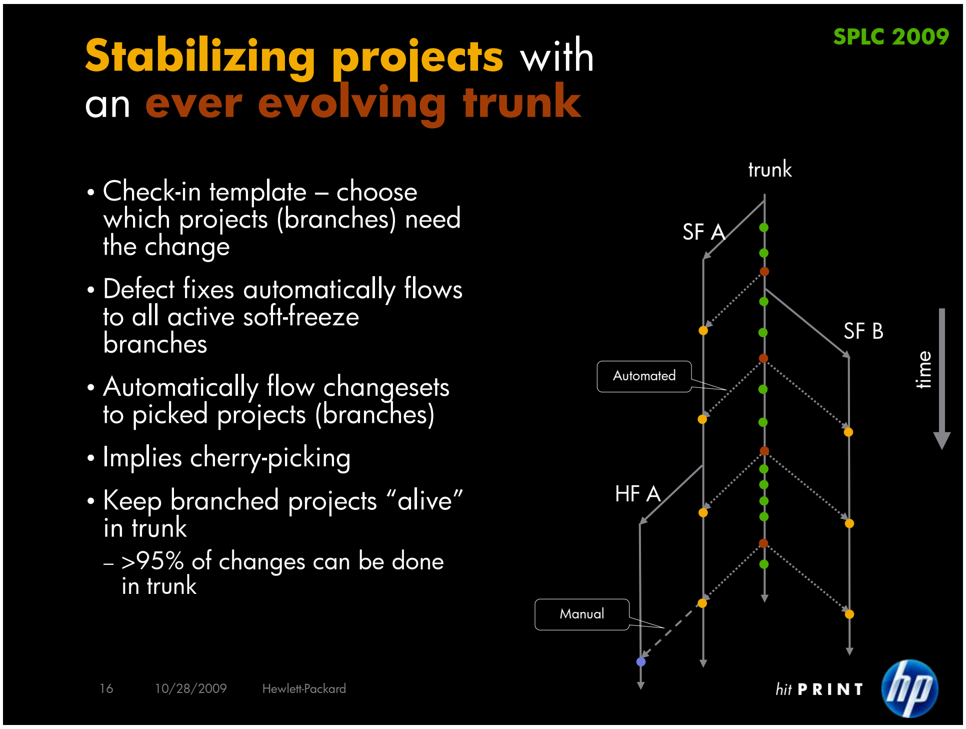
- Automated builds
  - Whenever new code appears on trunk start builds of all* active projects
  - Lots of emails ☺
  - Expect develop to fix within reasonable time period
    - E.g. immediately, 4 hours, 1 day
- Testing
  - Builds are tested automatically w/small test suite on real HW
  - Some tests executed manually
- Process is ~24/7
  - There's always an accountable person to chase down build/test failures

* Not really all… but close enough

- Build system
  - **Needs to be VERY fast**
  - Makefile driven
  - Linux
  - Fast multi-core machines
- Automated/nightly builds
  - Distributed
- Helper tools
  - Check if a component builds correctly in all active projects

15 10/28/2009 Hewlett-Packard

hit **PRINT**



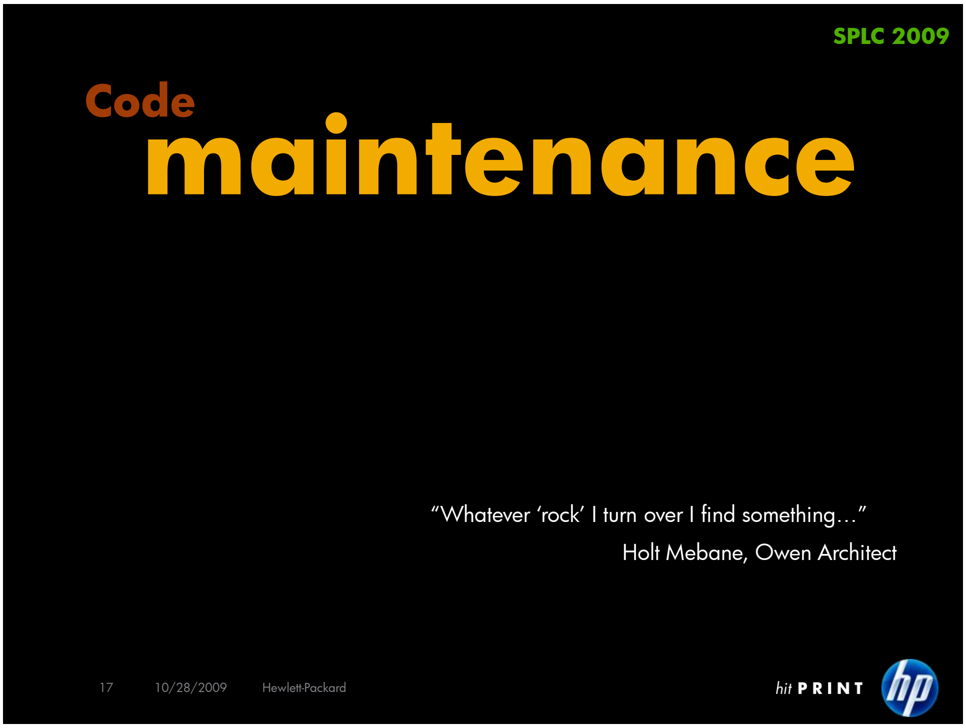
SPLC 2009

Stabilizing projects with an ever evolving trunk

- Check-in template – choose which projects (branches) need the change
- Defect fixes automatically flows to all active soft-freeze branches
- Automatically flow changesets to picked projects (branches)
- Implies cherry-picking
- Keep branched projects "alive" in trunk
  - >95% of changes can be done in trunk

16 10/28/2009 Hewlett-Packard

hit **PRINT**

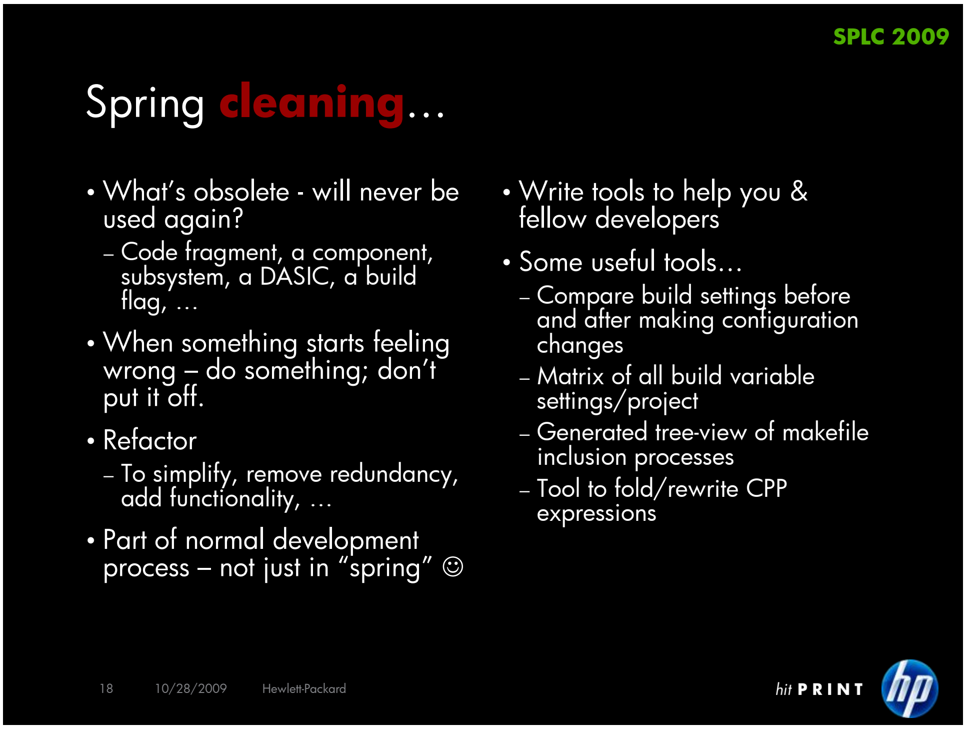


SPLC 2009

Code maintenance

“Whatever ‘rock’ I turn over I find something…”
Holt Mebane, Owen Architect

17 10/28/2009 Hewlett-Packard *hit* **PRINT**



SPLC 2009

Spring **cleaning**…

- What’s obsolete - will never be used again?
  - Code fragment, a component, subsystem, a DASIC, a build flag, …
- When something starts feeling wrong – do something; don’t put it off.
- Refactor
  - To simplify, remove redundancy, add functionality, …
- Part of normal development process – not just in “spring” ☺
- Write tools to help you & fellow developers
- Some useful tools…
  - Compare build settings before and after making configuration changes
  - Matrix of all build variable settings/project
  - Generated tree-view of makefile inclusion processes
  - Tool to fold/rewrite CPP expressions

18 10/28/2009 Hewlett-Packard *hit* **PRINT**

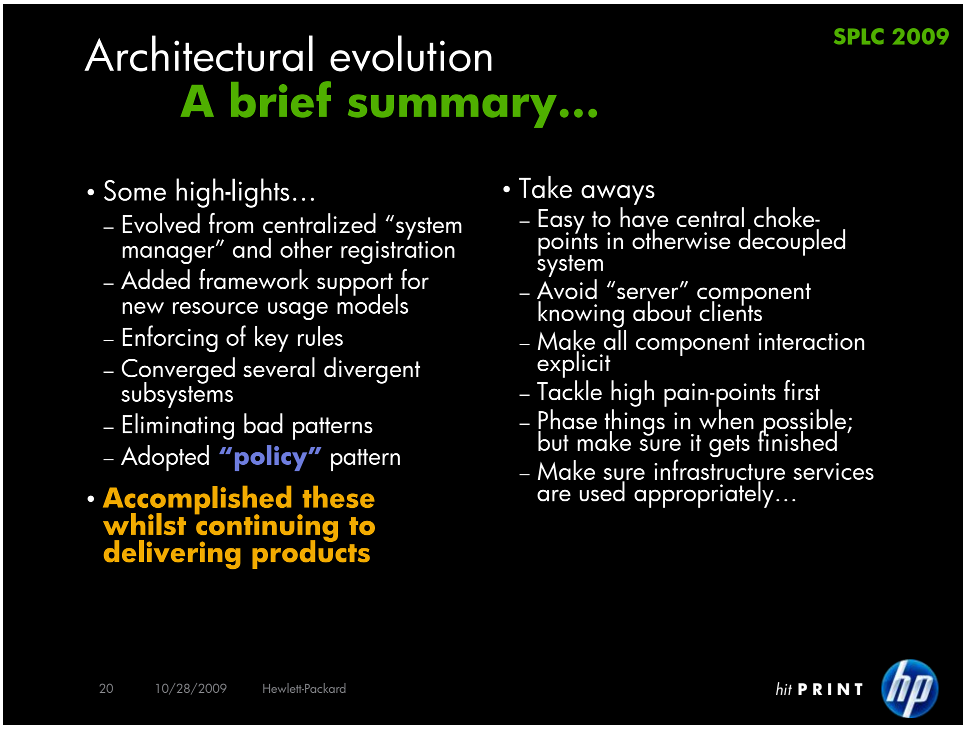
SPLC 2009

Architectural

evolution

19 10/28/2009 Hewlett-Packard

hit PRINT



SPLC 2009

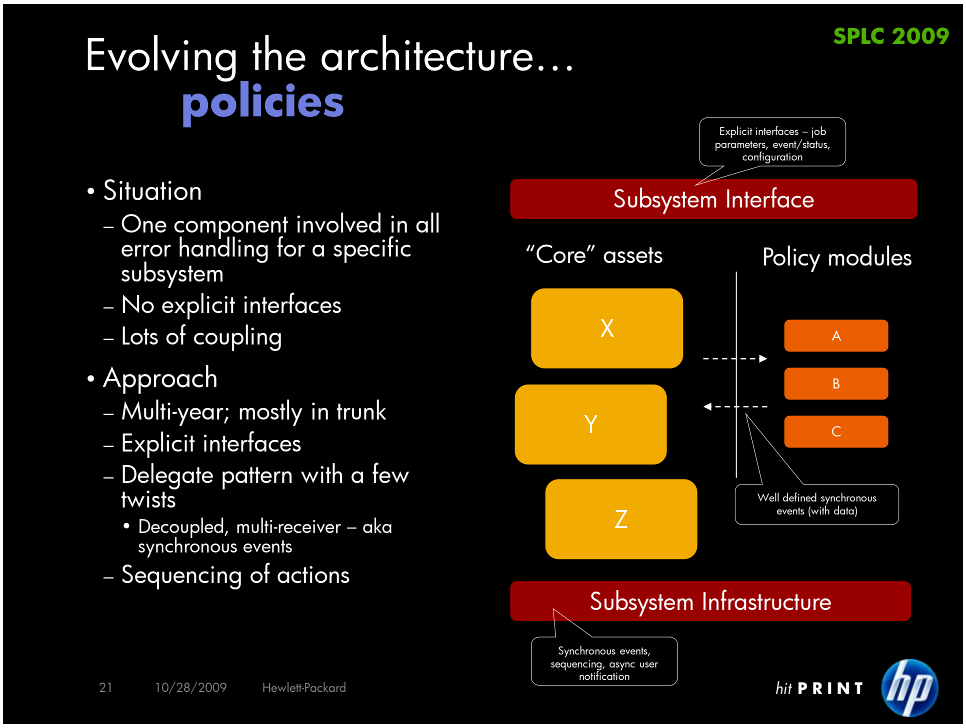
Architectural evolution
A brief summary…

- Some high-lights…
  - Evolved from centralized "system manager" and other registration
  - Added framework support for new resource usage models
  - Enforcing of key rules
  - Converged several divergent subsystems
  - Eliminating bad patterns
  - Adopted **"policy"** pattern
- **Accomplished these whilst continuing to delivering products**

- Take aways
  - Easy to have central choke-points in otherwise decoupled system
  - Avoid "server" component knowing about clients
  - Make all component interaction explicit
  - Tackle high pain-points first
  - Phase things in when possible; but make sure it gets finished
  - Make sure infrastructure services are used appropriately…

20 10/28/2009 Hewlett-Packard

hit PRINT



SPLC 2009

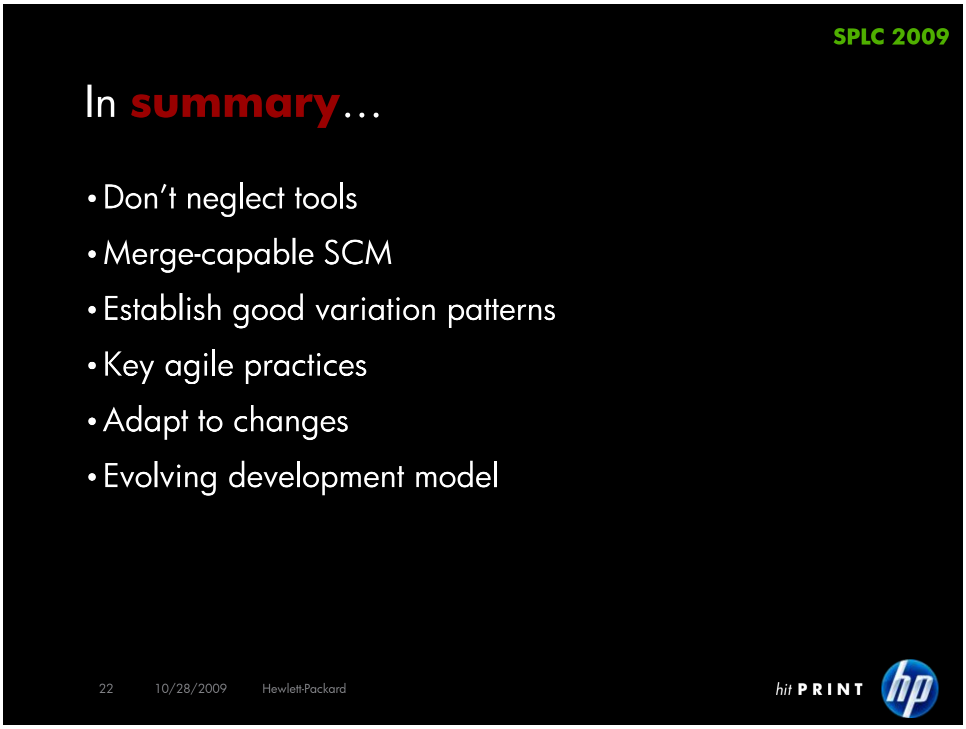
Evolving the architecture… policies

- Situation
  - One component involved in all error handling for a specific subsystem
  - No explicit interfaces
  - Lots of coupling
- Approach
  - Multi-year; mostly in trunk
  - Explicit interfaces
  - Delegate pattern with a few twists
    - Decoupled, multi-receiver – aka synchronous events
  - Sequencing of actions

Explicit interfaces – job parameters, event/status, configuration
Subsystem Interface
"Core" assets
Policy modules
X
Y
Z
A
B
C
Well defined synchronous events (with data)
Subsystem Infrastructure
Synchronous events, sequencing, async user notification

21 10/28/2009 Hewlett-Packard

hit PRINT



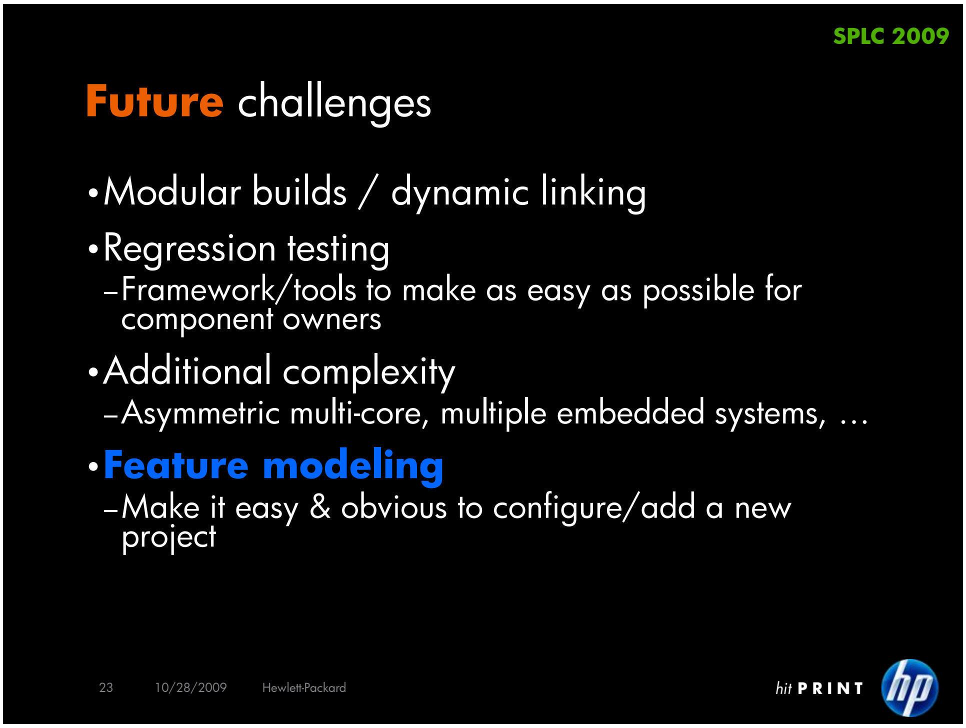
SPLC 2009

In **summary**…

- Don't neglect tools
- Merge-capable SCM
- Establish good variation patterns
- Key agile practices
- Adapt to changes
- Evolving development model

22 10/28/2009 Hewlett-Packard

hit PRINT



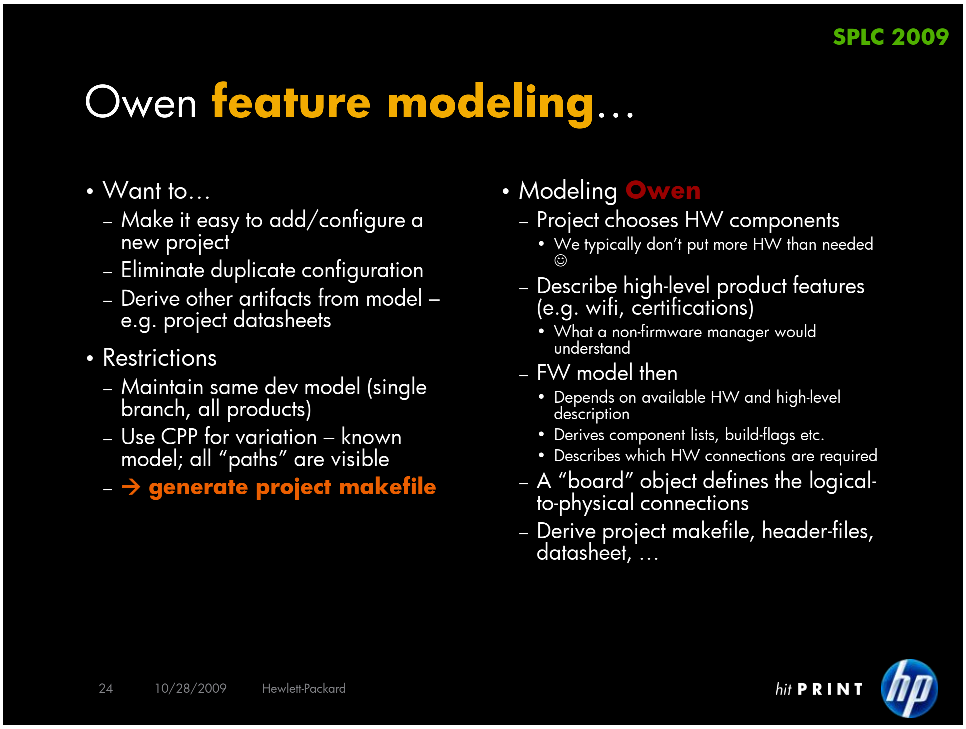
SPLC 2009

Future challenges

- Modular builds / dynamic linking
- Regression testing
  - Framework/tools to make as easy as possible for component owners
- Additional complexity
  - Asymmetric multi-core, multiple embedded systems, …
- **Feature modeling**
  - Make it easy & obvious to configure/add a new project

23 10/28/2009 Hewlett-Packard

hit **PRINT**



SPLC 2009

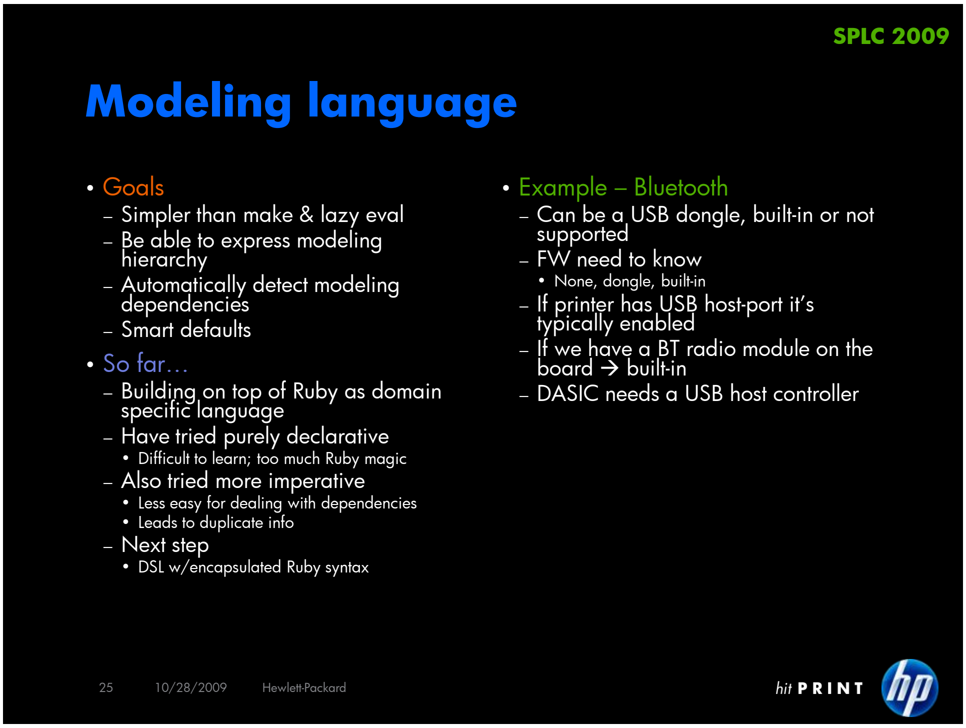
Owen **feature modeling**…

- Want to…
  - Make it easy to add/configure a new project
  - Eliminate duplicate configuration
  - Derive other artifacts from model – e.g. project datasheets
- Restrictions
  - Maintain same dev model (single branch, all products)
  - Use CPP for variation – known model; all “paths” are visible
  - → **generate project makefile**

- Modeling **Owen**
  - Project chooses HW components
    - We typically don’t put more HW than needed ☺
  - Describe high-level product features (e.g. wifi, certifications)
    - What a non-firmware manager would understand
  - FW model then
    - Depends on available HW and high-level description
    - Derives component lists, build-flags etc.
    - Describes which HW connections are required
  - A “board” object defines the logical-to-physical connections
  - Derive project makefile, header-files, datasheet, …

24 10/28/2009 Hewlett-Packard

hit **PRINT**



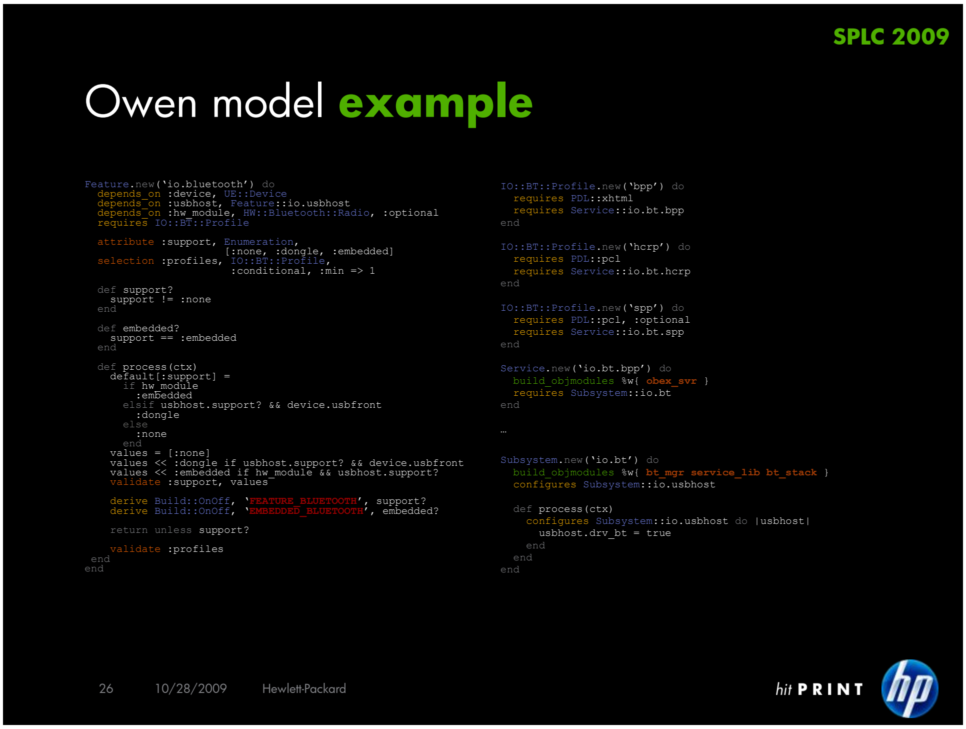
SPLC 2009

Modeling language

- Goals
  - Simpler than make & lazy eval
  - Be able to express modeling hierarchy
  - Automatically detect modeling dependencies
  - Smart defaults
- So far…
  - Building on top of Ruby as domain specific language
  - Have tried purely declarative
    - Difficult to learn; too much Ruby magic
  - Also tried more imperative
    - Less easy for dealing with dependencies
    - Leads to duplicate info
  - Next step
    - DSL w/encapsulated Ruby syntax
- Example – Bluetooth
  - Can be a USB dongle, built-in or not supported
  - FW need to know
    - None, dongle, built-in
  - If printer has USB host-port it's typically enabled
  - If we have a BT radio module on the board → built-in
  - DASIC needs a USB host controller

25 10/28/2009 Hewlett-Packard

hit **PRINT**



SPLC 2009

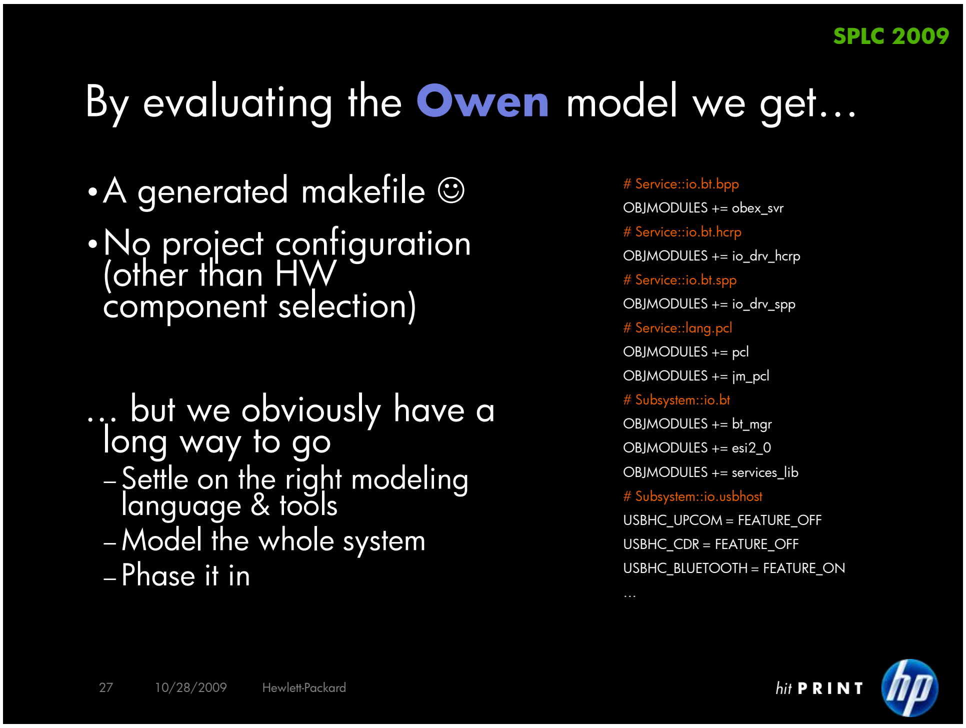
Owen model example

```
Feature.new('io.bluetooth') do
  depends_on :device, UE::Device
  depends_on :usbhost, Feature::io.usbhost
  depends_on :hw_module, HW::Bluetooth::Radio, :optional
  requires IO::BT::Profile

  attribute :support, Enumeration,
                      [:none, :dongle, :embedded]
  selection :profiles, IO::BT::Profile,
                       :conditional, :min => 1

  def support?
    support != :none
  end

  def embedded?
    support == :embedded
  end

  def process(ctx)
    default[:support] =
      if hw_module
        :embedded
      elsif usbhost.support? && device.usbfront
        :dongle
      else
        :none
      end
    values = [:none]
    values << :dongle if usbhost.support? && device.usbfront
    values << :embedded if hw_module && usbhost.support?
    validate :support, values

    derive Build::OnOff, 'FEATURE_BLUETOOTH', support?
    derive Build::OnOff, 'EMBEDDED_BLUETOOTH', embedded?

    return unless support?

    validate :profiles
 end
end
```

```
IO::BT::Profile.new('bpp') do
  requires PDL::xhtml
  requires Service::io.bt.bpp
end

IO::BT::Profile.new('hcrp') do
  requires PDL::pcl
  requires Service::io.bt.hcrp
end

IO::BT::Profile.new('spp') do
  requires PDL::pcl, :optional
  requires Service::io.bt.spp
end

Service.new('io.bt.bpp') do
  build_objmodules %w{ obex_svr }
  requires Subsystem::io.bt
end

…

Subsystem.new('io.bt') do
  build_objmodules %w{ bt_mgr service_lib bt_stack }
  configures Subsystem::io.usbhost

  def process(ctx)
    configures Subsystem::io.usbhost do |usbhost|
      usbhost.drv_bt = true
    end
  end
end
```

26 10/28/2009 Hewlett-Packard

hit **PRINT**



SPLC 2009

By evaluating the **Owen** model we get…

- A generated makefile ☺
- No project configuration (other than HW component selection)

… but we obviously have a long way to go

- Settle on the right modeling language & tools
- Model the whole system
- Phase it in

```
# Service::io.bt.bpp
OBJMODULES += obex_svr
# Service::io.bt.hcrp
OBJMODULES += io_drv_hcrp
# Service::io.bt.spp
OBJMODULES += io_drv_spp
# Service::lang.pcl
OBJMODULES += pcl
OBJMODULES += jm_pcl
# Subsystem::io.bt
OBJMODULES += bt_mgr
OBJMODULES += esi2_0
OBJMODULES += services_lib
# Subsystem::io.usbhost
USBHC_UPCOM = FEATURE_OFF
USBHC_CDR = FEATURE_OFF
USBHC_BLUETOOTH = FEATURE_ON
…
```

27 10/28/2009 Hewlett-Packard

hit **PRINT**

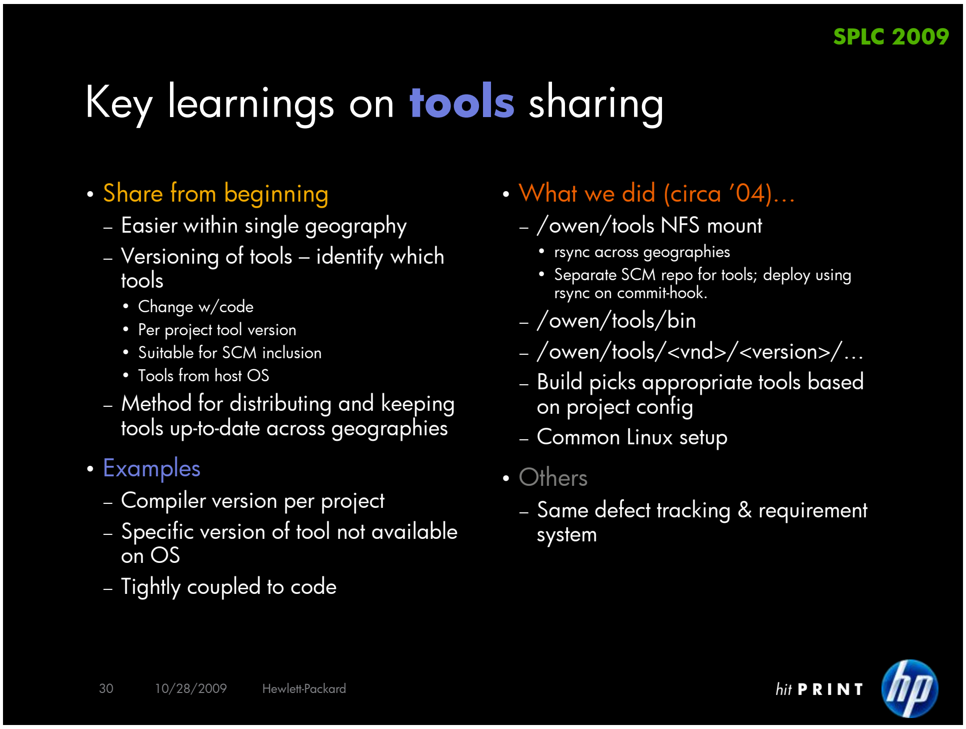
hit **PRINT**

SPLC 2009

backup slides

29 10/28/2009 Hewlett-Packard

hit PRINT



SPLC 2009

Key learnings on **tools** sharing

- Share from beginning
  - Easier within single geography
  - Versioning of tools – identify which tools
    - Change w/code
    - Per project tool version
    - Suitable for SCM inclusion
    - Tools from host OS
  - Method for distributing and keeping tools up-to-date across geographies
- Examples
  - Compiler version per project
  - Specific version of tool not available on OS
  - Tightly coupled to code

- What we did (circa '04)…
  - /owen/tools NFS mount
    - rsync across geographies
    - Separate SCM repo for tools; deploy using rsync on commit-hook.
  - /owen/tools/bin
  - /owen/tools/<vnd>/<version>/…
  - Build picks appropriate tools based on project config
  - Common Linux setup
- Others
  - Same defect tracking & requirement system

30 10/28/2009 Hewlett-Packard

hit PRINT

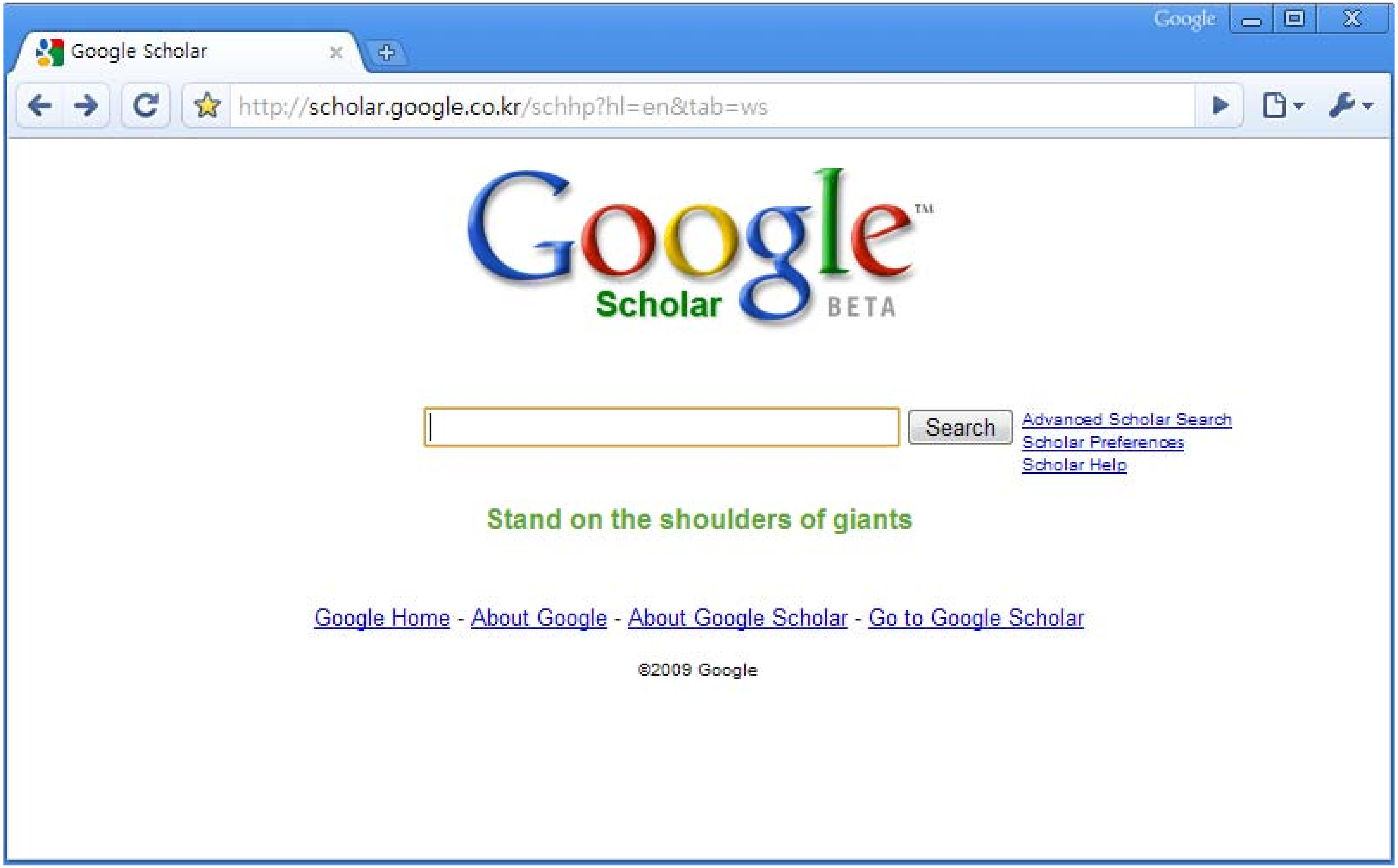
FODA: Twenty Years of Perspective on Feature Models

Kyo Chul Kang

SPLC 2009
August 24-28, 2009
San Francisco, CA, USA

Pohang University of Science and Technology (POSTECH)

Prologue

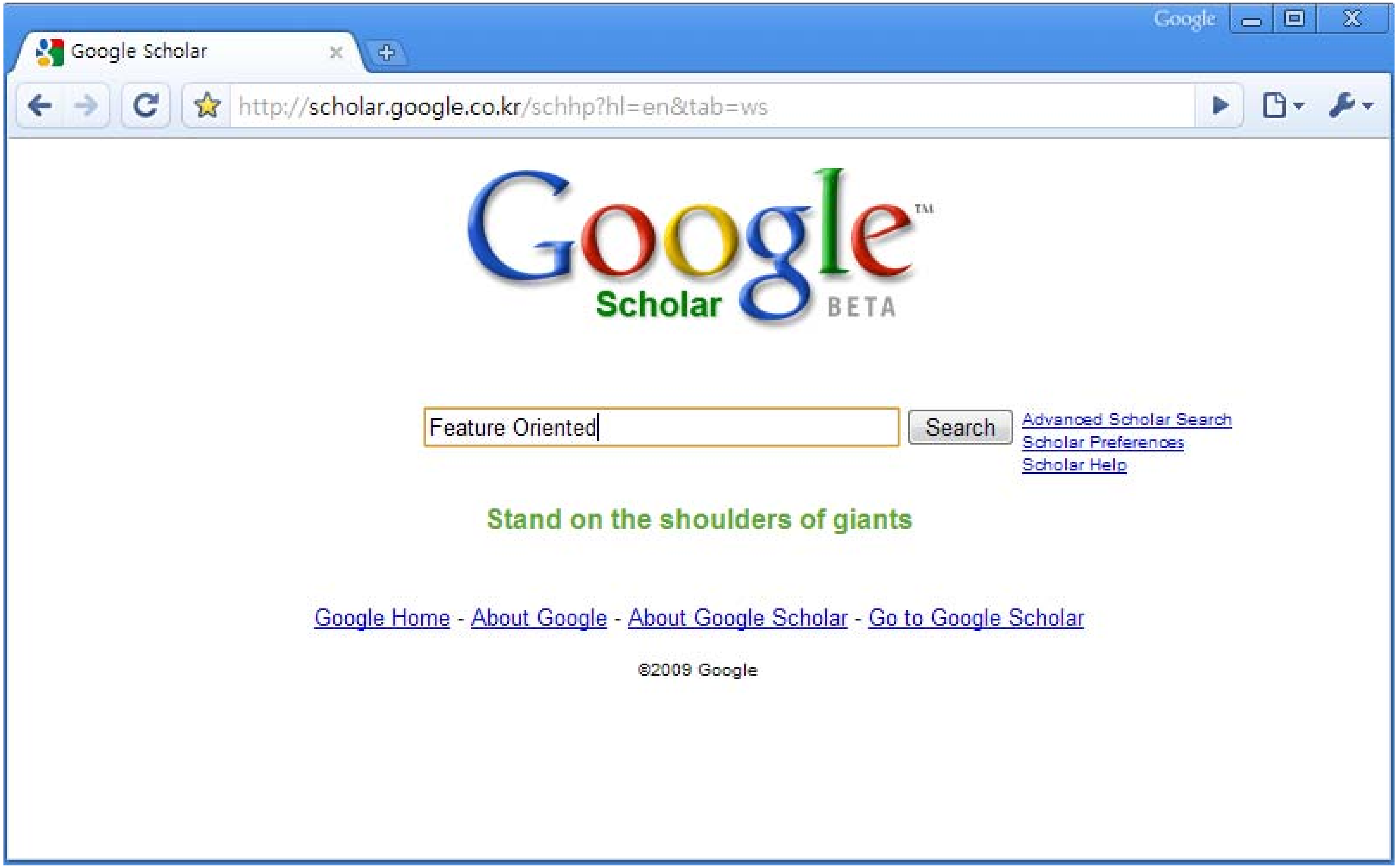


Stand on the shoulders of giants

Google Home - About Google - About Google Scholar - Go to Google Scholar

©2009 Google

Prologue

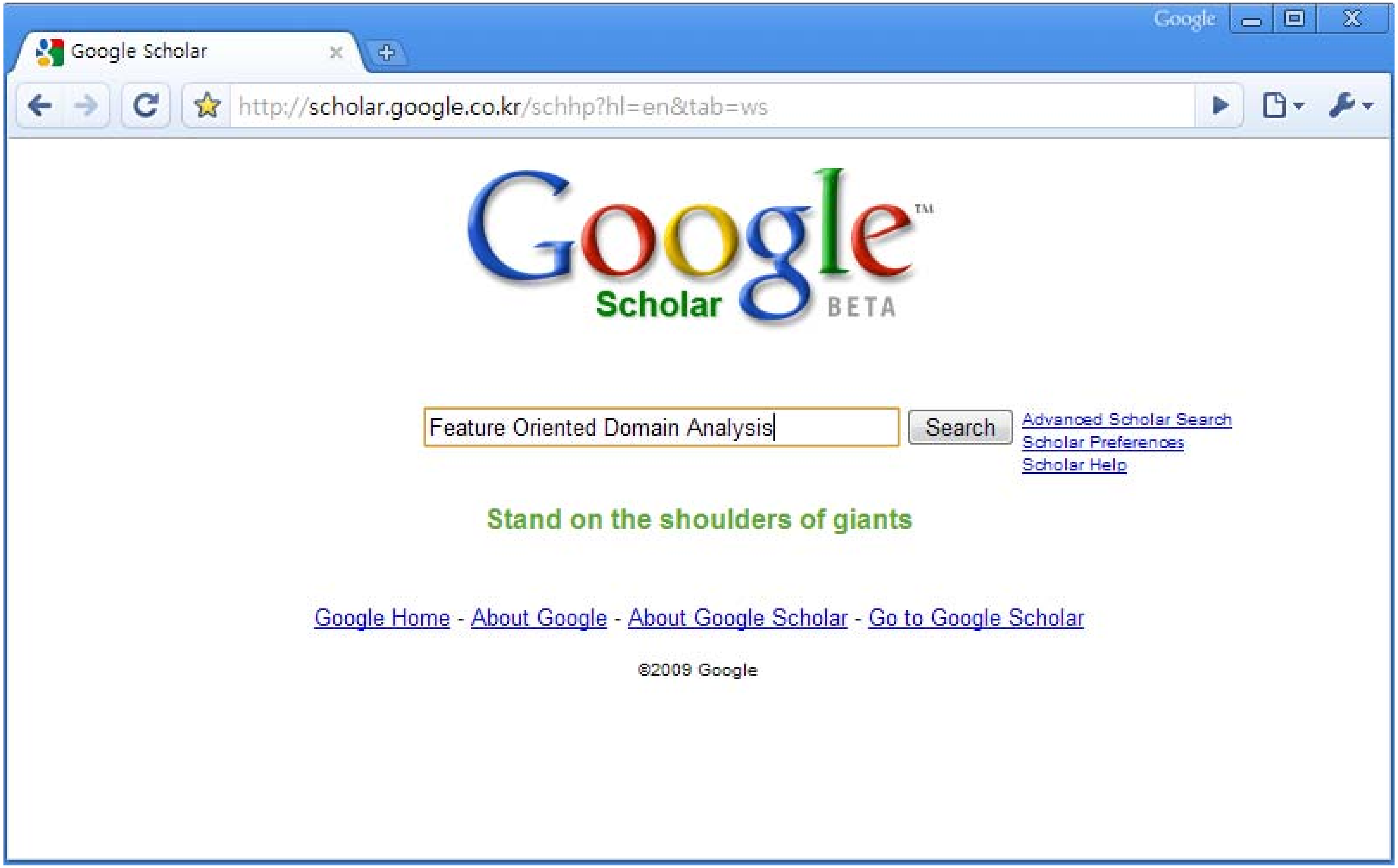

Google Scholar
http://scholar.google.co.kr/schhp?hl=en&tab=ws
Google Scholar BETA
Search
Advanced Scholar Search
Scholar Preferences
Scholar Help
Google Home - About Google - About Google Scholar - Go to Google Scholar

Feature Oriented

Stand on the shoulders of giants

©2009 Google

Prologue



Google Scholar
http://scholar.google.co.kr/schhp?hl=en&tab=ws
Google
Scholar
BETA

Feature Oriented Domain Analysis

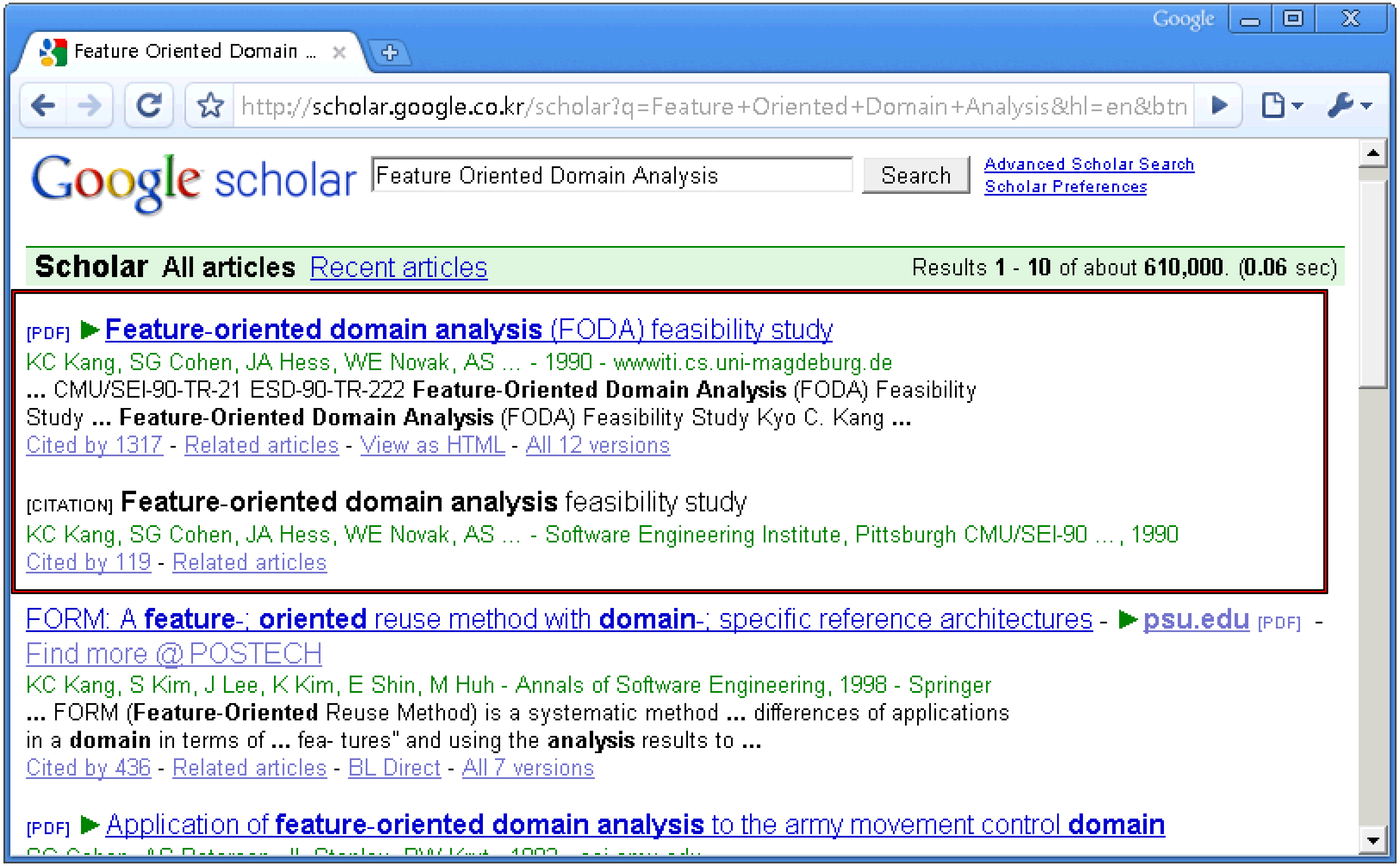
Search
Advanced Scholar Search
Scholar Preferences
Scholar Help

Stand on the shoulders of giants

Google Home - About Google - About Google Scholar - Go to Google Scholar

©2009 Google

Prologue



http://scholar.google.co.kr/scholar?q=Feature+Oriented+Domain+Analysis&hl=en&btn

Feature Oriented Domain Analysis

Scholar **All articles** Recent articles — Results **1** - **10** of about **610,000**. (**0.06** sec)

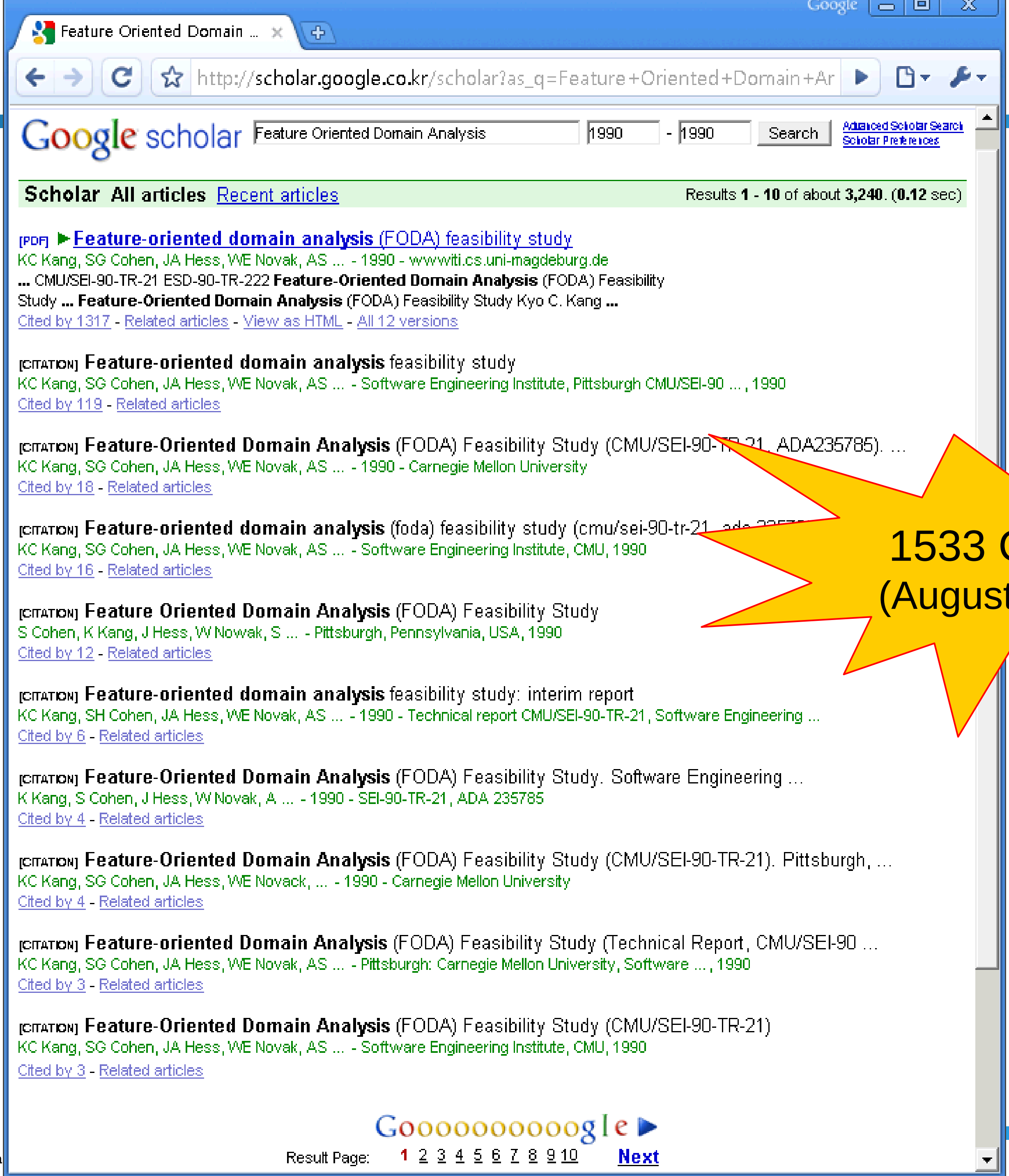
[PDF] ▶ **Feature-oriented domain analysis** (FODA) feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - 1990 - wwwiti.cs.uni-magdeburg.de
... CMU/SEI-90-TR-21 ESD-90-TR-222 **Feature-Oriented Domain Analysis** (FODA) Feasibility Study **... Feature-Oriented Domain Analysis** (FODA) Feasibility Study Kyo C. Kang **...**
Cited by 1317 - Related articles - View as HTML - All 12 versions

[CITATION] **Feature-oriented domain analysis** feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - Software Engineering Institute, Pittsburgh CMU/SEI-90 ..., 1990
Cited by 119 - Related articles

FORM: A **feature**-; **oriented** reuse method with **domain**-; specific reference architectures - ▶ **psu.edu** [PDF] - Find more @ POSTECH
KC Kang, S Kim, J Lee, K Kim, E Shin, M Huh - Annals of Software Engineering, 1998 - Springer
... FORM (**Feature-Oriented** Reuse Method) is a systematic method **...** differences of applications in a **domain** in terms of **...** fea- tures" and using the **analysis** results to **...**
Cited by 436 - Related articles - BL Direct - All 7 versions

[PDF] ▶ Application of **feature-oriented domain analysis** to the army movement control **domain**
SG Cohen, AS Peterson, JL Stanley, RW Krut - 1992 - sei.cmu.edu

Prologue



Feature Oriented Domain ...

http://scholar.google.co.kr/scholar?as_q=Feature+Oriented+Domain+Ar

Google scholar Feature Oriented Domain Analysis 1990 - 1990 Search Advanced Scholar Search Scholar Preferences

Scholar **All articles** Recent articles Results **1** - **10** of about **3,240**. (**0.12** sec)

[PDF] ▶ **Feature-oriented domain analysis** (FODA) feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - 1990 - wwwiti.cs.uni-magdeburg.de
... CMU/SEI-90-TR-21 ESD-90-TR-222 **Feature-Oriented Domain Analysis** (FODA) Feasibility Study **... Feature-Oriented Domain Analysis** (FODA) Feasibility Study Kyo C. Kang **...**
Cited by 1317 - Related articles - View as HTML - All 12 versions

[CITATION] **Feature-oriented domain analysis** feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - Software Engineering Institute, Pittsburgh CMU/SEI-90 ... , 1990
Cited by 119 - Related articles

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study (CMU/SEI-90-TR-21, ADA235785). ...
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - 1990 - Carnegie Mellon University
Cited by 18 - Related articles

[CITATION] **Feature-oriented domain analysis** (foda) feasibility study (cmu/sei-90-tr-21, ada 23578
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - Software Engineering Institute, CMU, 1990
Cited by 16 - Related articles

[CITATION] **Feature Oriented Domain Analysis** (FODA) Feasibility Study
S Cohen, K Kang, J Hess, W Nowak, S ... - Pittsburgh, Pennsylvania, USA, 1990
Cited by 12 - Related articles

[CITATION] **Feature-oriented domain analysis** feasibility study: interim report
KC Kang, SH Cohen, JA Hess, WE Novak, AS ... - 1990 - Technical report CMU/SEI-90-TR-21, Software Engineering ...
Cited by 6 - Related articles

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study. Software Engineering ...
K Kang, S Cohen, J Hess, W Novak, A ... - 1990 - SEI-90-TR-21, ADA 235785
Cited by 4 - Related articles

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study (CMU/SEI-90-TR-21). Pittsburgh, ...
KC Kang, SG Cohen, JA Hess, WE Novack, ... - 1990 - Carnegie Mellon University
Cited by 4 - Related articles

[CITATION] **Feature-oriented Domain Analysis** (FODA) Feasibility Study (Technical Report, CMU/SEI-90 ...
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - Pittsburgh: Carnegie Mellon University, Software ... , 1990
Cited by 3 - Related articles

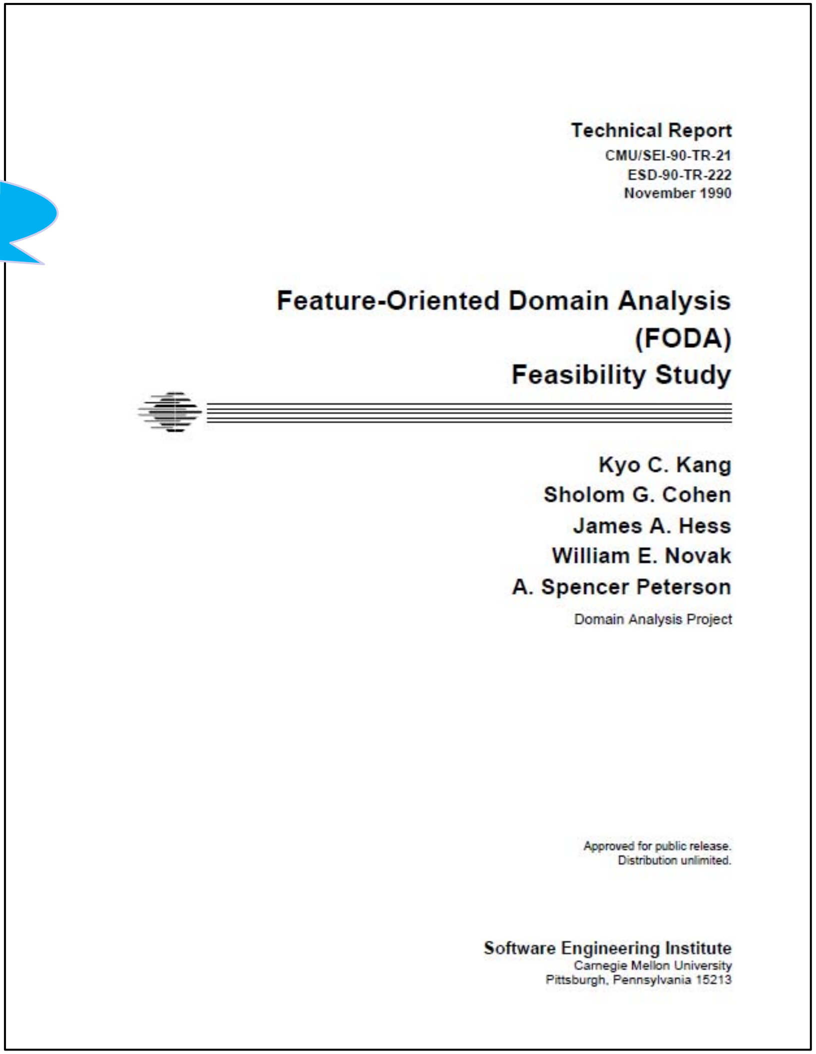
[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study (CMU/SEI-90-TR-21)
KC Kang, SG Cohen, JA Hess, WE Novak, AS ... - Software Engineering Institute, CMU, 1990
Cited by 3 - Related articles

Gooooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

1533 Citations!
(August 12, 2009)

Prologue

We are talking about FODA



Technical Report
CMU/SEI-90-TR-21
ESD-90-TR-222
November 1990

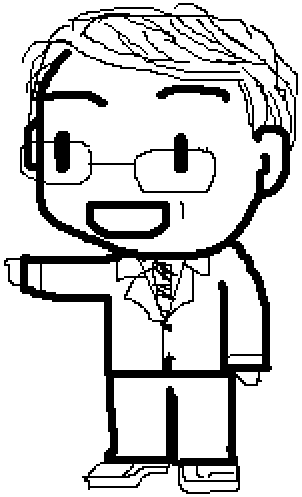
Feature-Oriented Domain Analysis (FODA) Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson

Domain Analysis Project

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213



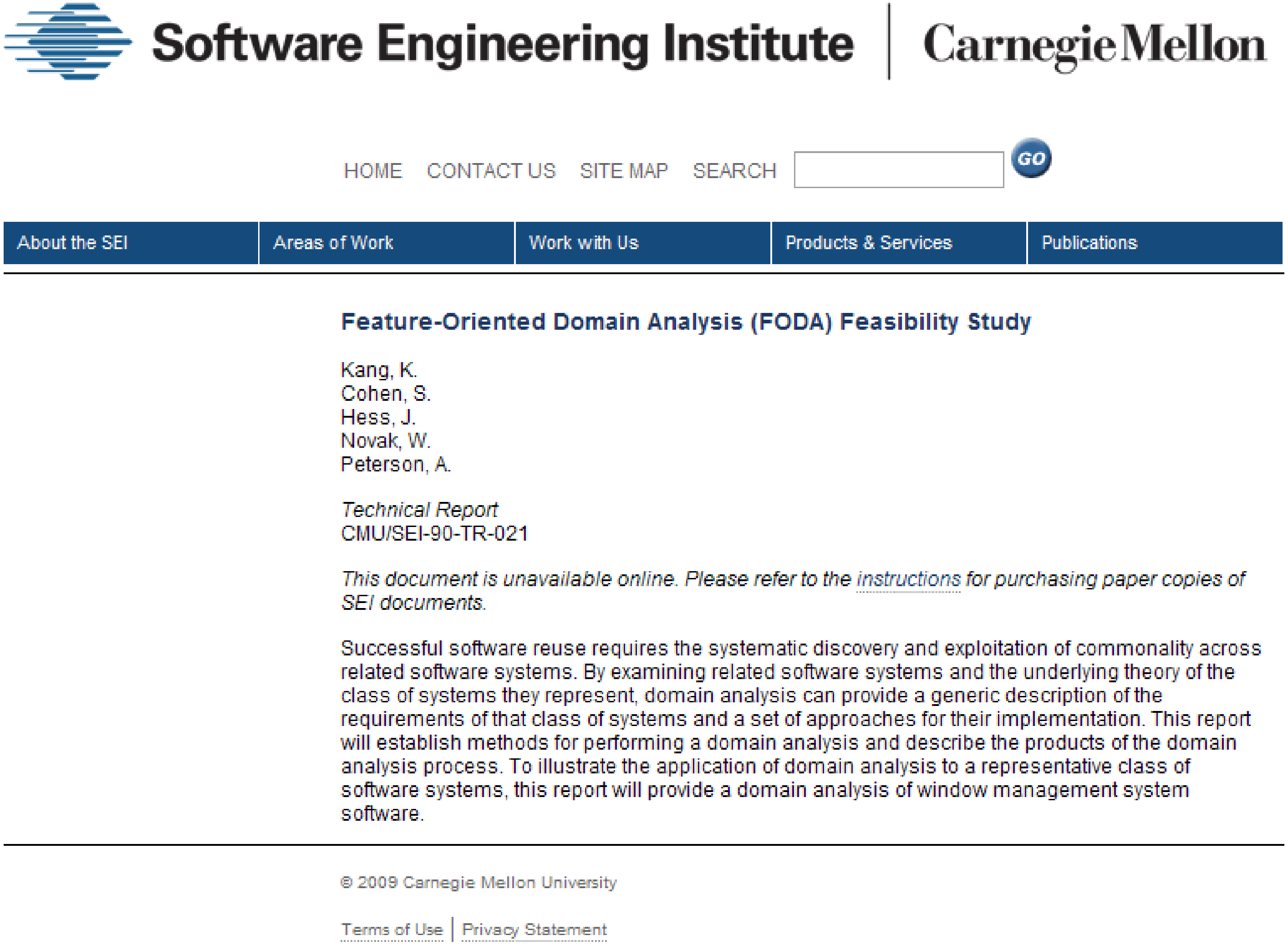
Agenda

- Introduction
- Looking Back
  - Number of Citations
  - Feature Model Genealogy
  - Why Popular?
  - Salient Features of FODA Report
- Looking Forward
  - Future Works
  - Other Issues
- Acknowledgement

Agenda

- Introduction
- Looking Back
  - Number of Citations
  - Feature Model Genealogy
  - Why Popular?
  - Salient Features of FODA Report
- Looking Forward
  - Future works
  - Other Issues
- Acknowledgement

Introduction



Feature-Oriented Domain Analysis (FODA) Feasibility Study

Kang, K.
Cohen, S.
Hess, J.
Novak, W.
Peterson, A.

Technical Report
CMU/SEI-90-TR-021

This document is unavailable online. Please refer to the instructions for purchasing paper copies of SEI documents.

Successful software reuse requires the systematic discovery and exploitation of commonality across related software systems. By examining related software systems and the underlying theory of the class of systems they represent, domain analysis can provide a generic description of the requirements of that class of systems and a set of approaches for their implementation. This report will establish methods for performing a domain analysis and describe the products of the domain analysis process. To illustrate the application of domain analysis to a representative class of software systems, this report will provide a domain analysis of window management system software.

People

Sholom G. Cohen

William E. Novak

A. Spencer Peterson

James A. Hess

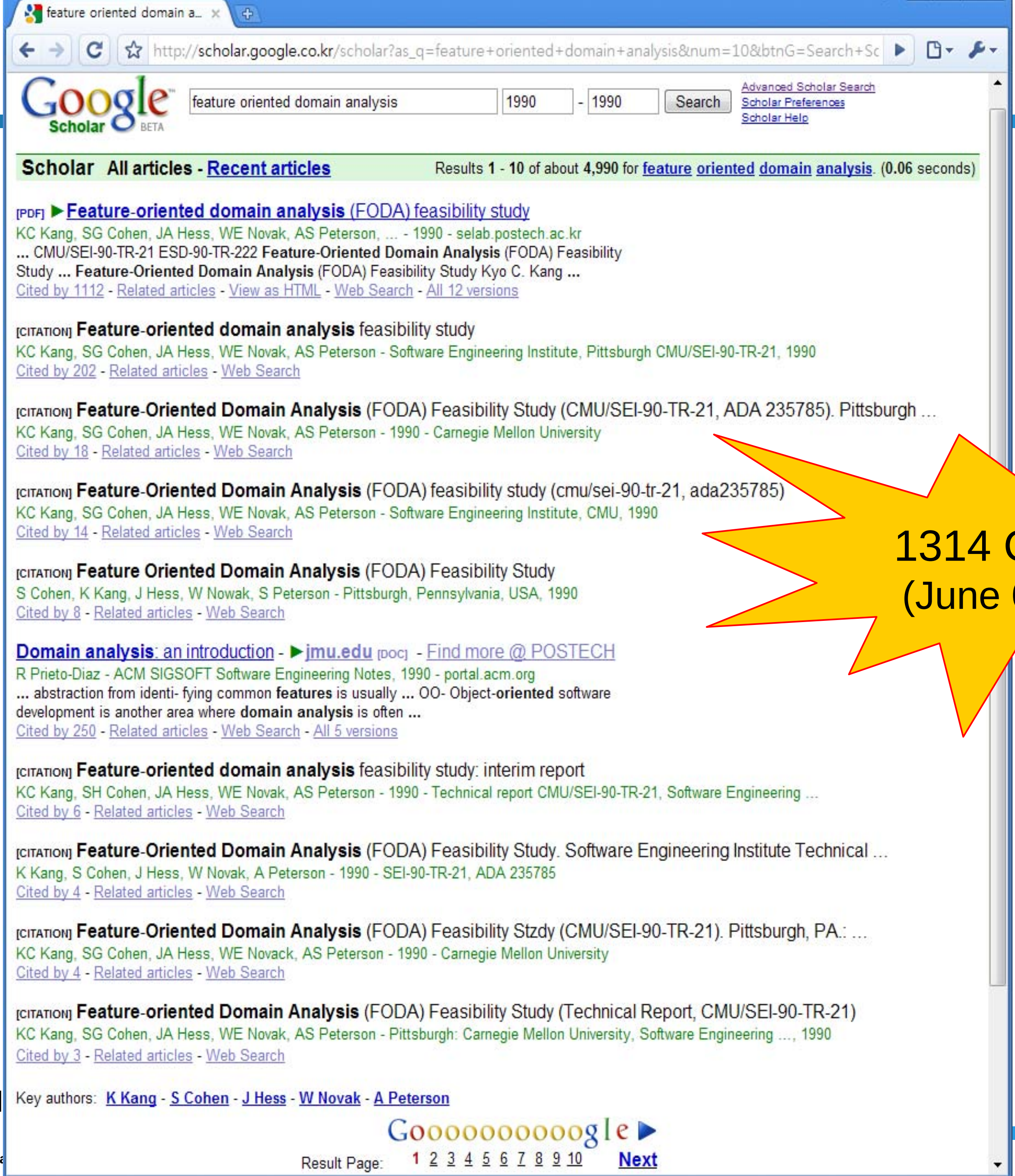
Kyo C. Kang

August 2009

Agenda

- Introduction
- Looking Back
  - Number of Citations
  - Feature Model Genealogy
  - Why Popular?
  - Salient Features of FODA Report
- Looking Forward
  - Future Works
  - Other Issues
- Acknowledgement

Prologue



feature oriented domain a...
http://scholar.google.co.kr/scholar?as_q=feature+oriented+domain+analysis&num=10&btnG=Search+Sc
Google Scholar BETA
feature oriented domain analysis
1990 - 1990
Search
Advanced Scholar Search
Scholar Preferences
Scholar Help

Scholar All articles - Recent articles   Results **1** - **10** of about **4,990** for feature oriented domain analysis. (**0.06** seconds)

[PDF] ▶ **Feature-oriented domain analysis** (FODA) feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson, ... - 1990 - selab.postech.ac.kr
... CMU/SEI-90-TR-21 ESD-90-TR-222 **Feature-Oriented Domain Analysis** (FODA) Feasibility Study ... **Feature-Oriented Domain Analysis** (FODA) Feasibility Study Kyo C. Kang ...
Cited by 1112 - Related articles - View as HTML - Web Search - All 12 versions

[CITATION] **Feature-oriented domain analysis** feasibility study
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson - Software Engineering Institute, Pittsburgh CMU/SEI-90-TR-21, 1990
Cited by 202 - Related articles - Web Search

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study (CMU/SEI-90-TR-21, ADA 235785). Pittsburgh ...
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson - 1990 - Carnegie Mellon University
Cited by 18 - Related articles - Web Search

[CITATION] **Feature-Oriented Domain Analysis** (FODA) feasibility study (cmu/sei-90-tr-21, ada235785)
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson - Software Engineering Institute, CMU, 1990
Cited by 14 - Related articles - Web Search

[CITATION] **Feature Oriented Domain Analysis** (FODA) Feasibility Study
S Cohen, K Kang, J Hess, W Nowak, S Peterson - Pittsburgh, Pennsylvania, USA, 1990
Cited by 8 - Related articles - Web Search

Domain analysis: an introduction - ▶ **jmu.edu** [DOC] - Find more @ POSTECH
R Prieto-Diaz - ACM SIGSOFT Software Engineering Notes, 1990 - portal.acm.org
... abstraction from identi- fying common **features** is usually ... OO- Object-**oriented** software development is another area where **domain analysis** is often ...
Cited by 250 - Related articles - Web Search - All 5 versions

[CITATION] **Feature-oriented domain analysis** feasibility study: interim report
KC Kang, SH Cohen, JA Hess, WE Novak, AS Peterson - 1990 - Technical report CMU/SEI-90-TR-21, Software Engineering ...
Cited by 6 - Related articles - Web Search

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Study. Software Engineering Institute Technical ...
K Kang, S Cohen, J Hess, W Novak, A Peterson - 1990 - SEI-90-TR-21, ADA 235785
Cited by 4 - Related articles - Web Search

[CITATION] **Feature-Oriented Domain Analysis** (FODA) Feasibility Stzdy (CMU/SEI-90-TR-21). Pittsburgh, PA.: ...
KC Kang, SG Cohen, JA Hess, WE Novack, AS Peterson - 1990 - Carnegie Mellon University
Cited by 4 - Related articles - Web Search

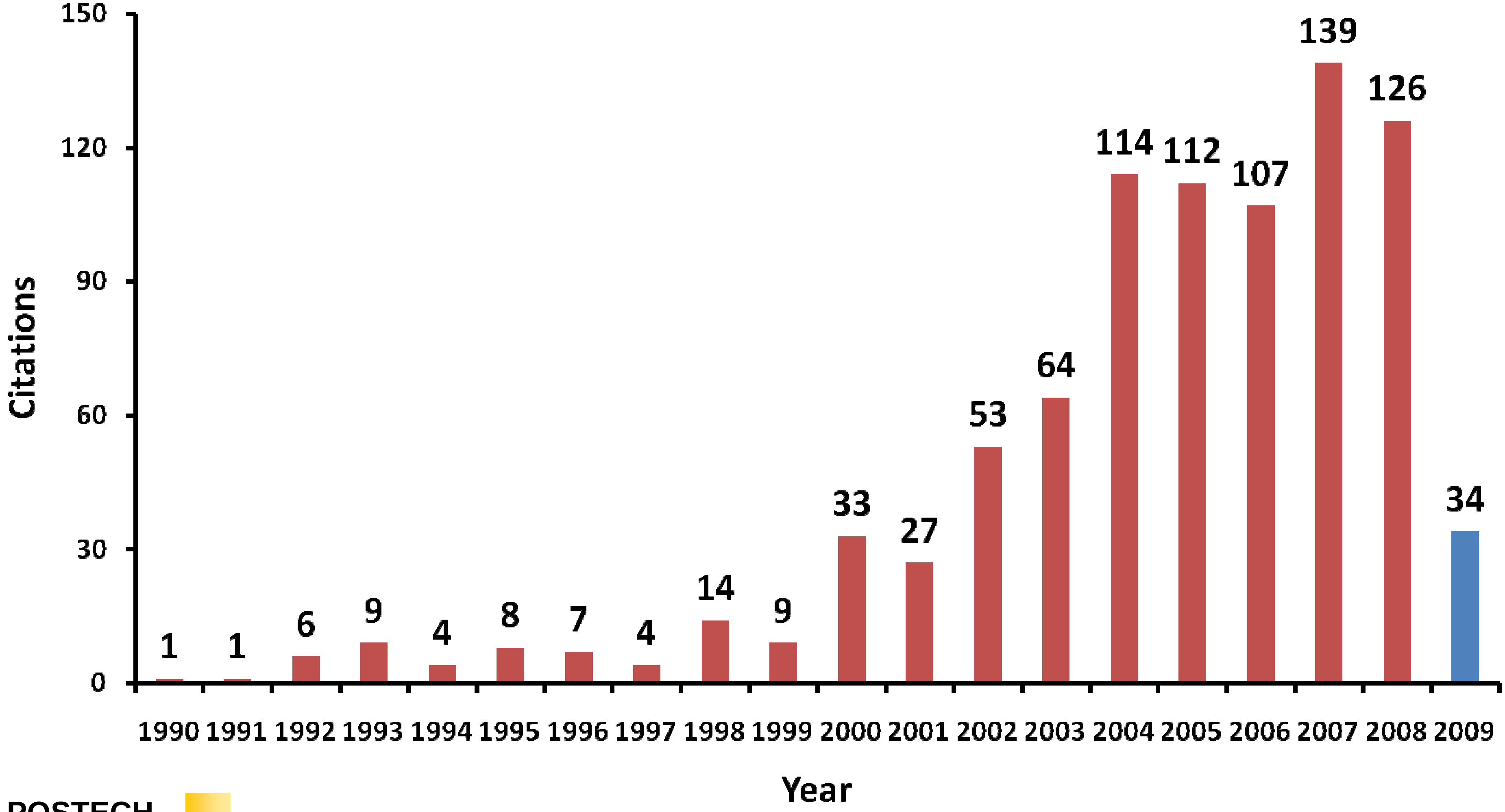
[CITATION] **Feature-oriented Domain Analysis** (FODA) Feasibility Study (Technical Report, CMU/SEI-90-TR-21)
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson - Pittsburgh: Carnegie Mellon University, Software Engineering ..., 1990
Cited by 3 - Related articles - Web Search

Key authors: K Kang - S Cohen - J Hess - W Novak - A Peterson

1314 Citations!
(June 06, 2009)

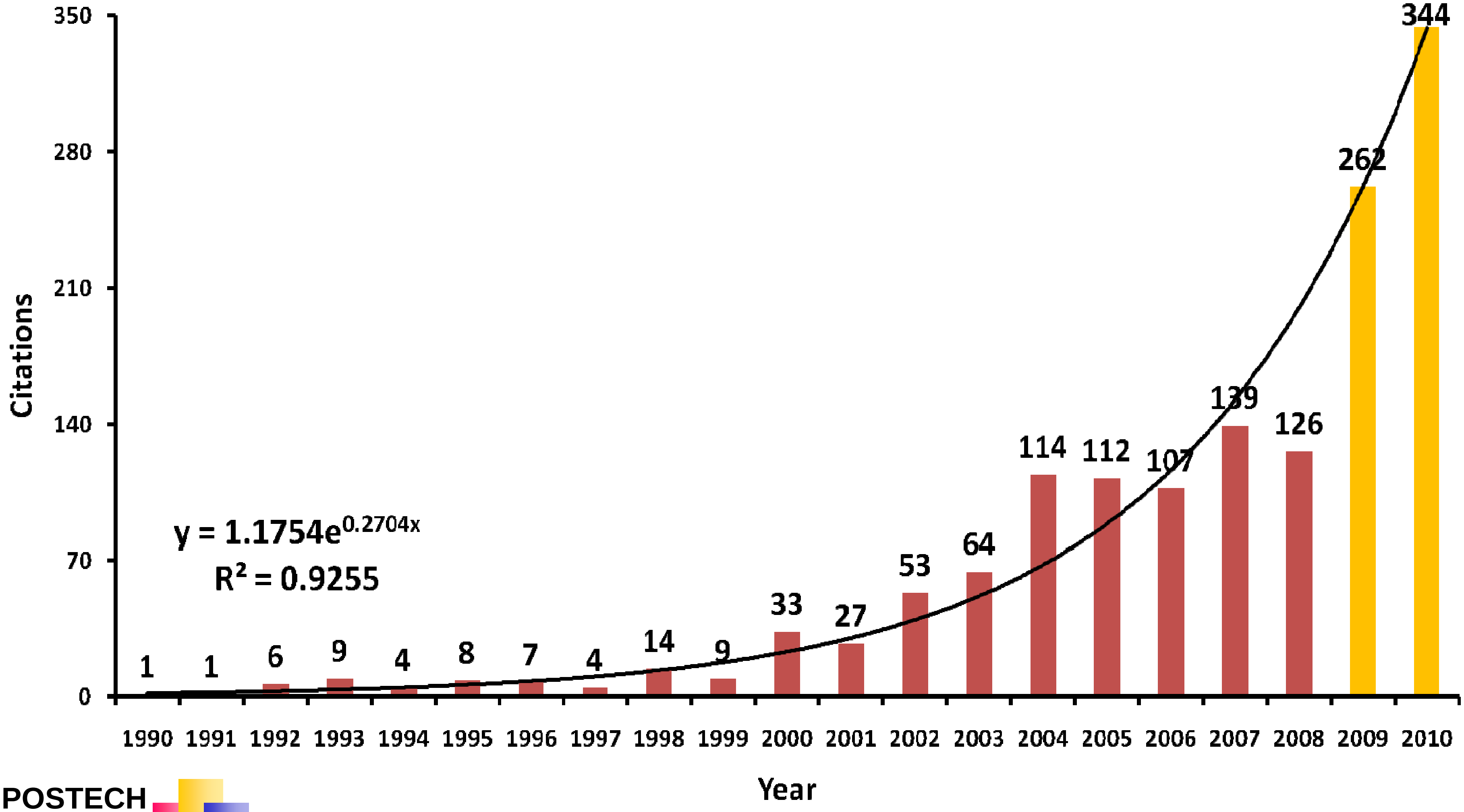
Number of Citations

1990~2009 (June 6)
Total: 1314
Unknown/Overlapping: 442



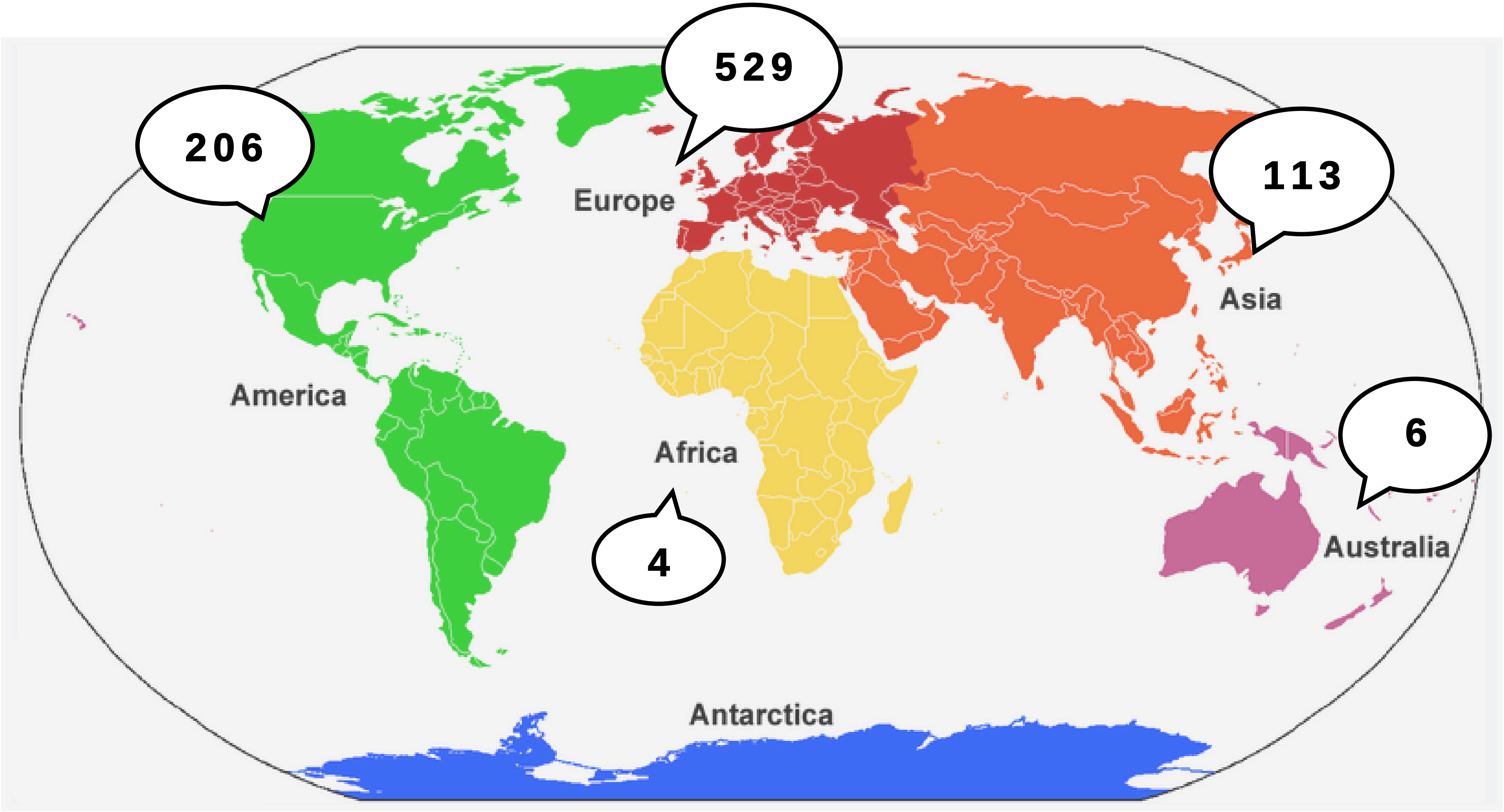
Number of Citations

1990~2008 (June 6)
Total: 1280
Unknown/Overlapping: 442



Regional Differences

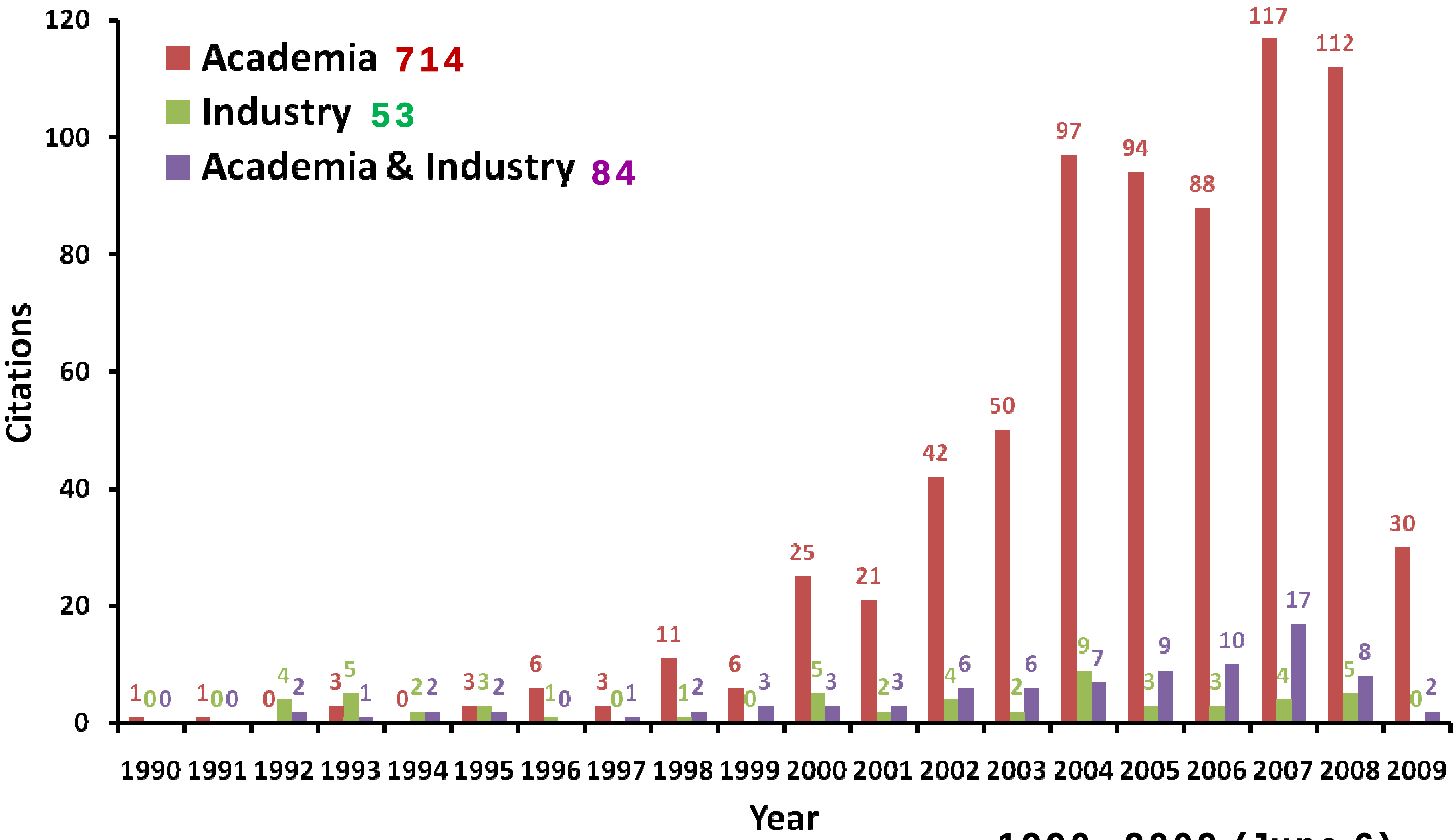
Number of Citations



1990~2009 (June 6)
Total: 1314
Unknown/Overlapping: 456

Industry vs. Academia

Number of Citations

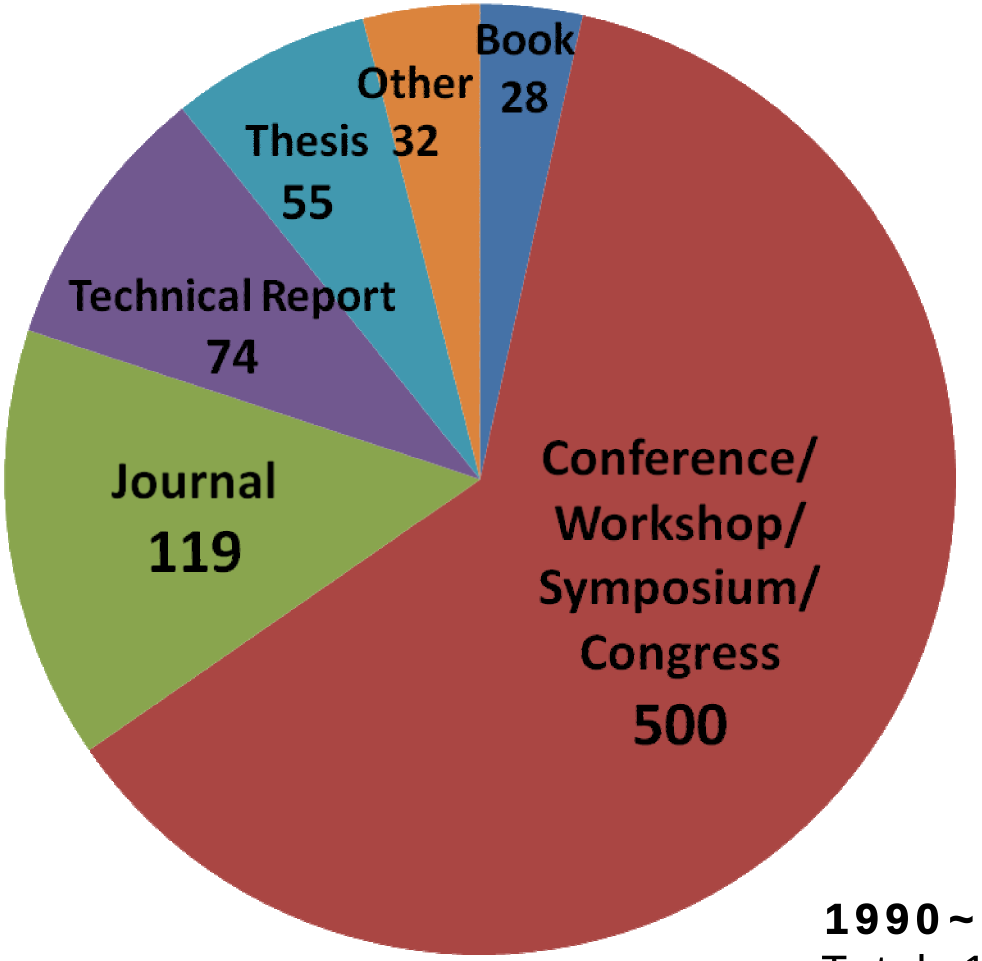


1990~2009 (June 6)
Total: 1314
Unknown/Overlapping: 467

POSTECH

Source Differences

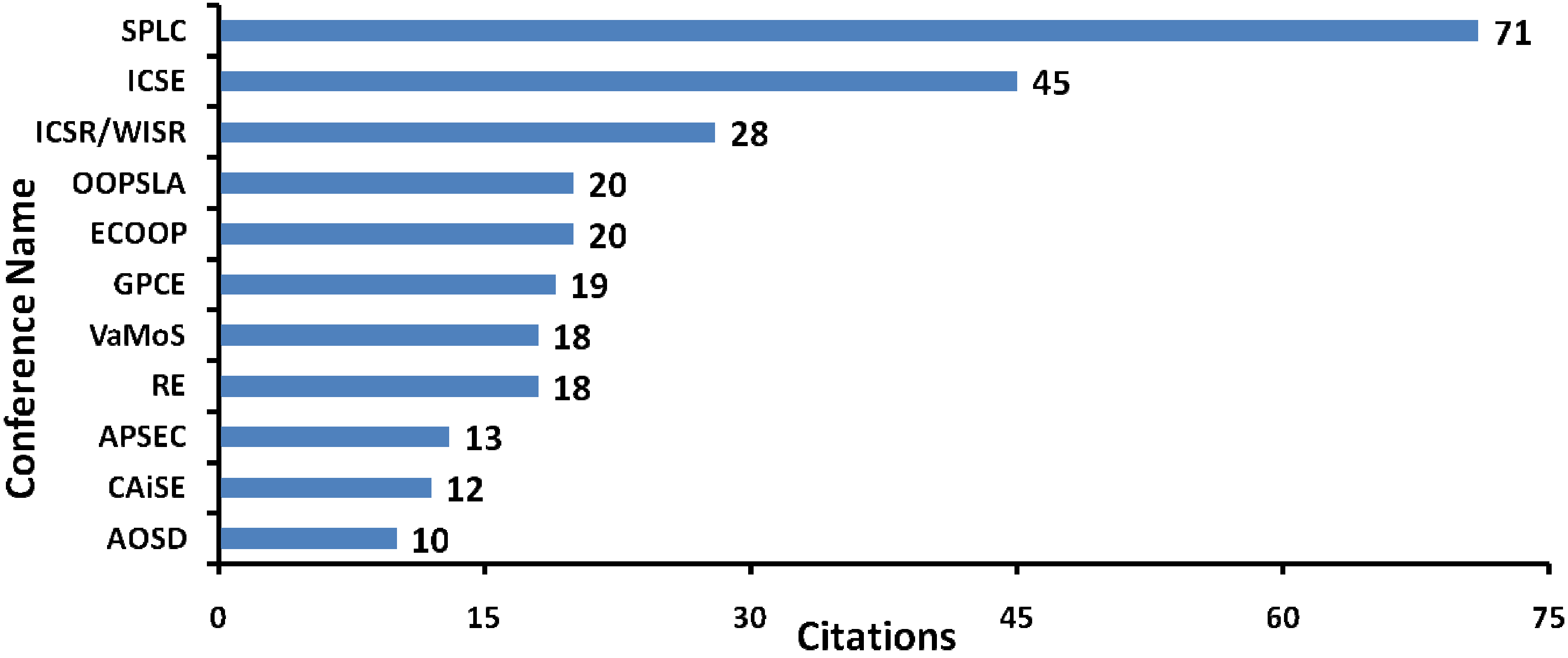
Number of Citations



1990~2009 (June 6)
Total: 1314
Unknown/Overlapping: 425
Not-English: 81

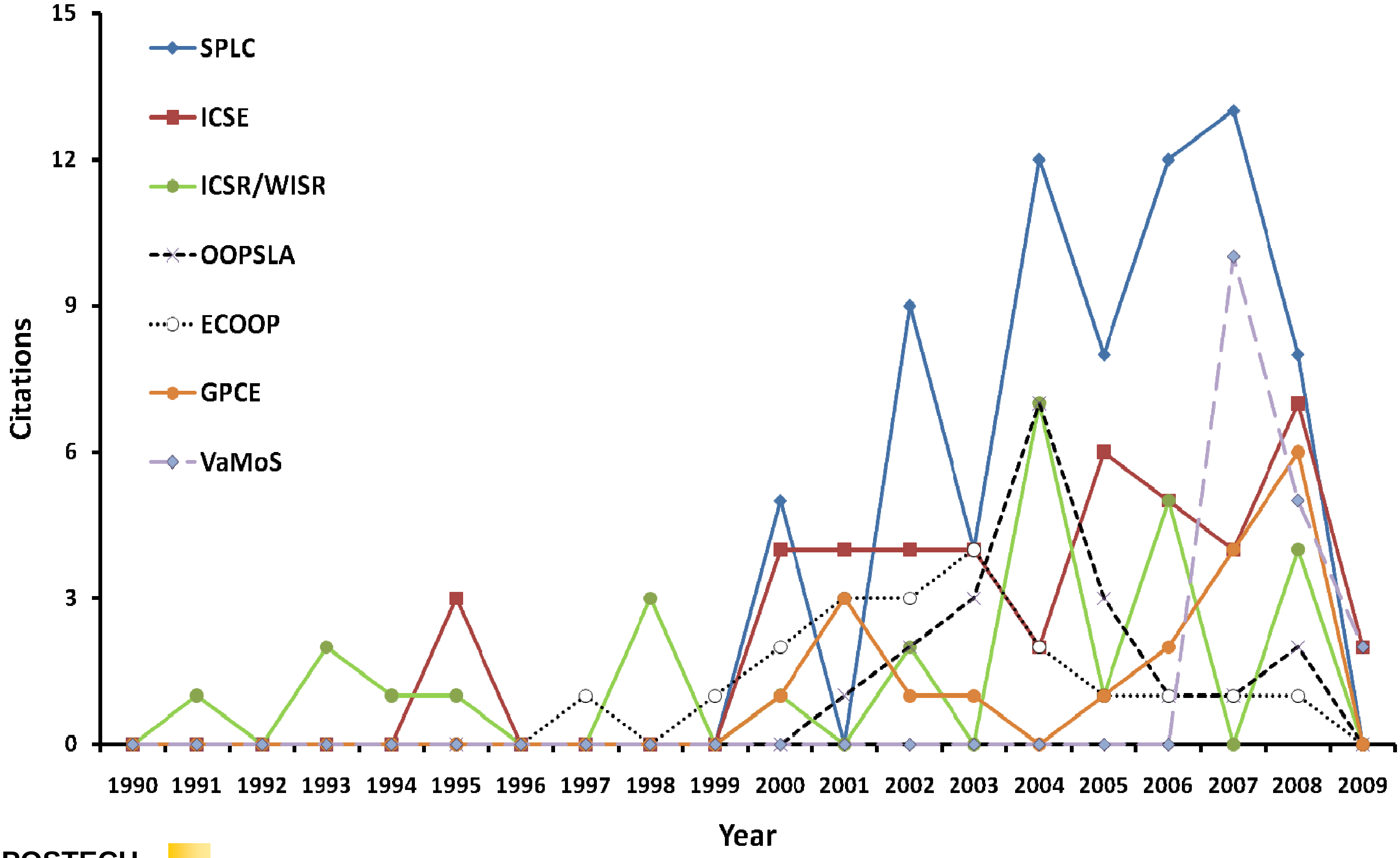
Top Conferences

Number of Citations



SPLC: International Software Product Line Conference
ICSE: International Conference on Software Engineering
ICSR/WISR: International Conference on Software Reuse/ Annual Workshops on Institutionalizing Software Reuse
OOPSLA: International Conference on Object Oriented Programming, Systems, Languages and Applications
ECOOP: European Conference on Object-Oriented Programming
GPCE: International Conference on Generative Programming and Component Engineering
VaMoS: International Workshop on Variability Modeling of Software-intensive Systems
RE: International Requirements Engineering Conference
APSEC: Asia-Pacific Software Engineering Conference
CAiSE: International Conference on Advanced Information Systems Engineering
AOSD: International Conference on Aspect-Oriented Software Development

Number of Citations



Number of Citations

Subject Categories

- Feature Modeling
- Feature Model Formalization
- Feature Model Extension

- Generative Programming
- Domain-Specific Language
- Feature-Oriented Design/Programming

- Product Configuration
- Variability Management

- Product Line Methodology /Product Line Adoption
- Doman Analysis /Requirements Analysis
- Domain-Specific Architecture
- Reusable Component Development
- Refactoring/Reengineering

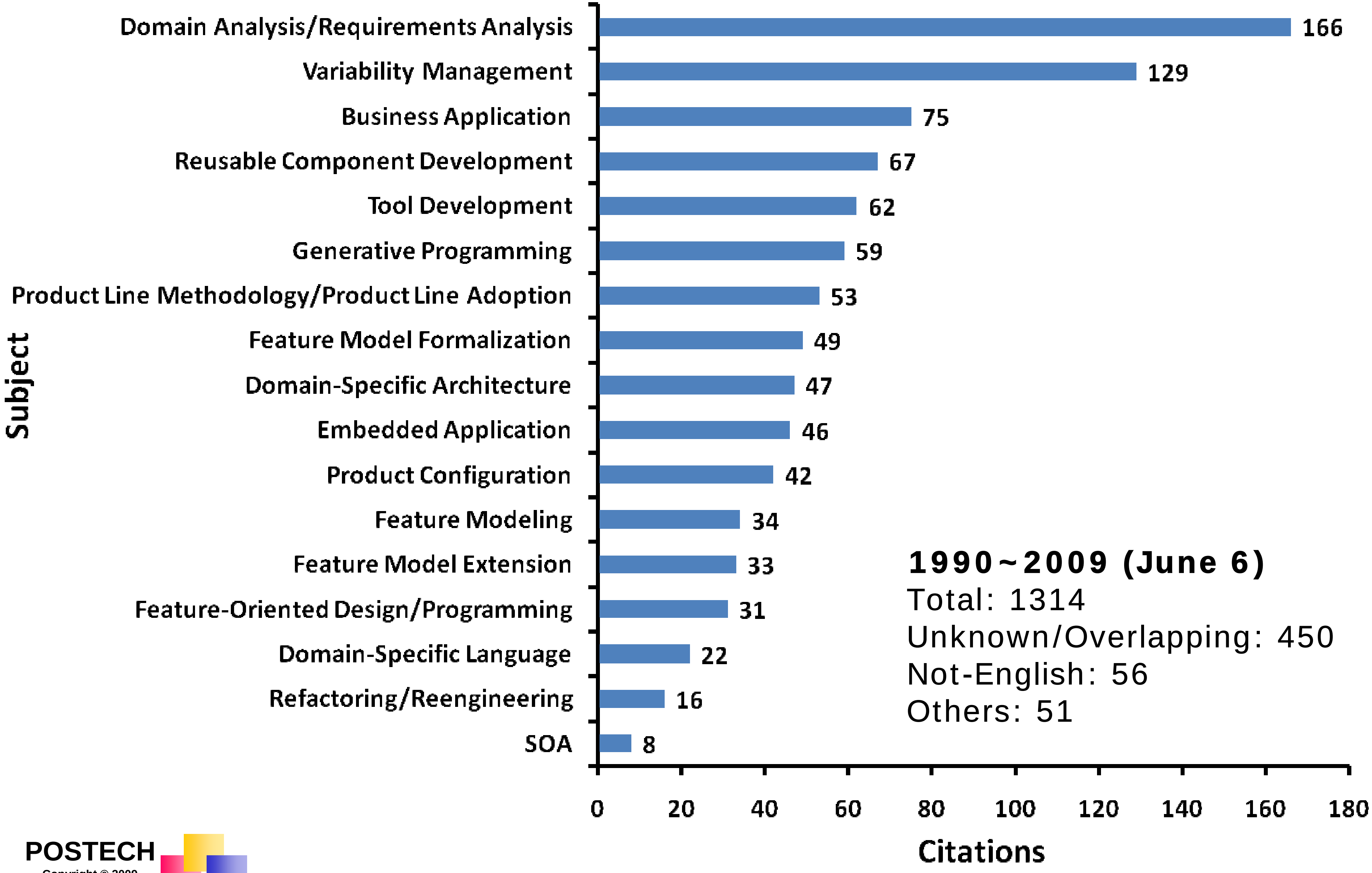
- Tool Development

- Business Application
- Embedded Application
- SOA

- Other

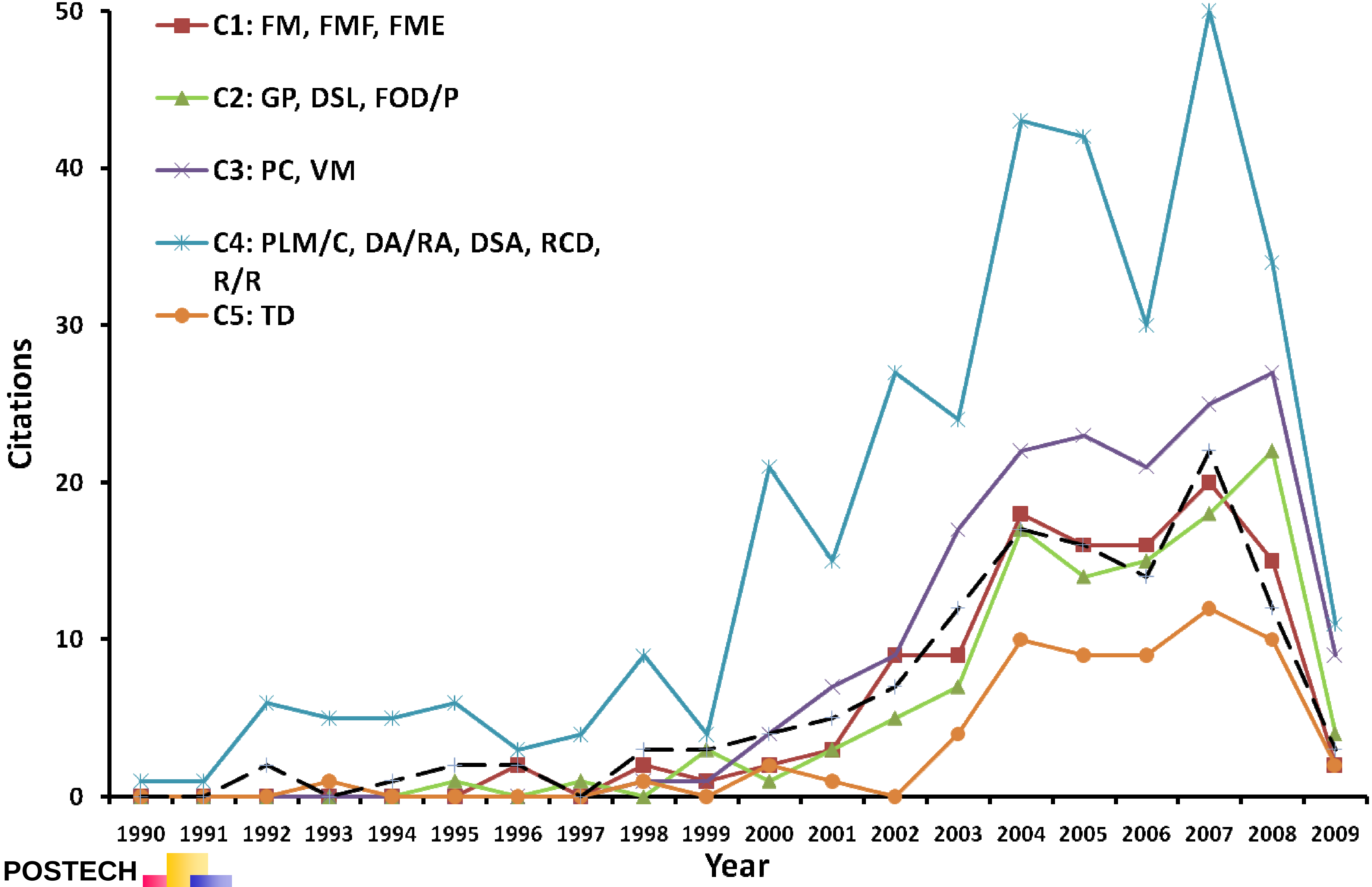
Subject Differences

Number of Citations



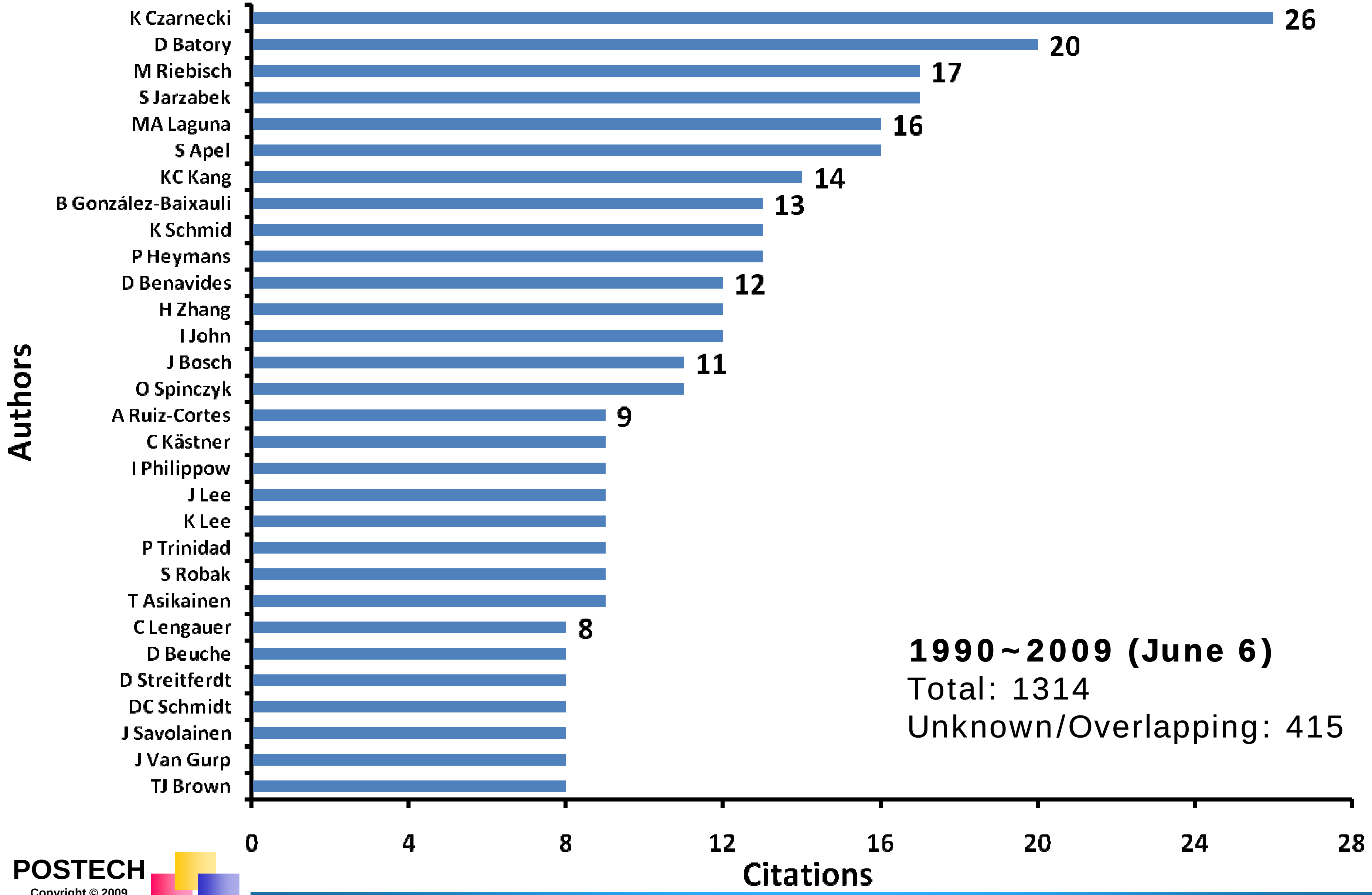
1990~2009 (June 6)
Total: 1314
Unknown/Overlapping: 450
Not-English: 56
Others: 51

Number of Citations



Top-30 People

Number of Citations



1990~2009 (June 6)

Total: 1314

Unknown/Overlapping: 415

Survey: Top-30 People

1. What are the contributions of FODA?

2. What are the remaining problems for FODA?

3. What is your own one-sentence definition/feeling about the feature model?

4. Anything else you would suggest for inclusion?

The Contributions of FODA:

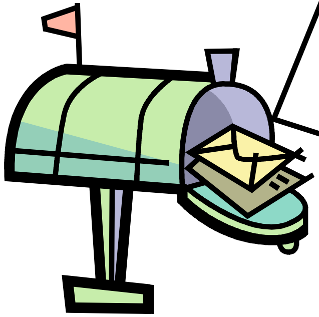
Systematic domain analysis method used in software product line engineering. (C Kastner, S Apel, C Lengauer)

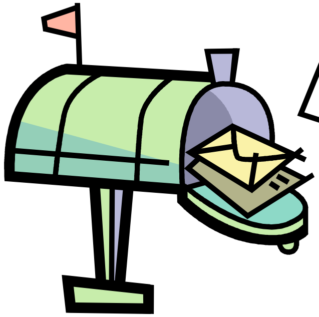
Strongly influencing how software configuration is seen in software product line, and popularizing the domain analysis concept. (K Schmid)

Laying the groundwork for feature analysis and feature modeling. (J Van Gurp)

Feature modeling :

- Essential technique for defining the space of programs that define a software product line. (D Batory)
- Simple but comprehensive way to modeling commonalities and variabilities in a domain. (H Zhang, J Lee, S Jarzabek, T Asikainen)
- Easy to use and communicate between stakeholders. (C Kastner)
- Giving a name to a fundamental form of modularity in context of product lines. (D Batory)





The Remaining Problems for FODA:

Addressing other parts of the life cycle (Especially application engineering). (K Schmid)

Clear mapping between features and software artifacts. (J Van Gurp, K Schmid)

Standardization of feature model extensions; Trade-off between expressiveness and simplicity. (C Kastner, S Apel, C Lengauer, T Asikainen, K Schmid)

Scalability of feature model. (C Kastner, D Batory)

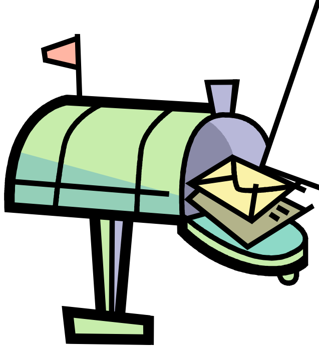
Managing Complexity in view of many inter-dependent features. (J Van Gurp, S Jarzabek)

Feature model evaluation. (J Lee)

Integration with UML Model. (S Jarzabek)

Good teaching materials. (D Batory)

Survey



Own One-Sentence Definition/Feeling about the Feature Model:

A model that provides the foundations of software reuse. (J Lee)

A simple means to describe the commonalities and variabilities of a domain / product line. (S Apel, C Lengauer)

Good and easy to understand, practical and fundamental. (H Zhang)

Easy to use (management-compatible) graphical model to describe variability. (C Kastner)

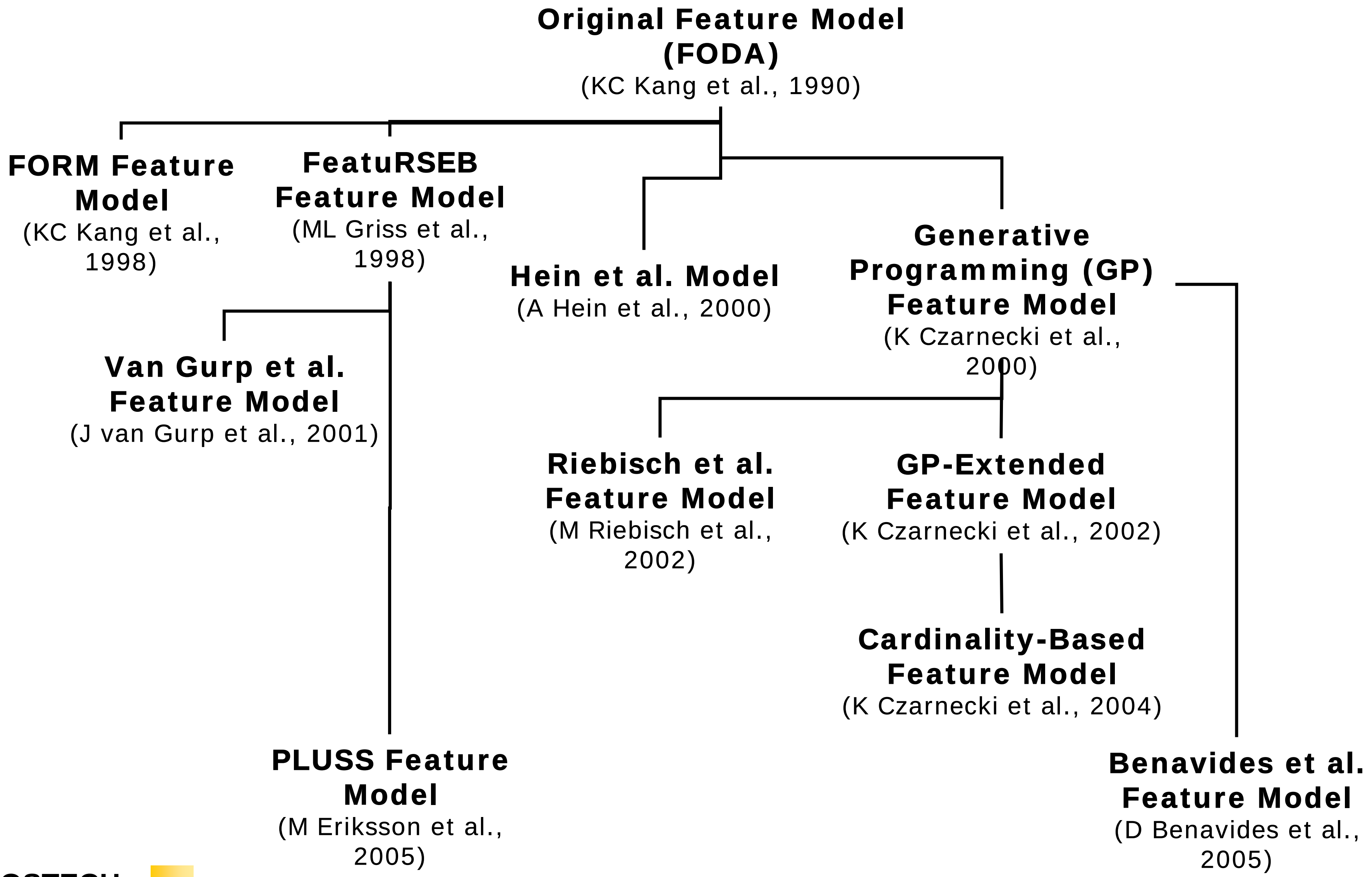
Great notation, accepted by all in the field. (S Jarzabek)

One of the useful ways to analyze requirements (required features) as well as a means to describe given software in terms of features (provided features). (J Van Gurp)

An idea well received by the community but still lacking content meeting the scientific standards. (T Asikainen)

Great approach to make one understand the core idea of product line engineering (But I'm not sure whether it is the right way of looking at variability for the actual development). (K Schmid)

Feature Model Genealogy



Original Feature Model (FODA)
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

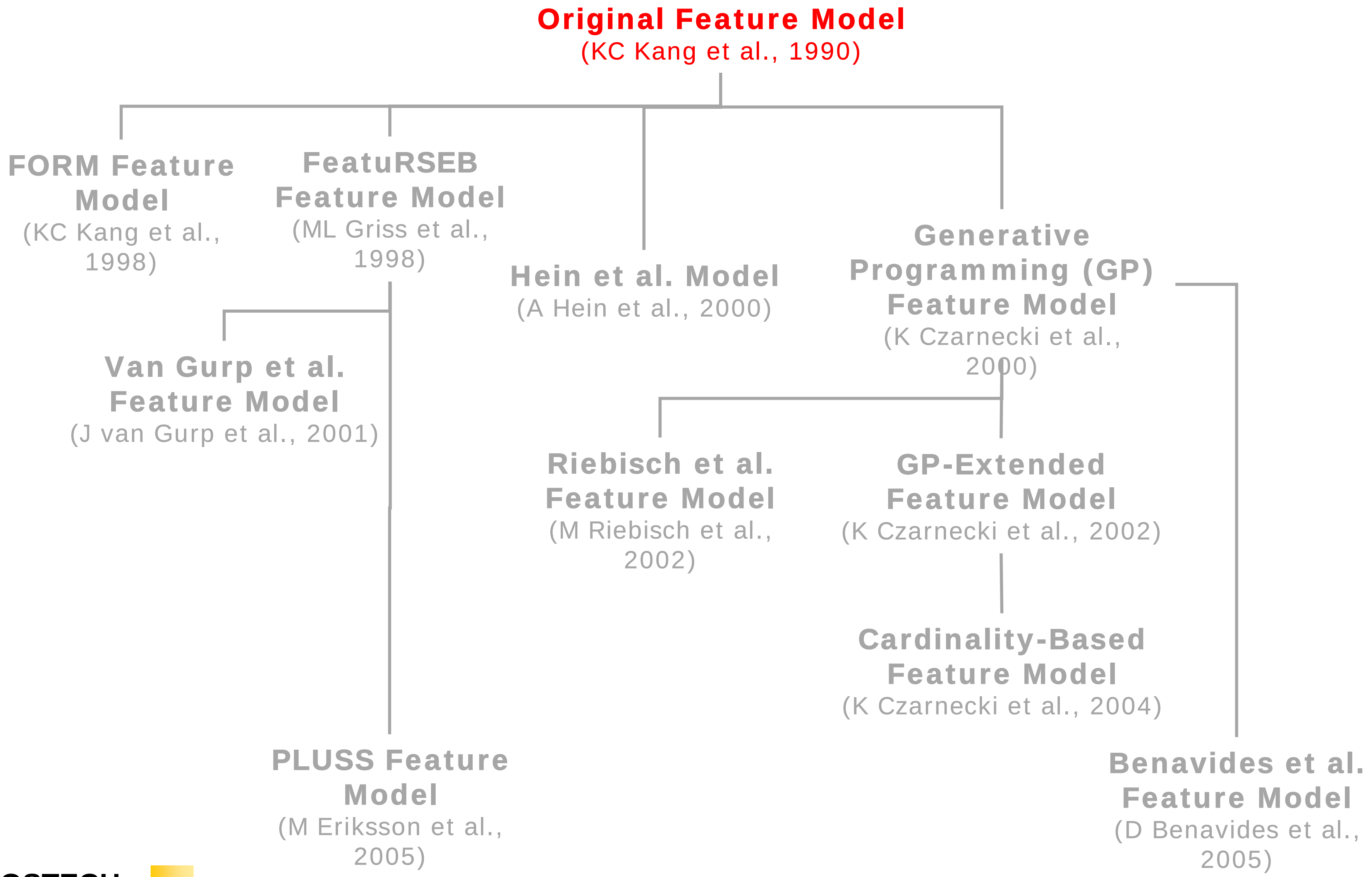
GP-Extended Feature Model
(K Czarnecki et al., 2002)

Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

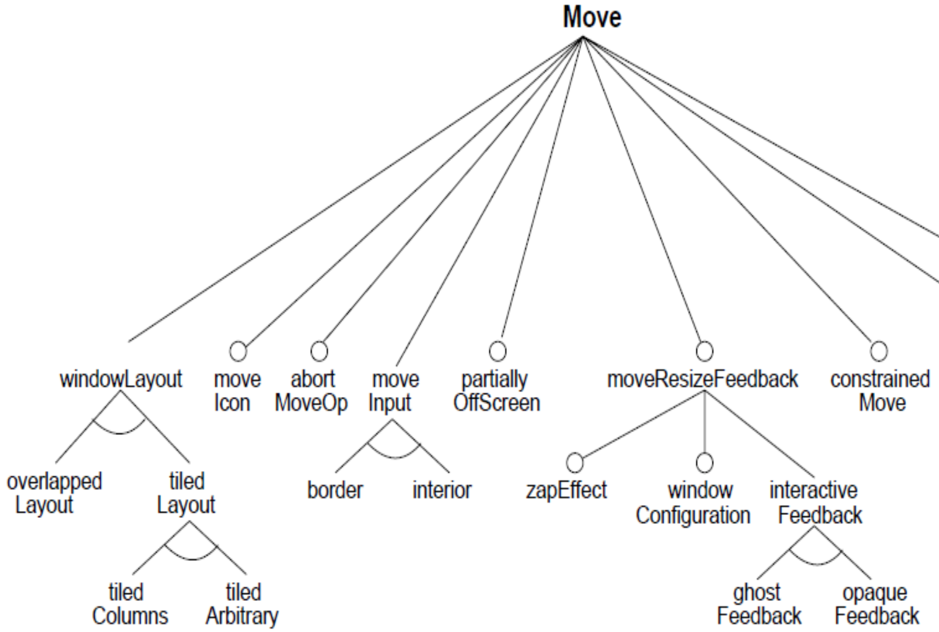
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

Original Feature Model
(KC Kang et al., 1990)

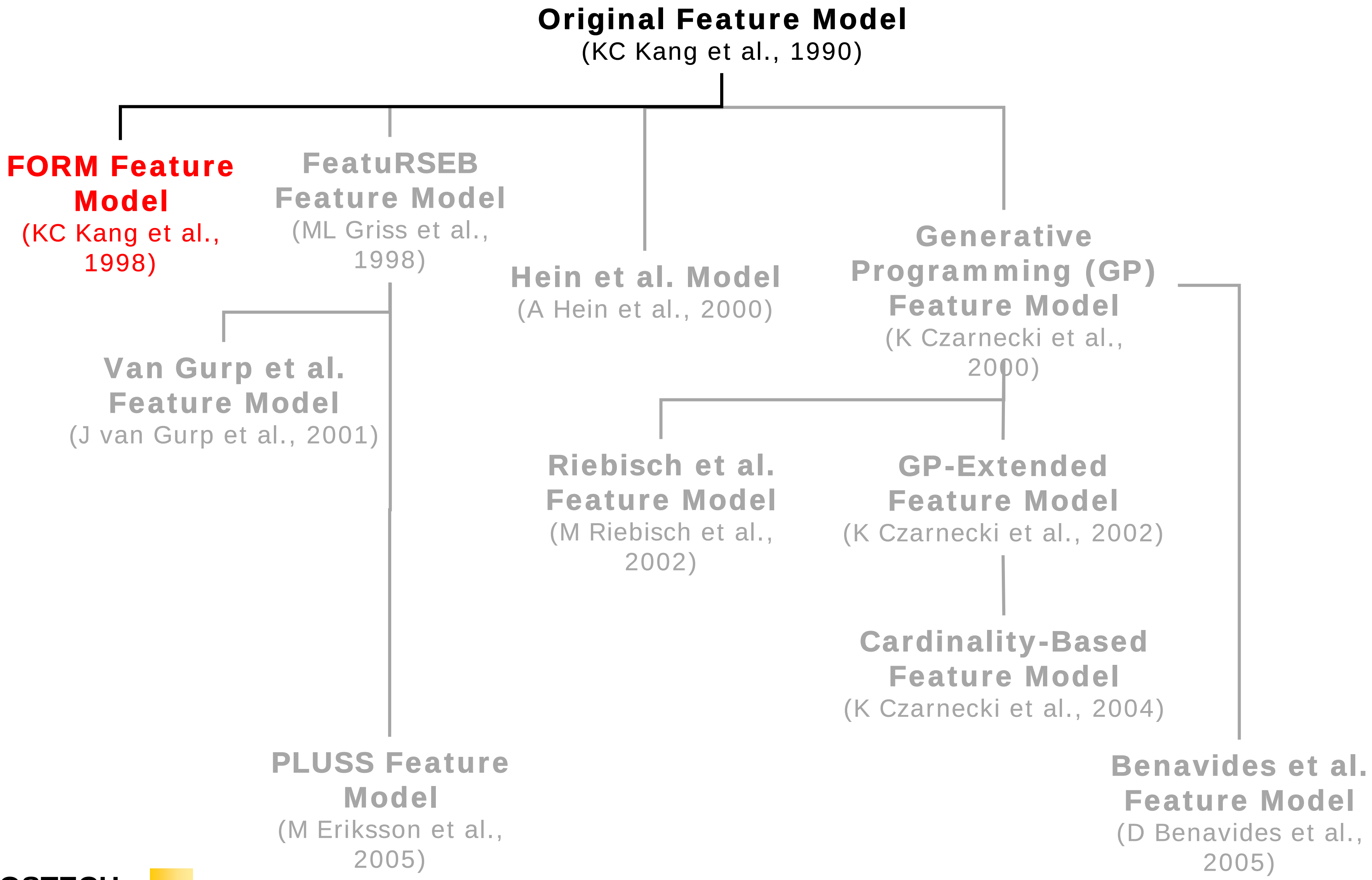


- Feature diagram: A Graphical And/Or Hierarchy of Features
  - Mandatory / Optional / Alternative Feature
  - Composed-of Relationship
- Composition Rules: Mutual Dependency (Requires) and Mutual Exclusion (Mutex-with) Relationships
- Issues and Decisions: Record of Trade-offs, Rationales, and Justifications
- System Feature Catalogue: Record of Existing System Features

Extensions from Original Feature Model

- Diagram Shape
- Layer
- Relationship Type
- Feature Type
- Feature Attribute
- Feature Cardinality
- Feature Group and Group Cardinality
- Constraint Notation
- Binding Time Notation

Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

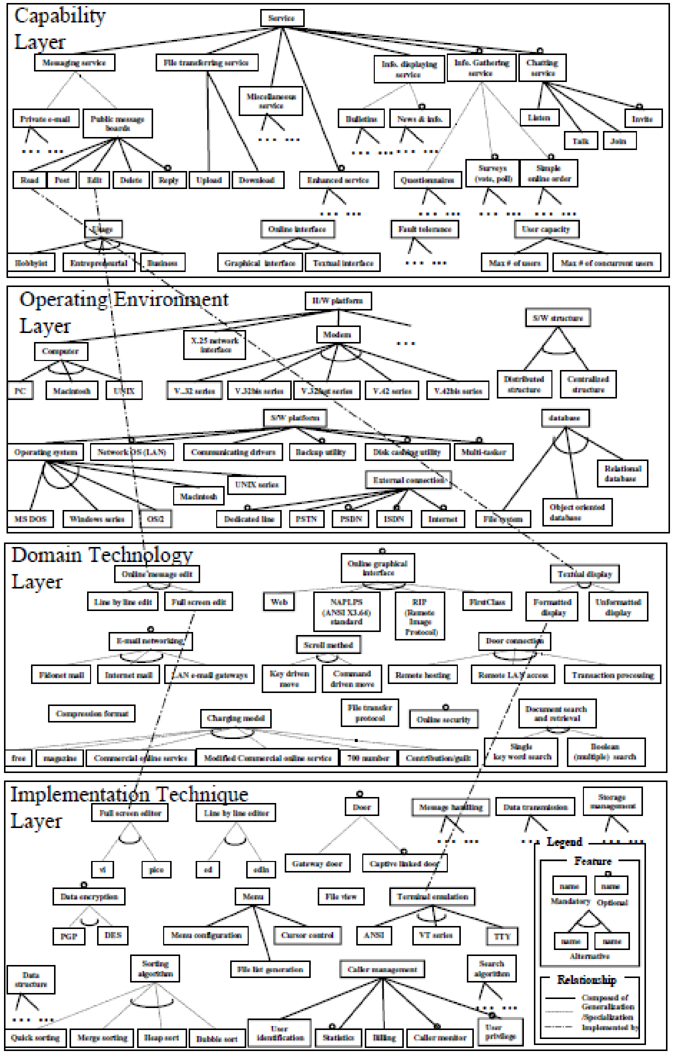
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

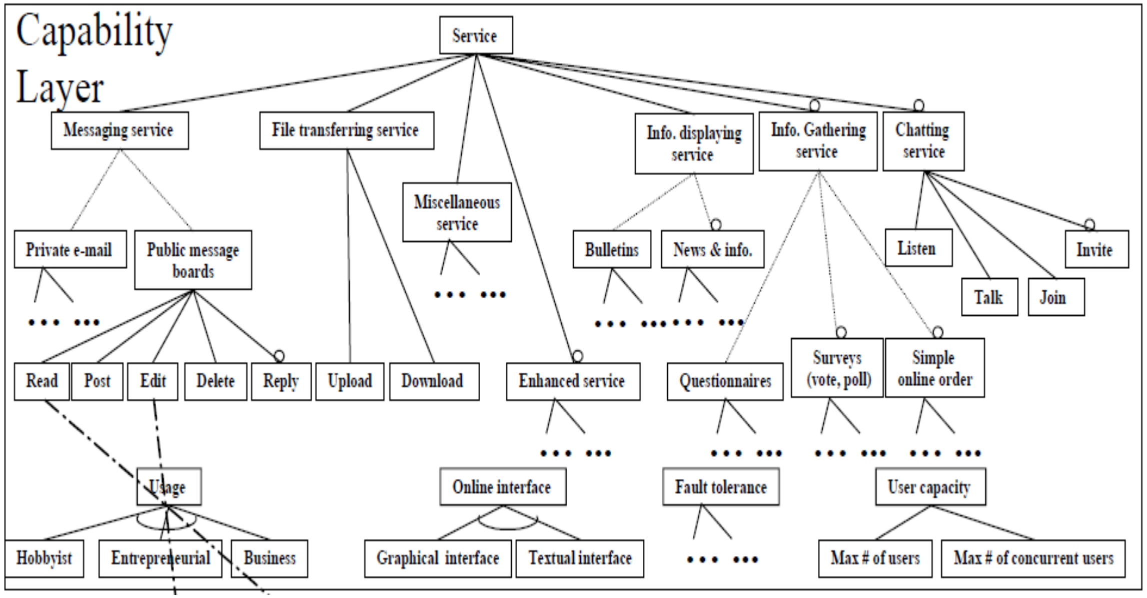
Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

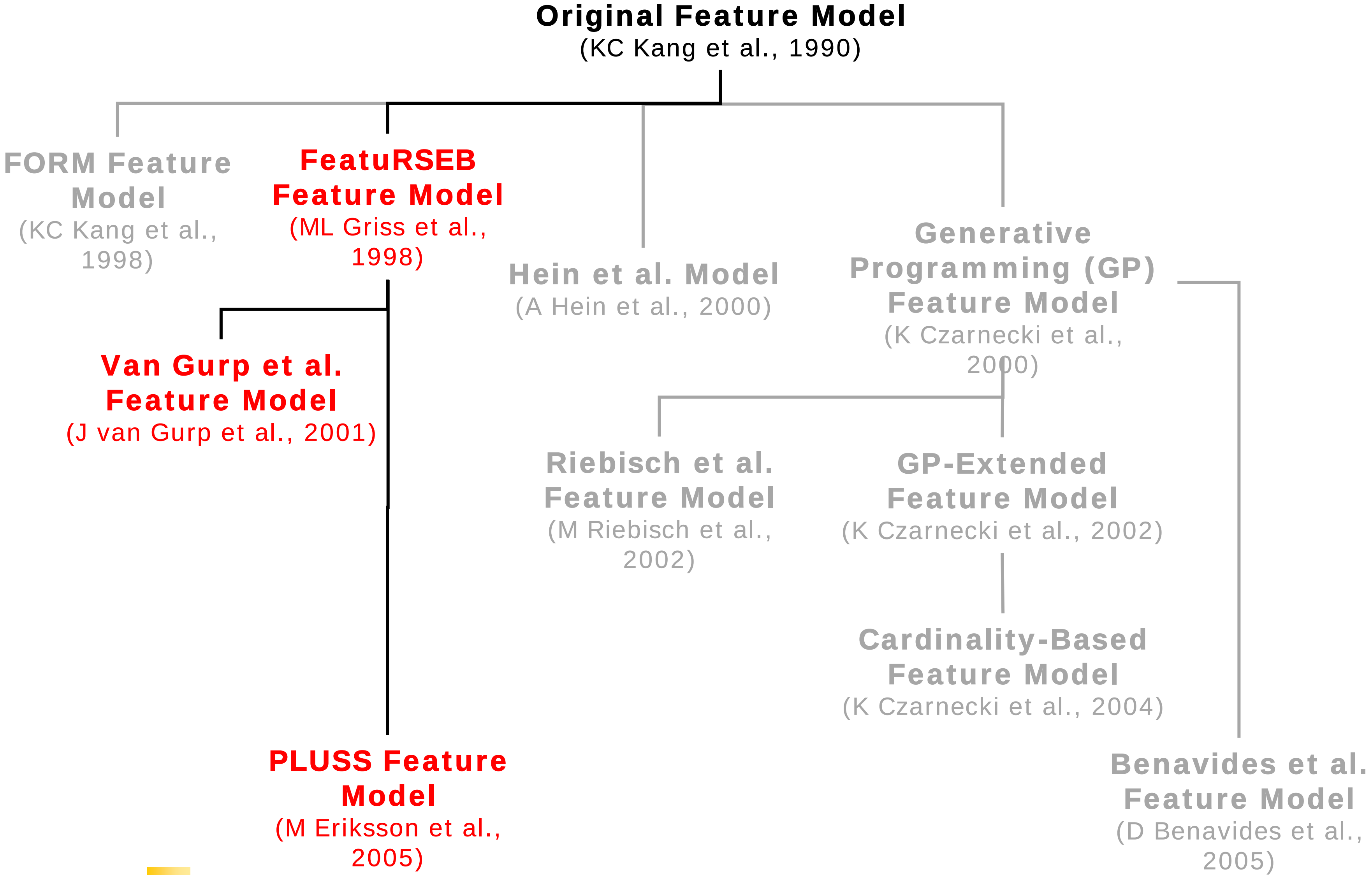
FORM Feature Model
(KC Kang et al., 1998)



- Layer
  - Capability
  - Operating Environment
  - Domain Technology
  - Implementation Technology
- Relationship Type:
  - Implemented-by



Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

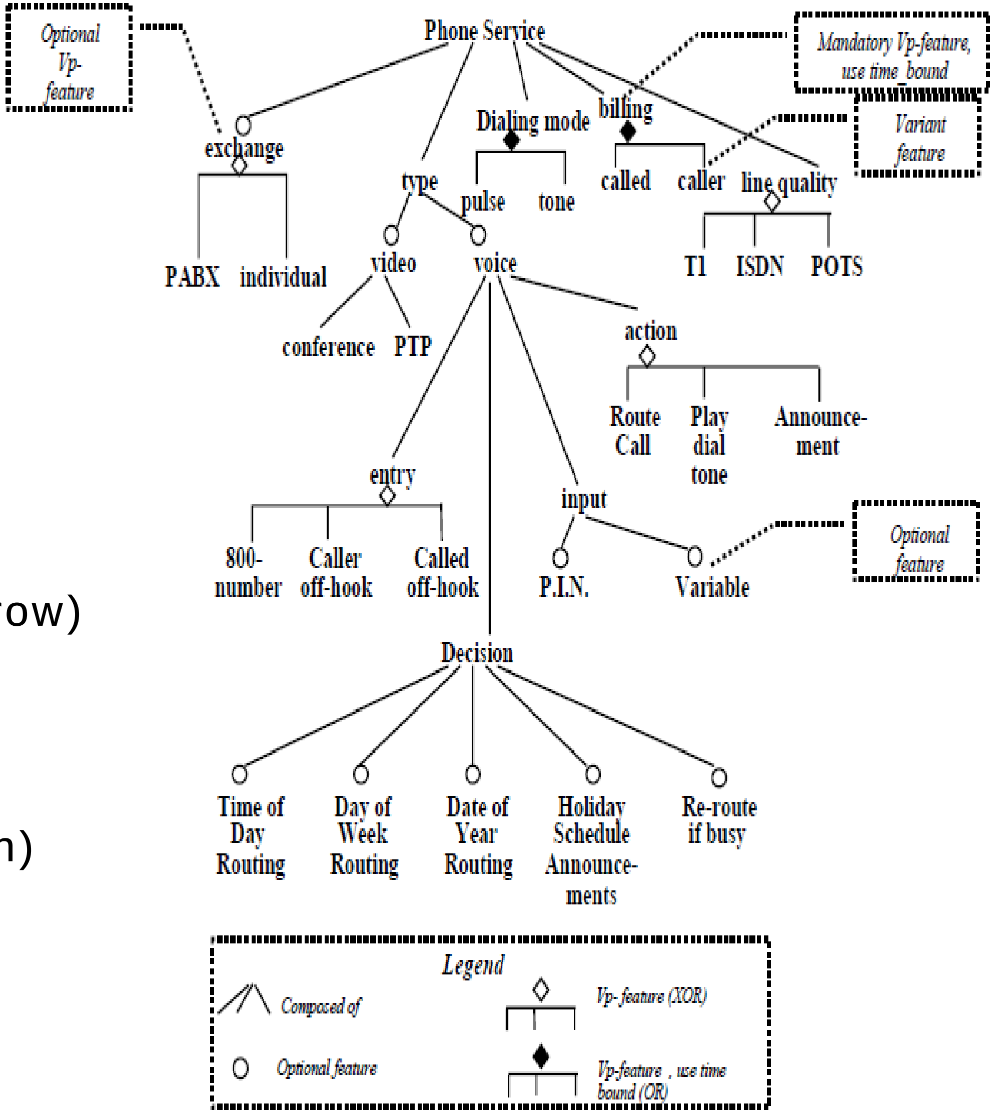
Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Combine FODA and the Reuse-Driven Software Engineering Business (RSEB)

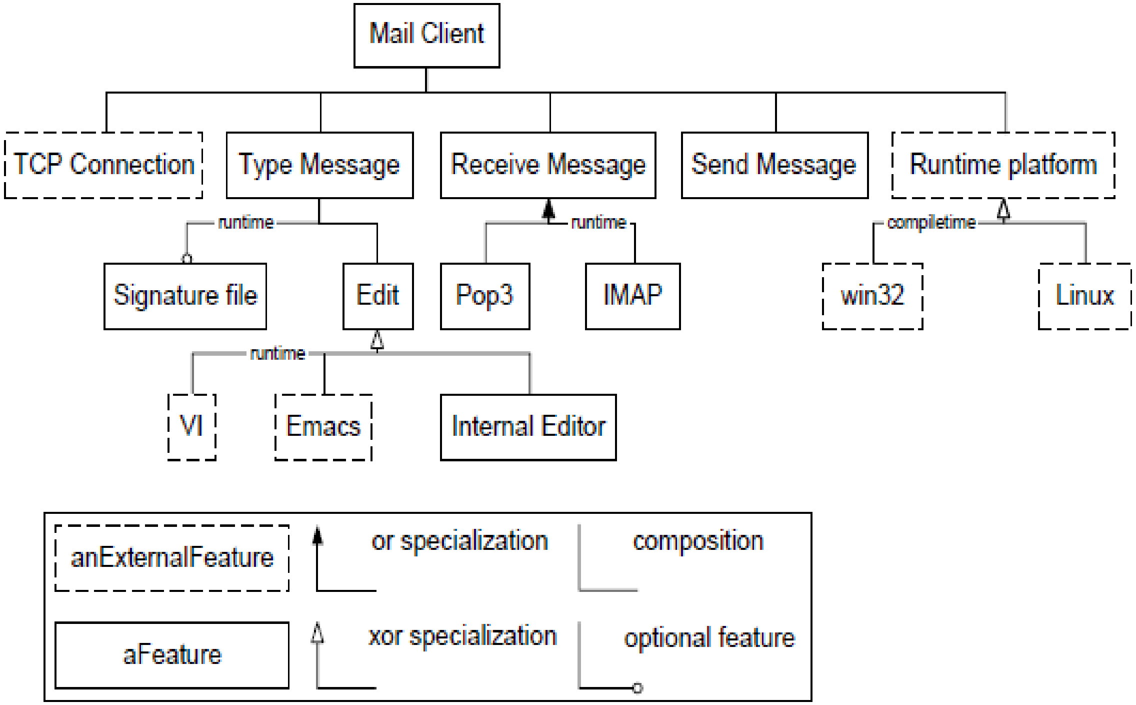
- Feature Type
  - Alternative Feature
    → Variation Point Feature / Variant Feature
- Constraint Notation (with Dashed Arrow)
- Bound Time Notation
  - Reuse Time Bound (XORed-disjunction)
  - Use Time Bound (ORed-disjunction)



Feature Model Genealogy

Van Gurp et al. Feature Model (J van Gurp et al., 2001)

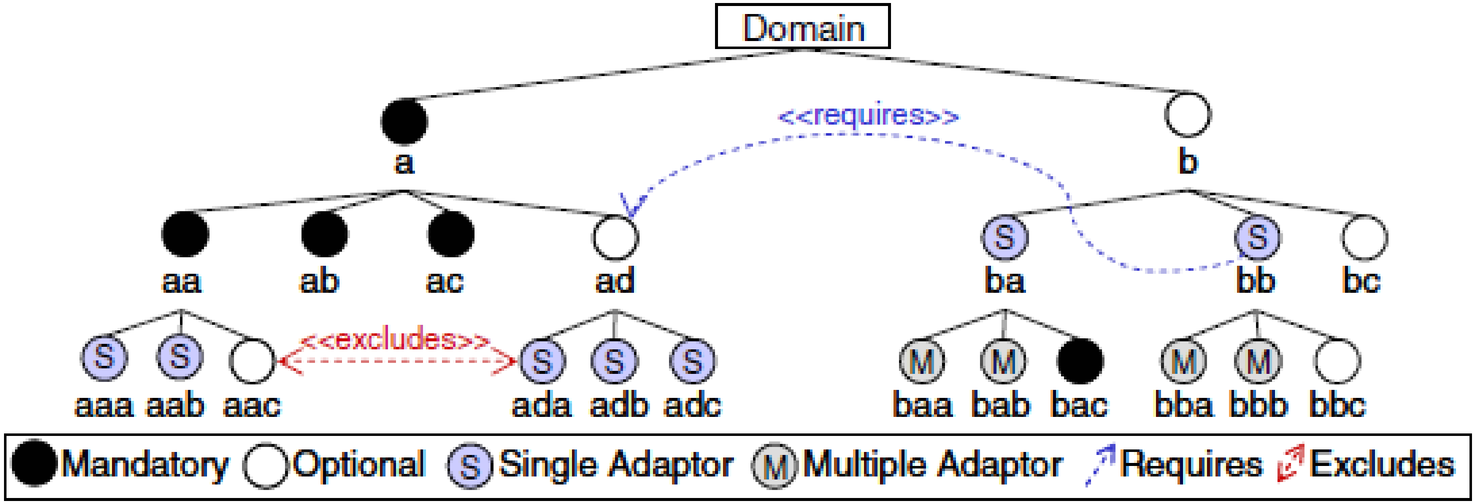
- Feature Type
  - External Feature
  - Alternative Feature
    → OR Specialization/ XOR Specialization
- Binding Time Notation

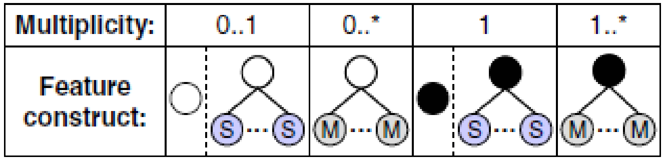


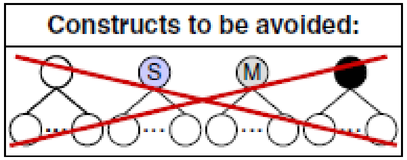
Feature Model Genealogy

PLUSS Feature Model
(M Eriksson et al., 2005)

- Diagram Shape
  - Feature as Circle (Black, White, ‘S’, ‘M’)
- Feature Type
  - Alternative Feature
    - → Single Adapter / Multiple Adapter
- Constraint Notation

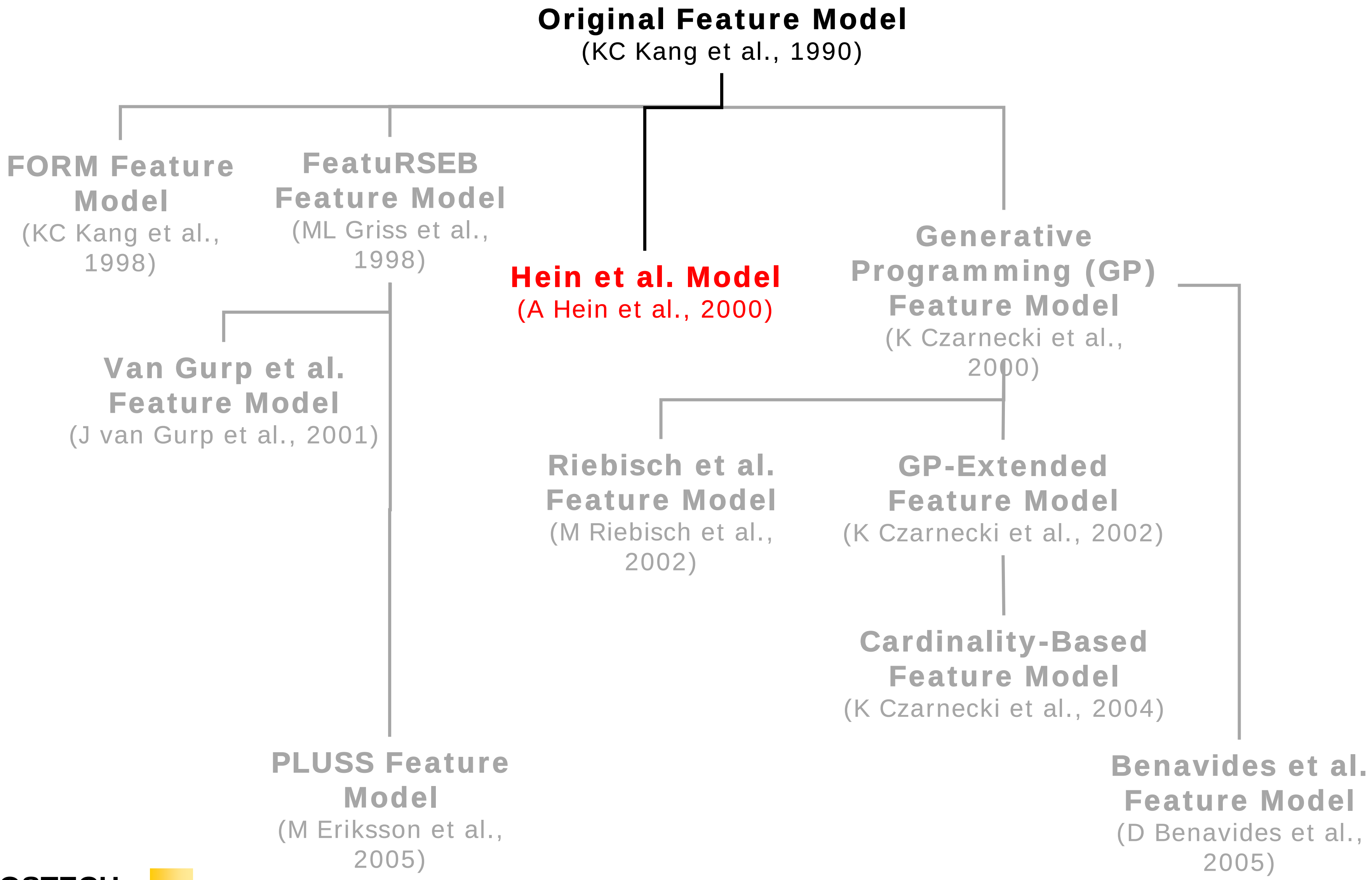






POSTECH

Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

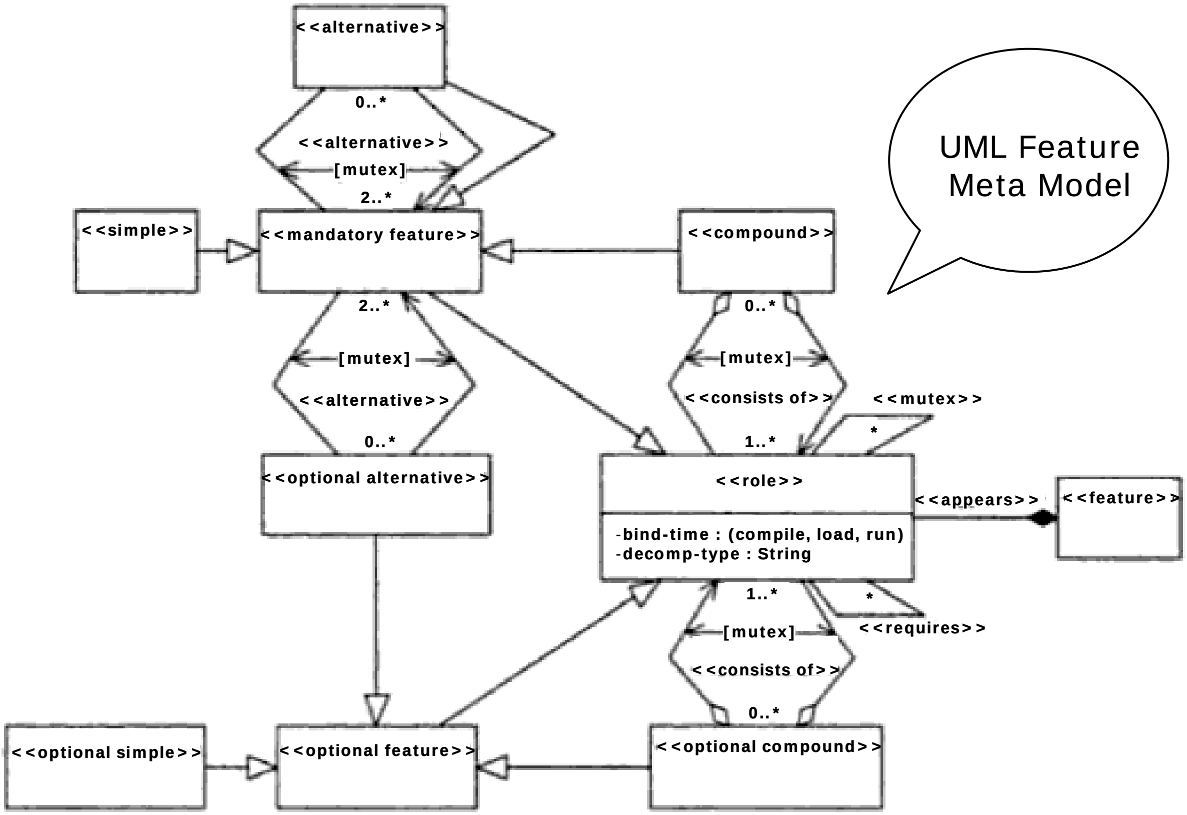
Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

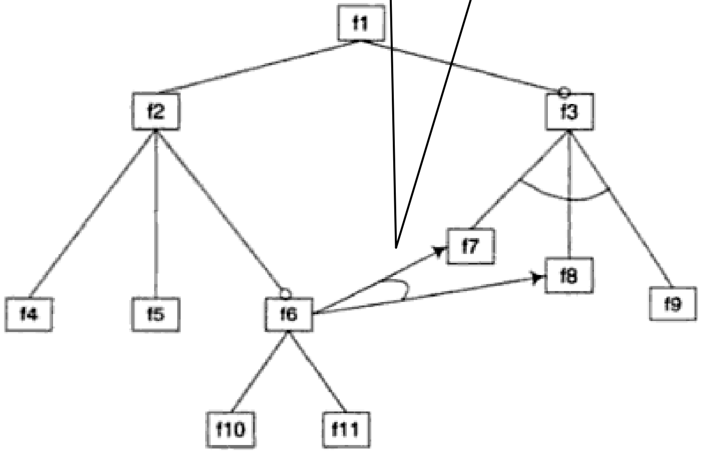
Hein et al. Feature Model (A Hein et al., 2000)

Use of UML

- Relationship Type
  - Arrow for Secondary Structure

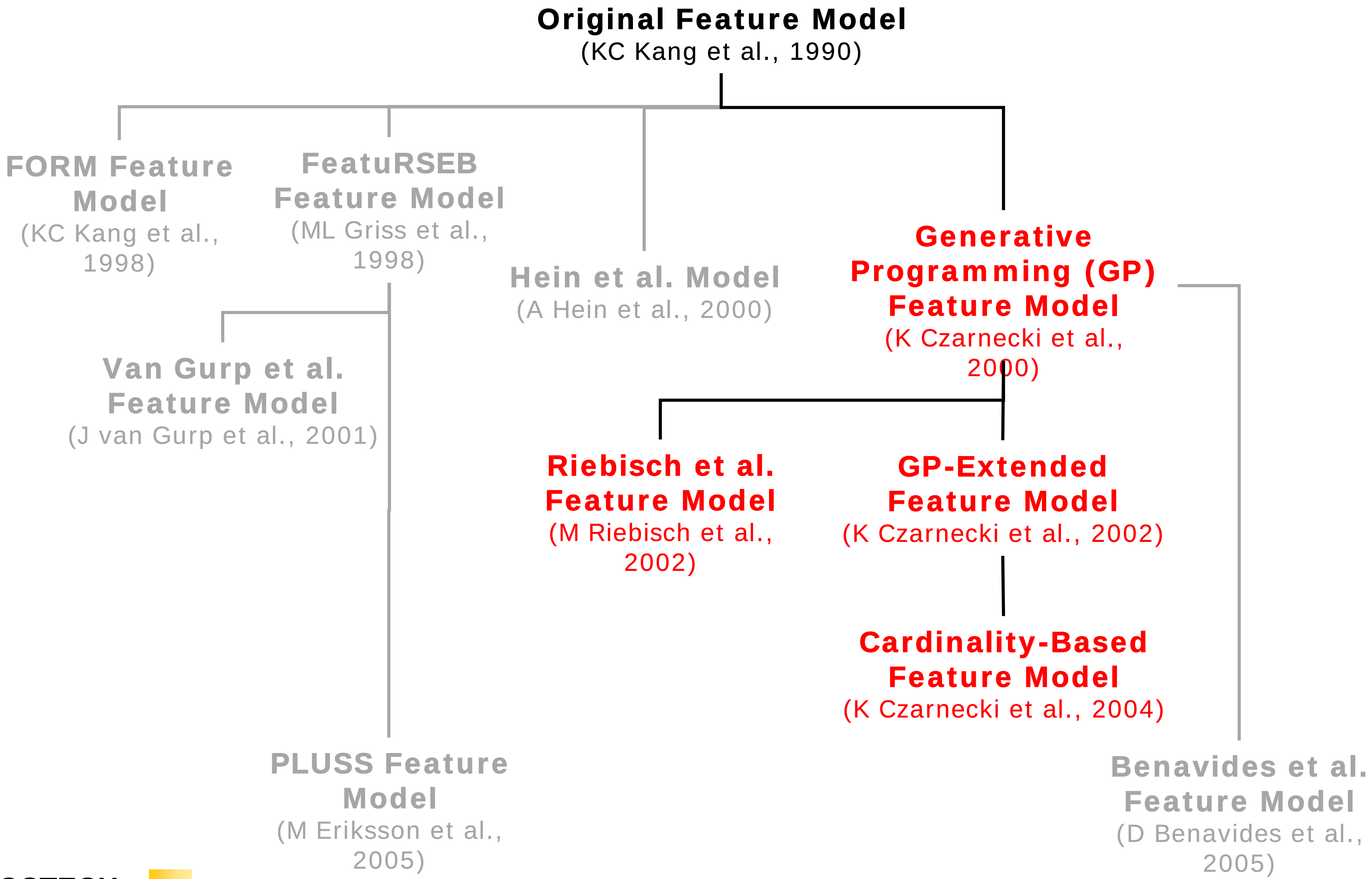


UML Feature Meta Model



“f6” is “Optional Compound” in Primary Structure and “Alternative” in Secondary Structure

Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

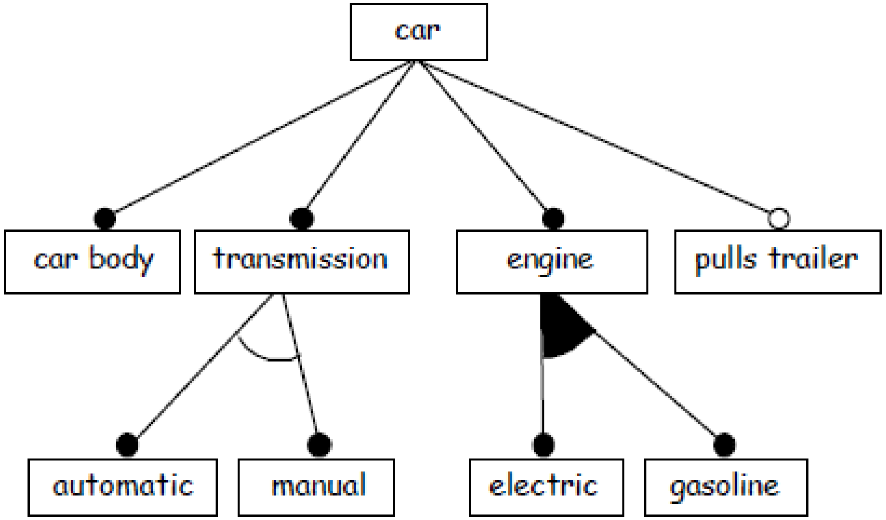
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

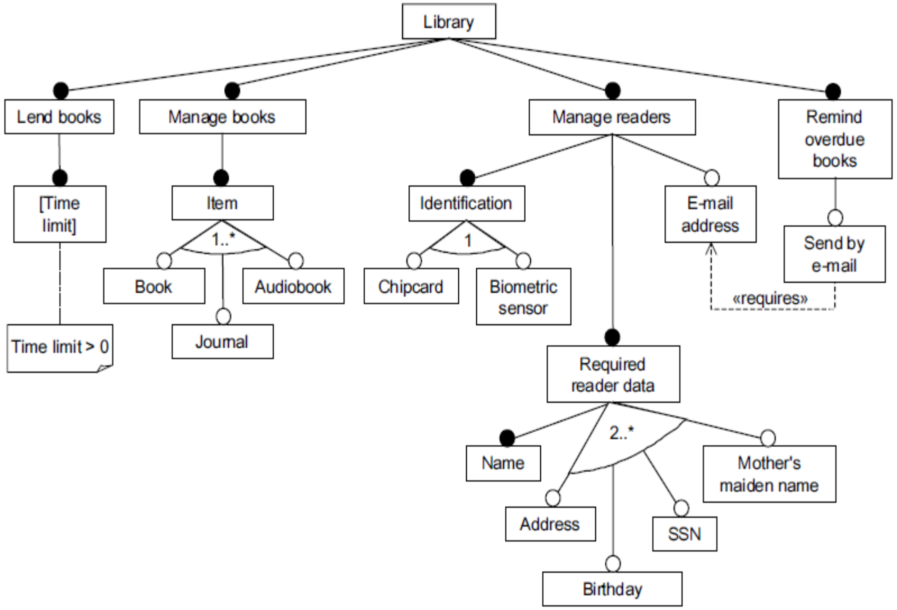
Generative Programming Feature Model (K Czarnecki et al., 2000)



- Diagram Shape
  - Feature Name in a Box
  - Box with Circle
    - : Black = Mandatory
    - : White = Optional
- Feature Type
  - OR Feature (Black Triangle)

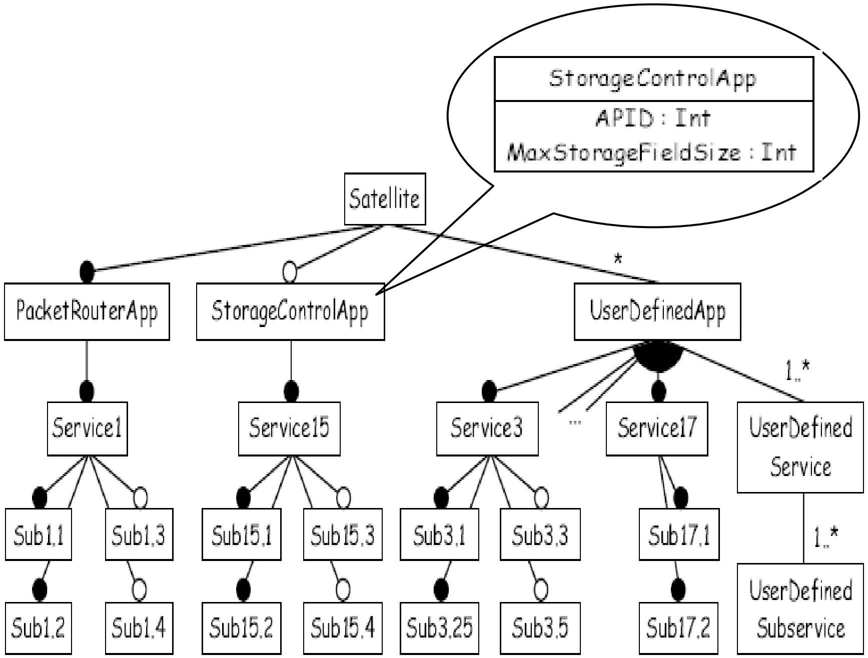
Feature Model Genealogy

Riebisch et al. Feature Model (M Riebisch et al., 2002)



- Diagram Shape / Feature Type
  - Inherit GP Feature Model
- Feature Group and Group Cardinality
- Constraint Notation

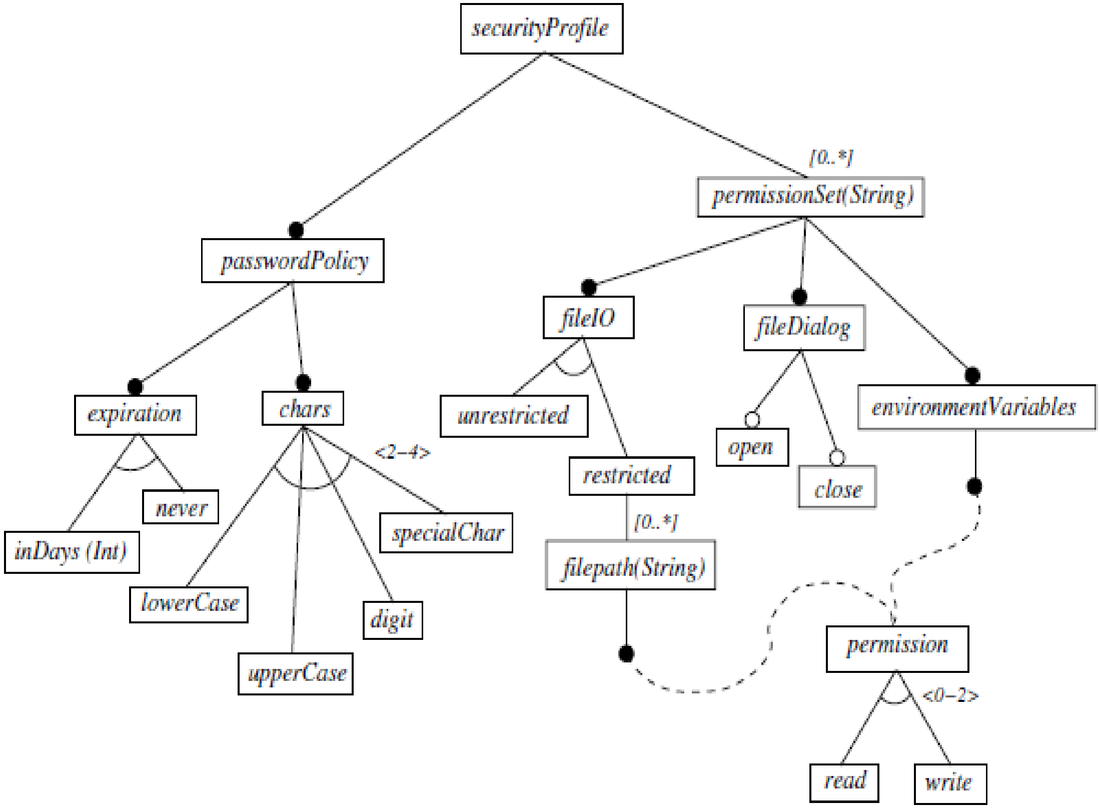
GP Extended Feature Model (K Czarnecki et al., 2002)



- Diagram Shape / Feature Type
  - Inherit GP Feature Model
- Feature Attributes
- Feature Cardinality

Feature Model Genealogy

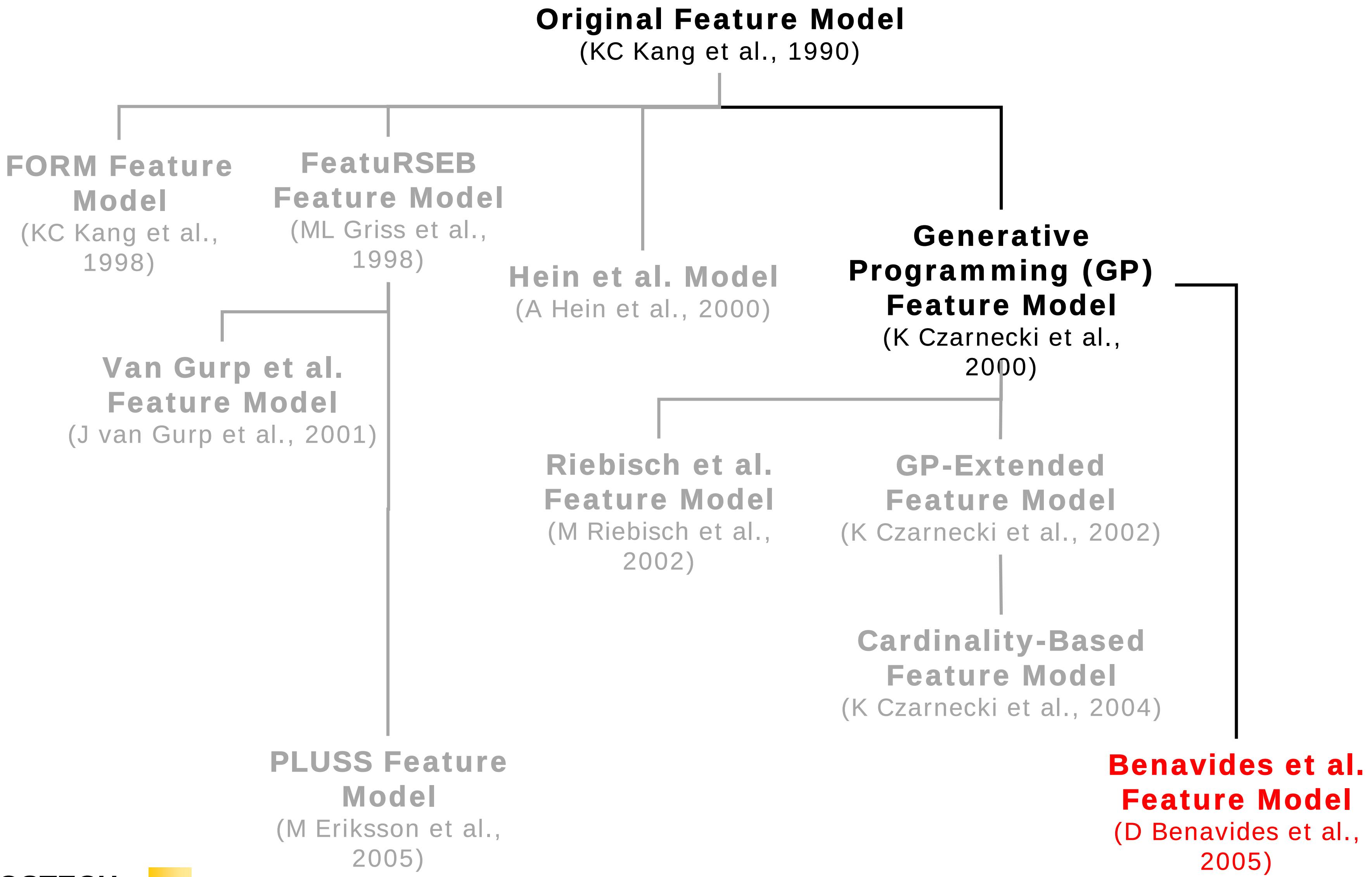
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)



- Diagram Shape / Feature Type
  - Inherit GP Feature Model
- Relationship Type
  - Feature Diagram Reference (Dashed Line)
- Feature Cardinality
- Feature Group and Group Cardinality

POSTECH

Feature Model Genealogy



Original Feature Model
(KC Kang et al., 1990)

FORM Feature Model
(KC Kang et al., 1998)

FeatuRSEB Feature Model
(ML Griss et al., 1998)

Hein et al. Model
(A Hein et al., 2000)

Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)

Van Gurp et al. Feature Model
(J van Gurp et al., 2001)

Riebisch et al. Feature Model
(M Riebisch et al., 2002)

GP-Extended Feature Model
(K Czarnecki et al., 2002)

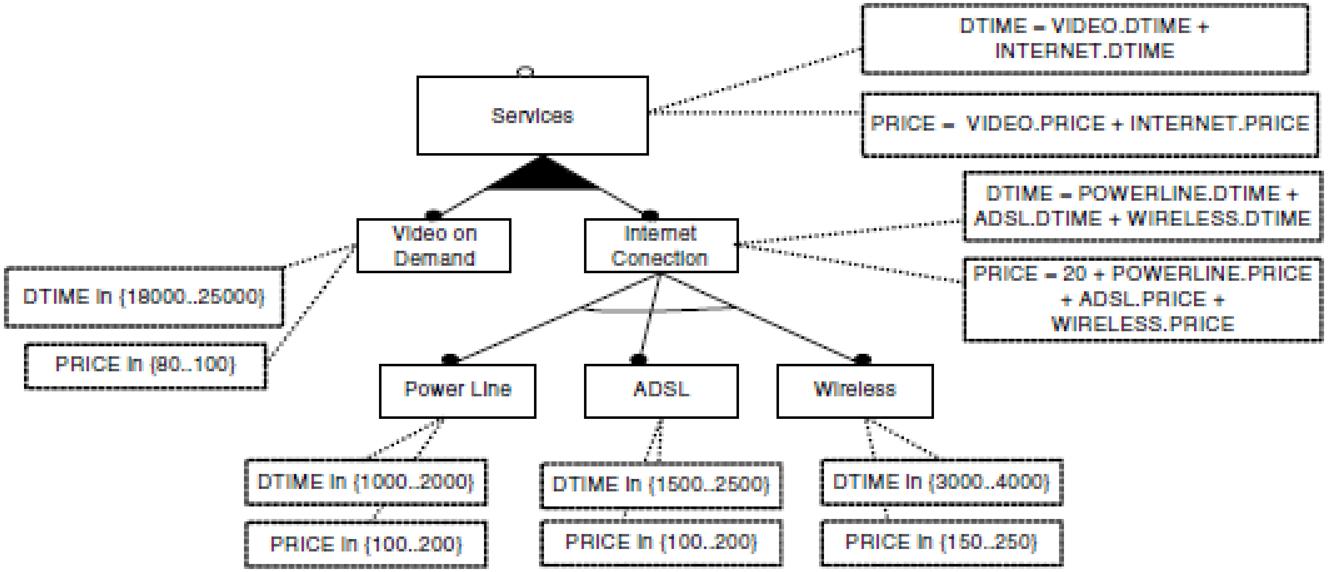
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)

PLUSS Feature Model
(M Eriksson et al., 2005)

Benavides et al. Feature Model
(D Benavides et al., 2005)

Feature Model Genealogy

Benavides et al. Feature Model
(D Benavides et al., 2005)



- Diagram Shape / Feature Type
  - Inherit GP Feature Model
- Relationship Type
  - Attribute Relationship (with Dashed Line)
- Feature Attribute

Why Popular?

- Codification of the Most Critical Information for Reuse
  - Commonality and Variability
  - Medium for Identifying Variation Points and Variants
- Simplicity
- Understandability
  - Intuitive
- Practicality
- Applicability

Salient Features of FODA Report

- " For example, features from the window manager domain such as *constrainedMove* and *zapEffect* could have been specified more precisely using a formal specification technique."

  → Formalization

- " If the domain is well-defined and is expected to remain stable, a preprocessor or an application generator development technique might be appropriate to process the compile-time features."

  → Generative Programming

- " The description should also indicate whether it is a compile-time, an activation-time, or a runtime feature."

  → Binding time, Dynamic Product Line

Salient Features of FODA Report

- " The classification of the features can be used in the components construction for modularization and for selection of appropriate development techniques."

  → Component Development

- " A record of the issues and decisions that arise in the course of the feature analysis must be incorporated into the feature model to provide the rationale for choosing options and selecting among several alternatives."

  → Configuration Decision Support

Agenda

- Introduction
- Looking Back
  - Number of Citations
  - Feature Model Genealogy
  - Why Popular?
  - Salient Features of FODA Report
- Looking Forward
  - Future Works
  - Other Issues
- Acknowledgement

Future Works

Feature analysis for different aspects at different phases of the life-cycle

PL Contexts	Requirements	Design	Implementation
Legal / Social Constraint User Profile Market / Business Usage Operating Environment Computing Resource / Platform Standards	Domain model Capability Use Case	Architecture Component Variation Point / Variants	Modules / Algorithms
	Configuration View Structural View Dependency View Binding View	FM-VP Decision Dependency Analysis (Consistency) QA Conformance Analysis View Consistency Analysis	Platform Conformance Analysis Variation Support Analysis Binding Time Support Connector Support

Decision Modeling (Rationales), Variability/Integrity Management

Other Issues

Domain analysis

- Different domains may require different approaches
  - Service analysis may be good for business applications domains
  - How about goal analysis?
  - "Goal -> Service->feature" as a unified method?
- Modeling mechanisms
  - Feature model is popular but many extensions
    - Should it be standardized?
  - Formalization
- Deciding the right level of abstraction; how to structure
- Feature explosion problem
  - How to model, analyze, and manage
  - High level of complex dependencies among them
- Feature interaction problem

Other Issues

- Goal-oriented (value-based) configuration of features
  - Knowledge-based configuration
  - Quality attributes or user-goals
- Going from domain analysis to architecture and component design
  - Designing architectures and components based on the analysis results (commonality and variability information)
    - SOA vs. agent-based vs. other architecture models
  - Building variability into architectures and components
  - Selecting appropriate mechanisms for the problem
  - Dealing with complex dependencies between features

Other Issues

- Specification of models
  - Reuse contexts and assumptions
- Verification of quality attributes of integrated systems
  - Safety, reliability, etc.
  - Detecting feature interaction problems
- Configuration management
  - Version control of components and architectures with multi-product nature
  - Evolution of the product line itself

Other Issues

- PL for systems in the newly emerging computing environments
  - Service Oriented Architecture
  - Ubiquitous computing environment/cloud computing
    - Dynamic binding of features
    - Run-time verification
  - From compile-time engineering to run-time engineering
    - Embedment of SE knowledge in running systems
- Tools!

Process

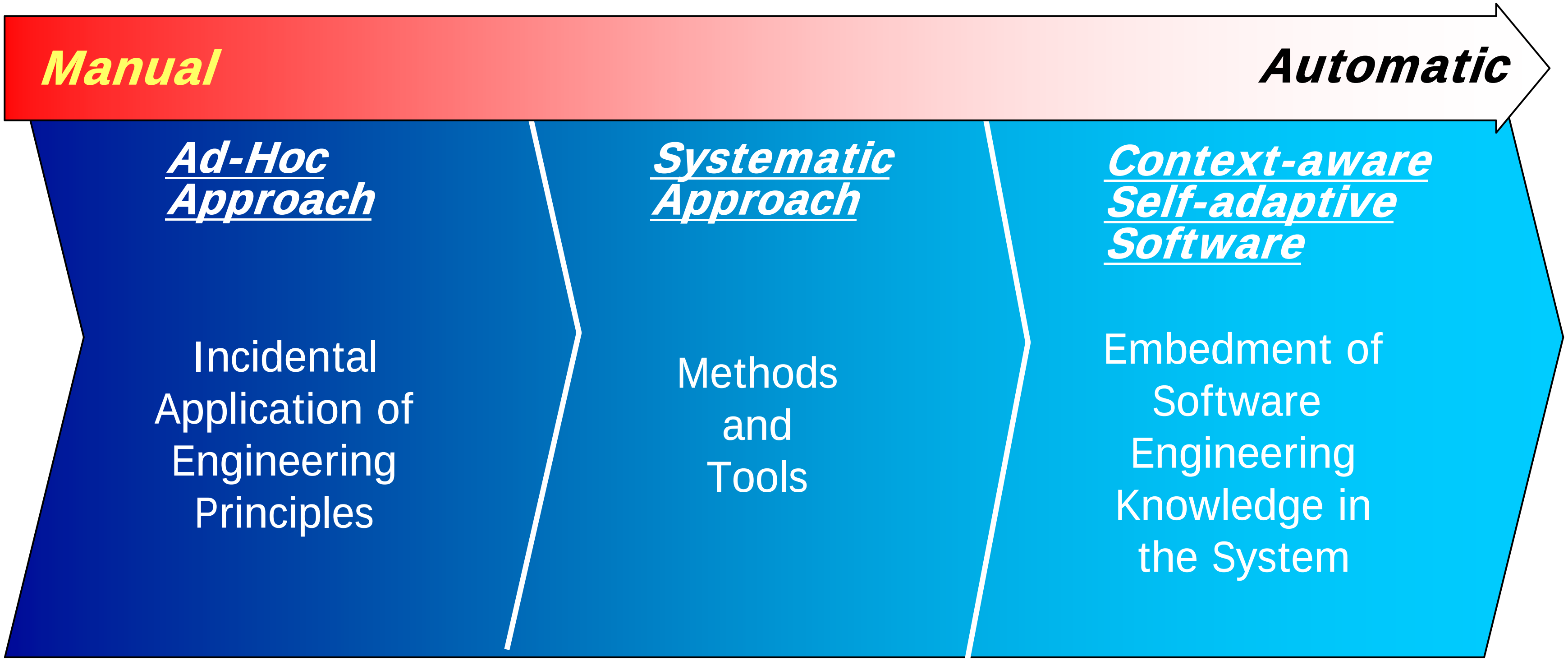
- How to change to PL-based organization
  - How to evolve: staged process model for reuse adoption
  - Key process areas
    - Best practices
  - Metrics
    - Key indicators: cost of production, time-to-market, project completion time, etc.
    - Relationship between reuse, quality, and productivity
    - Relationship between reuse and ROI for sustainability of a reuse program
- Process models
  - Proactive vs. reactive vs. extractive models
    - Best practices
  - PL process vs. agile methods

Other Issues

- Asset management (How to make PL-based development happen in an organization)-"Institutionalize" PL
  - Who should develop assets (with variation points)
  - Who should maintain assets (variability management)
  - Who will be responsible for quality assurance
  - Who should enforce the use of assets (policies)
  - Models (best practices)
    - Centralized vs. distributed
- Product line engineering in the context of a business strategy
  - "High option potentials"
  - ROI analysis
    - Estimating ROI from a reuse program
    - Estimating benefits from strategic market position
- Product line engineering in the global development environment
  - Component development outsourcing
    - Variability specification
    - Variability management

Embedment of SE Knowledge

Technical Trends


Manual
Automatic
Ad-Hoc Approach
Systematic Approach
Context-aware Self-adaptive Software
Incidental Application of Engineering Principles
Methods and Tools
Embedment of Software Engineering Knowledge in the System

- Modulization
- Information hiding
- Encapsulation
- ...

- Commonality and variability analysis
- Parameterization
- Template framework
- ...

- Monitor and dynamic reconfigurator
- Dynamic binding
- Architecture reconfiguration
- Run-time verification
- ...

Agenda

- ➢ Introduction
- ➢ Looking Back
  - Number of Citations
  - Feature Model Genealogy
  - Why Popular?
  - Salient Features of FODA Report
- ➢ Looking Forward
  - Future Works
  - Other Issues
- ➢ Acknowledgement

Acknowledgement

Special thanks to my students:

Hyesun Lee
Hyunsik Choi
Daesik Ham
Yoonho Cho
Yeonho Kim

for data analysis!

The End

Goldfish Bowl Panel
SPLC 2009:
How to maximize business return of SPL

08.09.2009 - v31

Participants

- Initial
  - Jan Bosch, Intuit
  - Ken Jackson IBM
  - Charles Krueger, BigLever
  - John McGregor, Clemson University
  - Andy Nolan, Rolls Royce
- More than 10 further goldfish in the bowl including Dirk Muthig, Ronny Kolb, Juha Savolainen, David Weiss, Stuart Jobbins, Kentaro Yoshimura....
- Organizers
  - Danilo Beuche, Mark Dalgano, Isabel John, Klaus Schmid, Christa Schwanninger

! Speed is the driver!

- There is only ONE reason for product lines: getting products out earlier
  - It's not the big that eat the small, it's the fast that eat the slow
  - We can never afford to slow down, because of the market pressure
  - But for introducing product lines, we have to slow down (at least a but), so the question is not the amount of variability but it's the duration of the stop that you have to take
  - ⊘ Big bang does NOT work
- There is no other reason (quality etc) as they are too hard to measure and are not as visible as number of products
- It's a products matter.. you have to find things that your customer really wants to pay for
- We have to take into account product innovation and process innovation
  - PL is process innovation
  - Product innovation comes from business
  - So there is a gap
- Counter example what about Iphone vs. Nokia, there speed was not enough
  - Yes, that's a risk
  - But Iphone is not really a product line
  - ? So the initial question is: Is it worth at all?

! Value is the driver !

- You need some direct value that immediately is there when you start introducing product lines... value for later is not the business that managers want
- We need scoping as a business activity, not as an engineering activity because business value is the driver, not engineering
- In your company YOU have to define, what is value for you
- ? It's not easy counting value.. so is there something easier? Having more variability is expensive, so is there a measure that counts the value of your portfolio
- You need an effective PL , not a valuable one because ROI gets larger when cost gets down. If you want to increase the value, build a cost model
- You need some direct value that immediately is there when you start introducing product lines... value for later is not the business that managers want

! Strategy is the driver !

- You do not need a cost model, you need a strategy to stay on the market and to discover new markets
- And we do not net a strategy that goes too far on the specialisation edge, but a general strategy for our product line
  - 💡 The hurdles to enter a business might disappear in the future
  - So rapidly capitalzing with a small team might be THE upcoming business model and THE upcoming strategy
- ?! For the strategy we have to decide: Business vs. Engineering: The engineering people do not understand the business that they are in (and vice versa?!?)

? The process is the driver?

- You need a certain maturity for starting with product lines, so getting this maturity is an important point
- Most of the conference is on engineering... but better engineering is not the best way to make money, you need something better
- Migrating to an automatic production process can be a driver
- You need an engineering process and a product line business for good PL benefit
- So one possible business driver could be having the options: think of variabilities as option.. as points where you have a quick choice to react on the market

!$ Cost is the driver!

- You need some deterministic factors and you need a const model in order to count the the business return
- You must understand and quantify the business side of your product line, then you can start maximizing its return
- If you do not define it for YOUR organisation, then you will fail
- To decide what is YOUR value, you can use one of the standard value or cost models , but you should use one
- There is a strong connection between dealing with Options and Products: The faster number of products is the thing that is visible from your options... so for externals, business managers, that's the thing that counts
- For some organisations, we need to take the engineering resources out of the equations for the cost model as they should not count when introducing PL
- Empirical data shows that after 3 products you get ROI
- But a business manager is interested in Revenues, not in ROI

SPLC 2009

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 • Airport Marriott, San Francisco, CA, USA

Software Engineering Institute | Carnegie Mellon

SPLC 2009

Important Dates:
Submission: March 6, 2009
Notification: April 30, 2009
Camera ready: May 30, 2009

Review and Evaluation Criteria:
Submissions will be evaluated according to the relevance, originality, and feasibility of the work. For each paper at least one reviewer will be available at the symposium and there will be a unique opportunity for discussion among reviewers and participants.

Acceptance:
Accepted research abstracts will appear in the SPLC 2009 Proceedings (second volume). All submitters will be expected to be able to meet the tight deadlines for camera-ready submissions and to present their work at the SPLC 2009 conference. Authors will be notified of acceptance by April 30, 2009.

Doctoral Symposium Chair:
Eduardo Santana de Almeida
Recife Center for Advanced Studies and
Systems - C.E.S.A.R
Reuse in Software Engineering - RiSE

Doctoral Symposium Committee Members:

- David Weiss, Avaya Labs
- Jan Bosch, Intuit
- Jaejoon Lee, Lancaster University
- John D. McGregor, Clemson University
- Klaus Schmid, University of Hildesheim

Symposium Organization:
The symposium is a half-day event to be held in conjunction with SPLC 2009. Each participant gets the chance to present his/her work (either as full presentation or as short presentation) and will get feedback from the panelists and the audience. In particular, the presenters will be provided with an opportunity for direct discussions with the reviewers.

Please visit the conference website for all details on deadlines, required formats, paper evaluation criteria, and so forth.

We invite you to be part of SPLC 2009. For more information about the venue, program, and conference organization, please visit the conference homepage at **www.sei.cmu.edu/splc2009/**.

Submission Guidelines for Doctoral Symposium

Eduardo de Almeida, Cesar, Brazil, SPLC 2009 Doctoral Symposium Chair

Goal:
The goal of the SPLC Doctoral Symposium is to provide a supportive, but challenging environment that enables students to further improve their research work leading to a Ph.D.

Students will have the opportunity to discuss their research, especially goals, methods, and preliminary results with the main researchers in the area. Thus, it is a unique opportunity for Ph.D. students to gather valuable expert feedback with respect to all aspects of their research work and to get into contact with other students who are at a similar stage of the Ph.D. research. The overall aim is to improve the quality and quantity of successful Ph.D. work in the area of software product lines.

Scope:
The event is dedicated to Ph.D. candidates (2nd year or later) with initial results that are still not mature enough for a full paper submission. The intent is to promote fruitful discussions and provide valuable feedback to the candidate, to be integrated into the final version of his/her thesis.

All topics that are relevant to the SPLC are also relevant to the doctoral symposium.

Submission and Evaluation:
How to Submit
Please read all of these instructions prior to submitting your paper.

To apply as a student participant in the Doctoral Symposium, you should prepare a submission package consisting of two parts, both of which must be submitted no later than the deadline, which is March 6, 2009.

Part I: Research Abstract
To participate, students should submit a research abstract electronically (PDF) to esa"at" rise.com.br. The submissions must be a maximum of 8 pages in the IEEE proceedings 8.5x11-inch, Two-Column Format. All submissions must be in English and in PDF format. To submit your abstract go to http://cyberchairpro3.borbala.net/splcpapers/submit/.

The research abstract should cover:

- The technical problem to be solved with a justification of its importance.
- A description of the related and prior work explaining why the identified problem has not been solved.
- The research hypothesis or claim.
- A sketch of the proposed solution.
- The expected contributions of your thesis research.
- Progress in solving the stated problem.
- The methods you are using or will use to carry out your research.
- A plan for evaluating your work and presenting credible evidence of your results to the research community.

Students at relatively early stages of their research will have some difficulty addressing some of these areas, but should attempt to do so the best they can. The research abstract should include the title of your work, your name, email address, postal address, personal website, and a one paragraph short summary in the style of an abstract for a regular paper. Submissions should contain no proprietary or confidential material and should cite no proprietary or confidential publications.

Part II: Letter of Recommendation
Ask your thesis advisor for a letter of recommendation. This letter should include your name and a candidate assessment of the current status of your thesis research and an expected date for thesis submission. The letter should be in PDF, and sent to: esa" at"rise.com.br with the subject of: SPLC-Doctoral-Symposium.

SPLC 2009

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 • Airport Marriott, San Francisco, CA, USA

Software Engineering Institute | Carnegie Mellon

SPLC 2009

General Chair
Dirk Muthig, Fraunhofer IESE, Germany

Program Chair
John D. McGregor, Clemson University, USA

Industry Track
Paul Jensen, Overwatch, USA
Kentaro Yoshimura, Hitachi, Japan
Michael Schumpelt, ETAS, Germany

Workshops
Jaejoon Lee, Lancaster University, UK

Demonstrations and Posters
Ronny Kolb, Honeywell, Switzerland

Doctoral Symposium
Eduardo de Almeida, Cesar, Brazil

Tutorials
Gary Chastek, Software Engineering Institute, USA

Publicity
Pat Donohoe, Software Engineering Institute, USA

Please visit the conference website for all details on deadlines, required formats and paper evaluation criteria.

We invite you to be part of SPLC 2009. For more information about the venue, program, and conference organization, please visit the conference homepage at **www.sei.cmu.edu/splc2009/**.

Call for Participation in SPLC 2009 Industry Track

Industry Track Co-chairs:
Paul Jensen, Overwatch, USA
Kentaro Yoshimura, Hitachi, Japan
Michael Schumpelt, ETAS, Germany

For more than a decade, organizations have been taking advantage of software product line practices to achieve business advantages in time to market, cost, quality, and agility. These organizations have encountered a wide range of challenges, successes, and adaptations in their software product line experience. We are seeking contributions from industry that share those challenges, successes, and adaptations during all stages of software product line maturity—ranging from adoption to evolution.

Specific topics of interest are

- experiences in implementing a software product line
  - retrospectives that summarize your organization’s experience with software product lines, including the context in which you implemented a product line, the challenges you faced, and how those challenges were addressed
- tools and technologies used in implementing a software product
  - how your organization used or adapted tools in your software product line experience
- software product line architectures
  - summarize the architecture that was implemented to support your organization’s software product line, emphasizing the unique attributes that made it suitable for this purpose
- production planning
  - summarize your organization’s experience with production planning including the tools and methods used
- issues that are not adequately addressed by researchers with respect to creating and running a software product line

We invite you to present your perspectives to your peers in the product line community at SPLC 2009. Experience reports will be reviewed by fellow practitioners from an industry perspective.

Submissions
Submitted reports must not exceed 10 pages in the IEEE Computer Society Conference Format for 8.5x11-inch Proceedings Manuscripts. Accepted reports will be published electronically.
To submit your abstract go to http://cyberchairpro3.borbala.net/splcpapers/submit/.

Important Dates

Submission:	March 20, 2009
Notification:	April 28, 2009
Camera ready:	May 20, 2009

www.sei.cmu.edu/splc2009/

SPLC 2009

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 • Airport Marriott, San Francisco, CA, USA

Software Engineering Institute | Carnegie Mellon

SPLC 2009

General Chair
Dirk Muthig, Fraunhofer IESE, Germany

Program Chair
John D. McGregor, Clemson University, USA

Industry Track
Paul Jensen, Overwatch, USA
Kentaro Yoshimura, Hitachi, Japan
Michael Schumpelt, ETAS, Germany

Workshops
Jaejoon Lee, Lancaster University, UK

Demonstrations and Posters
Ronny Kolb, Honeywell, Switzerland

Doctoral Symposium
Eduardo de Almeida, Cesar, Brazil

Tutorials
Gary Chastek, Software Engineering Institute, USA

Publicity
Pat Donohoe, Software Engineering Institute, USA

Please visit the conference website for all details on deadlines, required formats and paper evaluation criteria.

We invite you to be part of SPLC 2009. For more information about the venue, program, and conference organization, please visit the conference homepage at **www.sei.cmu.edu/splc2009/**.

.

Call for Workshops

Jaejoon Lee (j.lee at comp.lancs.ac.uk), SPLC 2009 Workshop Chair

The Organizing Committee of SPLC 2009 invites submissions for workshop proposals.

The purpose of the workshop program is to provide a platform for bringing together people from industry, academia, and research institutions to present and discuss experiences and practices in the area of software product line development. Workshops that address the specific needs of major industry sectors such as automotive, mobile communications, and medical systems are particularly welcome.

Workshop position papers will be distributed in a flash drive at the conference site.

Workshops should be organized as full-day events, and they are expected to be arranged on the first and second conference days, August 24th and 25th, 2009.

Proposals for workshops (max. three pages) should contain:

- title of the workshop
- summary of the workshop including
  - description of objectives in relation to the conference
  - list of workshop topics
  - intended audience
- preliminary schedule for the workshop
- preliminary dates for workshop submissions
- name, postal address, phone number, and email address of main organizer (primary contact, one person only please)
- name, postal address, phone number, and email address of co-organizers
- references to previous workshops (e.g., websites) - if applicable
- technical requirements (beamer, whiteboard etc.)

A one-page summary of each accepted workshop will be published in the SPLC proceedings.

The summary should include a motivation, a list of workshop topics, and references (max. three references).

All workshop proposals must conform to the IEEE proceedings 8.5x 11," two-column format.

Please send your workshop proposals to Jaejoon Lee (j.lee at comp.lancs.ac.uk), SPLC 2009 Workshop Chair, by March 6, 2009. You can download the proposal template at www.sei.cmu.edu/splc2009/files/CallforWorkshops.doc.

Important Dates:
Workshop Proposal Submission: March 6, 2009
Notification of Acceptance: March 20, 2009
1-Page Workshop Summary (Camera-Ready): TBD

www.sei.cmu.edu/splc2009/

SPLC 2009

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 • Airport Marriott, San Francisco, CA, USA

Software Engineering Institute | Carnegie Mellon

SPLC 2009

General Chair
Dirk Muthig, Fraunhofer IESE, Germany

Program Chair
John D. McGregor, Clemson University, USA

Industry Track
Paul Jensen, Overwatch, USA
Kentaro Yoshimura, Hitachi, Japan
Michael Schumpelt, ETAS, Germany

Workshops
Jaejoon Lee, Lancaster University, UK

Demonstrations and Posters
Ronny Kolb, Honeywell, Switzerland

Doctoral Symposium
Eduardo de Almeida, Cesar, Brazil

Tutorials
Gary Chastek, Software Engineering Institute, USA

Publicity
Pat Donohoe, Software Engineering Institute, USA

Please visit the conference website for all details on deadlines, required formats and paper evaluation criteria.

We invite you to be part of SPLC 2009. For more information about the venue, program, and conference organization, please visit the conference homepage at **www.sei.cmu.edu/splc2009/**.

Call for Tutorial Proposals

Gary Chastek, Software Engineering Institute, SPLC 2009 Tutorials Chair

Tutorials provide a valuable opportunity for conference participants to expand their product line knowledge and skills. Tutorials may focus on introductory product line topics, such as how to introduce a product line approach into an organization, or on more advanced applied topics, such as industrial product line engineering practices.

Tutorials will be held during the conference week in half-day sessions.

A tutorial proposal consists of approximately two pages describing the topic, the plan for conducting the tutorial, and the backgrounds of the presenters and the tutorial.

- The Topic section should include the title of, goals of, and intended audience for the tutorial. The topic should be described in detail, stressing its importance and timeliness.
- The Plan section should include a
  - preliminary schedule of events including estimated times
  - detailed description of what the tutorial will cover
  - justification of the tutorial for a product line audience
  - explanation of how the tutorial will be conducted, including sample materials to be included in the tutorial notes
- The Presenters' Backgrounds section should include relevant biographical information and summaries of the presenters' technical, presentation, and tutorial experience.
- The Tutorial Background section should include a description of where and when the tutorial has been offered previously and any evaluations that were done.

A two-page description of all accepted tutorials will be published in the conference proceedings.

Important Dates:

Email tutorial proposals are due to gjc@sei.cmu.edu by March 16, 2009.

Acceptance notification will occur by April 17, 2009.

www.sei.cmu.edu/splc2009/

SPLC 2009

13th International Software Product Line Conference (SPLC)

August 24–28, 2009 • Airport Marriott, San Francisco, CA, USA

Software Engineering Institute | Carnegie Mellon

SPLC 2009

General Chair
Dirk Muthig, Fraunhofer IESE, Germany

Program Chair
John D. McGregor, Clemson University, USA

Industry Track
Paul Jensen, Overwatch, USA
Kentaro Yoshimura, Hitachi, Japan
Michael Schumpelt, ETAS, Germany

Workshops
Jaejoon Lee, Lancaster University, UK

Demonstrations and Posters
Ronny Kolb, Honeywell, Switzerland

Doctoral Symposium
Eduardo de Almeida, Cesar, Brazil

Tutorials
Gary Chastek, Software Engineering Institute, USA

Publicity
Pat Donohoe, Software Engineering Institute, USA

Please visit the conference website for all details on deadlines, required formats and paper evaluation criteria.

We invite you to be part of SPLC 2009. For more information about the venue, program, and conference organization, please visit the conference homepage at **www.sei.cmu.edu/splc2009/**.

.

Call for Tool and Demonstration Proposals

Ronny Kolb, Honeywell, SPLC 2009 Demonstrations and Posters Chair

Tools and the automation achieved through them are an important aspect of efficiently implementing software product lines in industrial practice and further improving productivity.

We are soliciting demonstrations of academic, open source, in-house, and commercial tools that support and automate specific aspects of product line engineering such as feature modeling, variant management, derivation and generation of products, product line testing, and so forth. Demonstrations of original, novel tools for some purpose, new versions of existing tools with a clearly identifiable new contribution, as well as customized extensions of standard tools are welcome. Commercial tool vendors are encouraged to demonstrate their tools together with an industrial customer using concrete examples.

In addition to demonstrations of practically applying product line engineering using (tailored) standard or product-line-specific tools, we are interested in

- approaches and results of evaluating and selecting product line tools
- integration of product line tools with general software development tools
- integration of product line tools in existing single-system tool chains
- extensions of standard product line tools for specific needs
- integrated solutions for the whole product line life cycle

In addition to tool demonstrations, we are interested in demonstrations of industrial practice for various activities in the product line engineering life cycle. Demonstrations are aimed at showing state of the practice in implementing product line engineering and exchanging practical experiences and challenges. Provided there is evidence of use in actual practice, process definitions and/or simulations using tools such as the Eclipse Process Framework (EPF) or IBM's Rational Method Composer (RMC) can be presented.

All those who wish to demonstrate a product-line-related tool or industrial practice should send their proposal of up to two pages in a free format to Ronny Kolb (ronny.kolb@honeywell.com) by **March 6, 2009**. Proposals will be evaluated beginning **March 6, 2009** and will continue until the deadline for camera-ready copy. A one-page paper about each accepted demo will be published in the conference proceedings.

www.sei.cmu.edu/splc2009/

2009 Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL 2009)

Enhancing Variation

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Bob Krut & Sholom Cohen
25 August 2009
San Francisco, California, USA

Agenda

8:30 - 8:45	Welcome, Agenda, and Summary of Previous Workshops
8:45 - 9:30	Invited Talk: Jaejoon Lee, Lancaster University, UK
9:30 - 10:00	Paper #1 Presentation & Discussion
10:00 - 10:30	Break
10:30 - 11:00	Paper #2 Presentation & Discussion
11:00 - 11:30	Paper #3 Presentation & Discussion
11:30 - 12:00	Context Setting for Afternoon Discussion
12:00 - 13:30	Lunch
13:30 - 14:00	Paper #4 Presentation & Discussion
14:00 - 15:00	Workshop Discussions
15:00 - 15:30	Break
15:30 - 16:30	Workshop Discussions Continued
16:30 - 17:00	Conclusion: Goals Addressed, Topics for SOAPL 2010, Future Work

Workshop Organizers

Sholom Cohen, Software Engineering Institute, USA

David Benavides, University of Seville, Spain

Dragan Gasevic, Athabasca University, Canada

Andreas Helferich, Universität Stuttgart, Germany

Robert Krut, Software Engineering Institute, USA

Grace Lewis, Software Engineering Institute, USA

Dennis Smith, Software Engineering Institute, USA

Christoph Wienands, Siemens Corporate Research, USA

Peter Dolog, Aalborg University, Denmark

The First Workshop on Service-Oriented Architectures and Product Lines (SOAPL 2007)

Part of the 2007 Software Product Line Conference (SPLC 2007),
10 September 2007, Kyoto, Japan.

Service Oriented Architectures and Product Lines - What is the Connection?
(http://splc.net/prev-conferences/soapl-2007.pdf)

SOAPL 2007 explored the connections from two perspectives:

1. Can services support product lines using a service-oriented architecture?
2. How can use of product line practices support services and service-oriented architectures?

Proceedings of the First Workshop on Service-Oriented Architectures and Product Lines (CMU/SEI-2008-SR-006).

http://www.sei.cmu.edu/publications/documents/08.reports/08sr006.html

The Second Workshop on Service-Oriented Architectures and Product Lines (SOAPL 2008)

Part of the 2008 Software Product Line Conference (SPLC 2008),
8 September 2008, Limerick, Ireland

Service Oriented Architectures and Product Lines - Putting Both Together (http://splc.net/prev-conferences/soapl-2008.pdf)

SOAPL 2008 explores experiences in integrating SOA and SPL:

1. How web services have been used to support product lines using a service-oriented architecture?
2. How product line practices have been used to support web services and service-oriented architectures?

Workshop papers are published in the
Proceedings of the 12th International Software Product Lines Conference (SPLC 2008), *Second Volume.* Limerick, Ireland, September 8-12, 2008. University of Limerick, Ireland: Lero International Science Centre, 2008 (ISBN 978-1-905952-06-9).

The outcome of SOAPL 2008 discussion will be the basis of today’s workshop.

The Third Workshop on Service-Oriented Architectures and Product Lines (SOAPL 2009)

Service-Oriented Architectures and Software Product Lines - Enhancing Variation (http://www.sei.cmu.edu/splc2009/soapl.html)

SOAPL 2009 will explore how service-oriented architectures and software product lines can benefit from each other, specifically

1. how service-oriented systems can benefit from software product lines' variation management approaches to identify and design services targeted to multiple service-oriented systems
2. how software product lines can benefit from service-oriented architectures by employing services as a mechanism for variation within a product line

Four position papers were accepted.

Accepted Papers

1) “*Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions*”
   Dennis Smith, Grace Lewis

2) “*Semantic Variability Modeling for Multi-staged Service Composition*”
   Bardia Mohabbati, Nima Kaviani, Dragan Gasevic

3) “*Managing SOA System Variation through Business Process Lines and Process Oriented Development*”
   Nicola Boffoli, Marta Cimitile, Fabrizio Maria Maggi, Giuseppe Visaggio

4) “*Towards an Approach for Service-Oriented Product Line Architectures*”
   Flávio Mota Medeiros, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira

Papers are available at http://www.sei.cmu.edu/splc2009/soapl.html

Workshop Theme

The two major themes addressed from the accepted SOAPL 2008 papers were:

1. Variability and variability mechanisms
2. Product composition

The aspects of scope, design approach, source of variation, application target, composition elements, and technical approach were the focus of discussion during the workshop.

These discussions lead to four principles for SOA and PL variation:

1. recognizing the commonality and variants across the scope of a product line or across some group of service-oriented systems within the enterprise
2. leveraging the recognized commonality by building core assets, including services, across the variants with established points of variation
3. recognizing the enterprise integration needs that service-oriented systems must address
4. addressing end user needs for variation within service-oriented systems

Workshop Theme (cont.)

For SOAPL 2009, we would like to address these four principles by focusing on

1. Scope - identifies those entities with which products in the product line will interact (that is, the product line context), and it also establishes the commonality and sets limits on the variability of the products in the product line. [Northrop, L. & Clements, P. *A Framework for Software Product Line Practice, Version 5.0* <http://www.sei.cmu.edu/productlines/framework.html> (2009).]

2. Source of variation – What is the source of variation within a scope or across products that define a new product line? Do SOA methods help identify new services and their variations?

3. Variation management - comprises all activities to explicitly model, manage, and document those parts, which vary among the products of a product line. [John, I.; Pech, D. *Scalable Variability Instantiation Strategies*. Scalable Modeling Techniques for Software Product Lines (SCALE 2009) Workshop, SPLC 2009.]

4. Variation mechanisms - a mechanism to support the creation and/or selection of variants that are compliant with the constraints for a variable part of a core asset . [Bachmann, F. & Clements, P. *Variability in Software Product Lines* (CMU/SEI-2005-TR-012, ADA450337). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.]

And from this years papers

1. Ontology's, semantic variability modeling
2. Business process variation

What other approaches do people use to address these four principles?

Invited Speaker

Dr. Jaejoon Lee
Lecturer
Computing Department
Lancaster University, UK

Main Research Areas:

- Product Line Software Engineering
- Software Architecture
- Service-oriented Software Systems

Presentation Title: “*A Negotiation Framework for Service-Oriented Product Line Development*”

Paper Presentation

Presenter: Dennis Smith
Senior Member of Technical Staff
Software Engineering Institute
Carnegie Mellon University

Main Research Areas:

- Migrating Legacy Systems to SOA Environments
- Integration of Software-Intensive Systems
- Service-oriented Architectures
- Software Product Lines

Presentation Title: "*Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions*"
Authors: Dennis Smith, Grace Lewis

Paper Presentation

Presenter: Bardia Mohabbati
Ph.D Student
Simon Fraser University
Laboratory for Ontological Research

Main Research Areas:

- Semantic Web Techniques in Software Engineering
- Software Language Engineering
- Business Process Modeling
- Software Product Line
- Service-oriented Architecture

Presentation Title: “*Semantic Variability Modeling for Multi-staged Service Composition*”
Authors: Bardia Mohabbati, Nima Kaviani, Dragan Gasevic

Paper Presentation

Presenter: Nicola Boffoli

PresentationTitle: “*Managing SOA System Variation through Business Process Lines and Process Oriented Development*”

Authors: Nicola Boffoli, Marta Cimitile, Fabrizio Maria Maggi, Giuseppe Visaggio

Paper Presentation

Presenter: Flávio Mota Medeiros

Presentation Title: “*Towards an Approach for Service-Oriented Product Line Architectures*”
Authors: Flávio Mota Medeiros, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira

Context Setting for this Afternoon

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Author: Sholom Cohen
Date: August 24, 2009

Goals

Use case study approach

- Identify commonality and variation within case study context
- Analyze C&V and consider variation mechanisms

Use case study as a starting point on an example product line

- define meaning of variation in scope, service features, others
- identify variation mechanisms, requirements for business process variation
- provide example product ontologies
- other topics covered in the various position papers.

Highlight means to construct a product line solution within case study context using SOA and PL approaches

Other goals:

How

Hear example application based on SOA and Product Line approaches

- “Towards an Approach for Service-Oriented Product Line Architectures” Flavio Mota Medeiros, Eduardo Santana de Almeida,Silvio Romero de Lemos Meira
- Listen for ways to use this example to
  - Highlight variations at several levels (scope, architecture, component)
  - Apply variation mechanisms to deal with these variations (feature, service or component)
  - Explore ontologies, business process variation, and other themes

Product: Create enhanced version of example application to illustrate bringing SOA and SPL approaches together.

Use SOA design as a pattern for multiple product lines

Case study is directed at the conference paper review process

- provides a pattern for scoping other potential product lines that involve
  - Submitting a data item for review
  - Comment on data item
  - Reporting result and suggesting follow up
- examples
  - medical record review and reporting
  - trouble report submission and tracking
  - item order tracking system

Think of other product lines as we further develop the pattern

- Consider infrastructure and other services they might share.
- What specific services would they require?

Product

- a sketch of common and specific services used within each product line
- unique services to support the specific target market

Example – Medical Related Services

Create scenarios

- Examples: a patient registers, a patient submits an insurance card, a clinician selects an report for review, a clinician submits a diagnosis, …
- Expand scenarios using activity diagrams

Identify and highlight commonality and variations

- in activity diagrams
- as tasks and patterns
- in feature model

Create use cases with extensions

- Identify actors
- Create use cases and dhow variations in use case dialogs
- Recognize patterns and model (e.g., context diagram)

Identify services and select service mechanisms

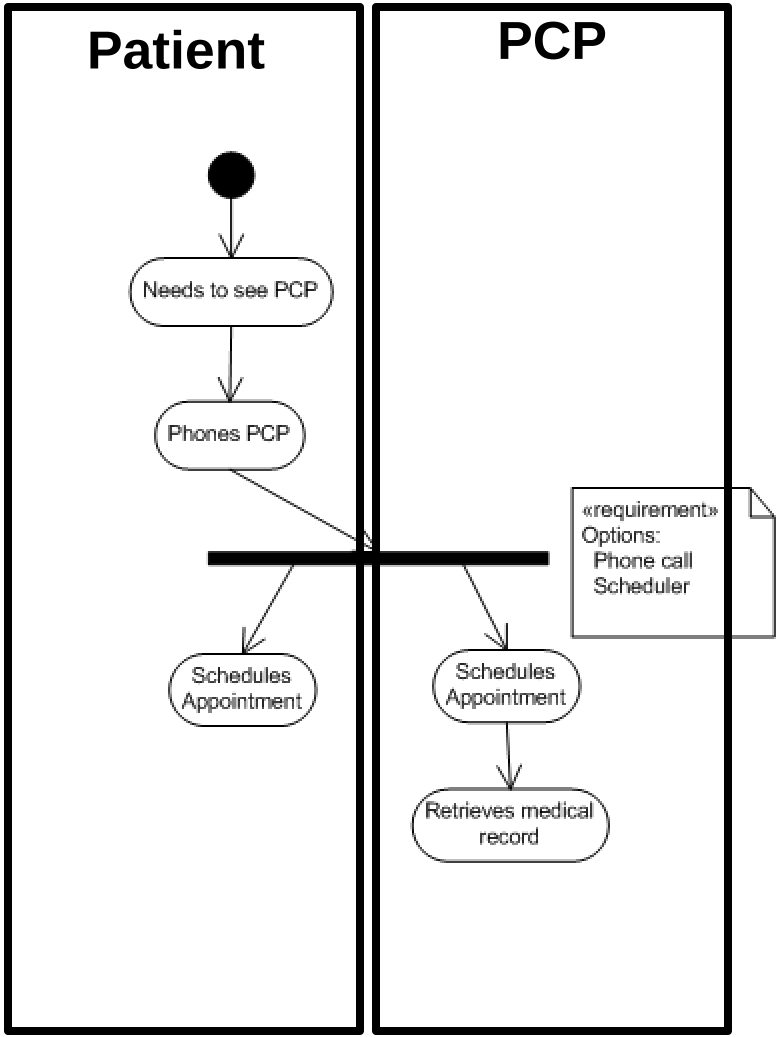
A Medical Story – Chapter 1

A patient with some history of cardiac problems decides to see his primary care physician

Call to doctor’s office

- “If you are otherwise all right, we can see you next week.”

Build an Activity Diagram



A Medical Story – Chapter 2

The patient checks in

- presents insurance card
- makes co-payment (relevant outside United States?)
- taken to exam room with medical chart

Assistant performs preliminaries and records in (paper/electronic) chart

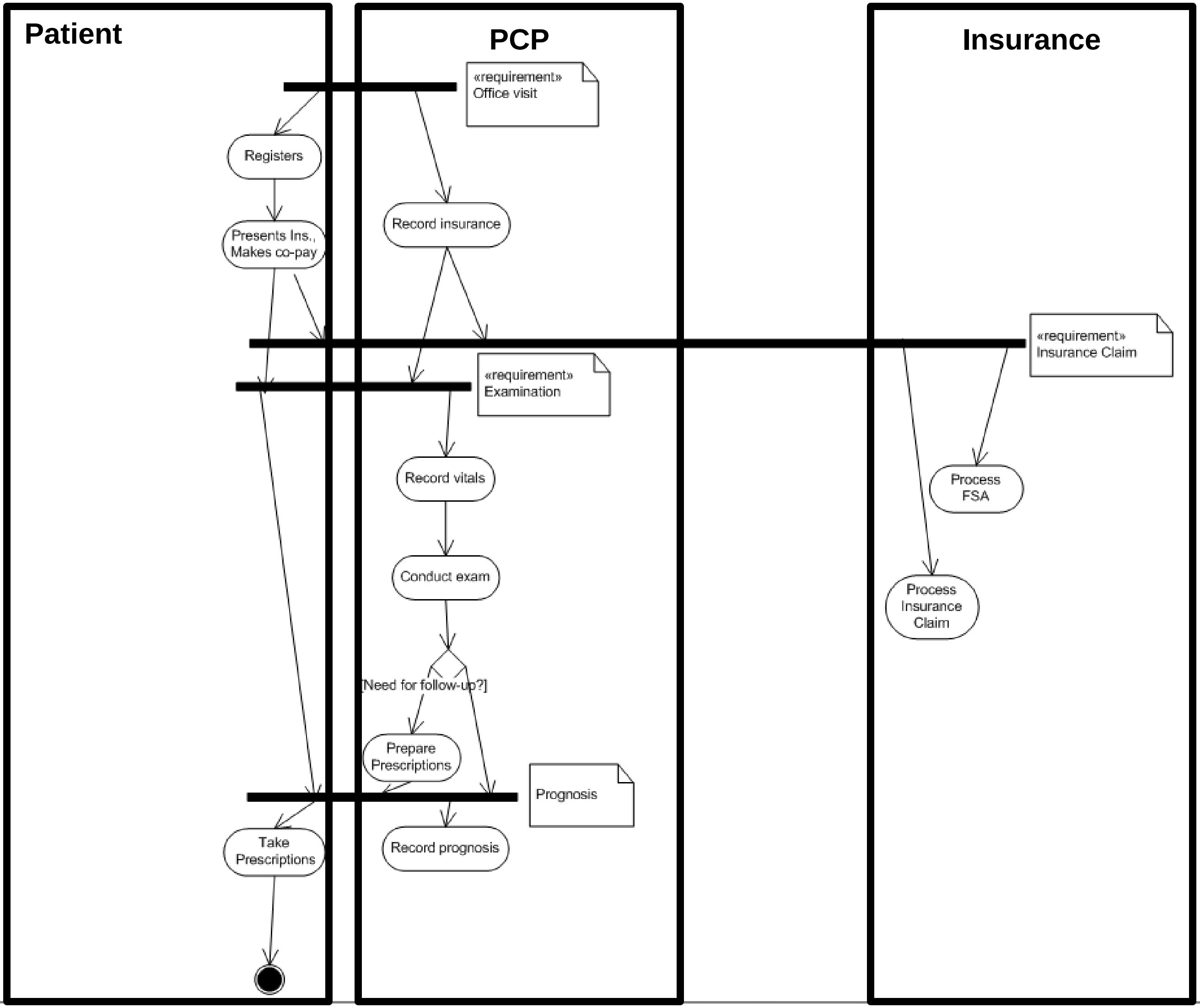
- weight, vitals, etc.
- reason for visit

Doctor examines patient (ontology support to build this?)

- patient overweight
- blood pressure marginal risk
- family history risk
- records report
- prescribes EKG, blood work, stress echocardiogram

Patient takes prescriptions and proceeds to hospital lab

Activity - Office Visit



A Medical Story – Chapter 3

The patient checks into lab

- presents insurance card
- ~~makes co-payment (not that good)~~
- taken to ~~exam~~ phlebotomy room with ~~medical chart~~ requisition

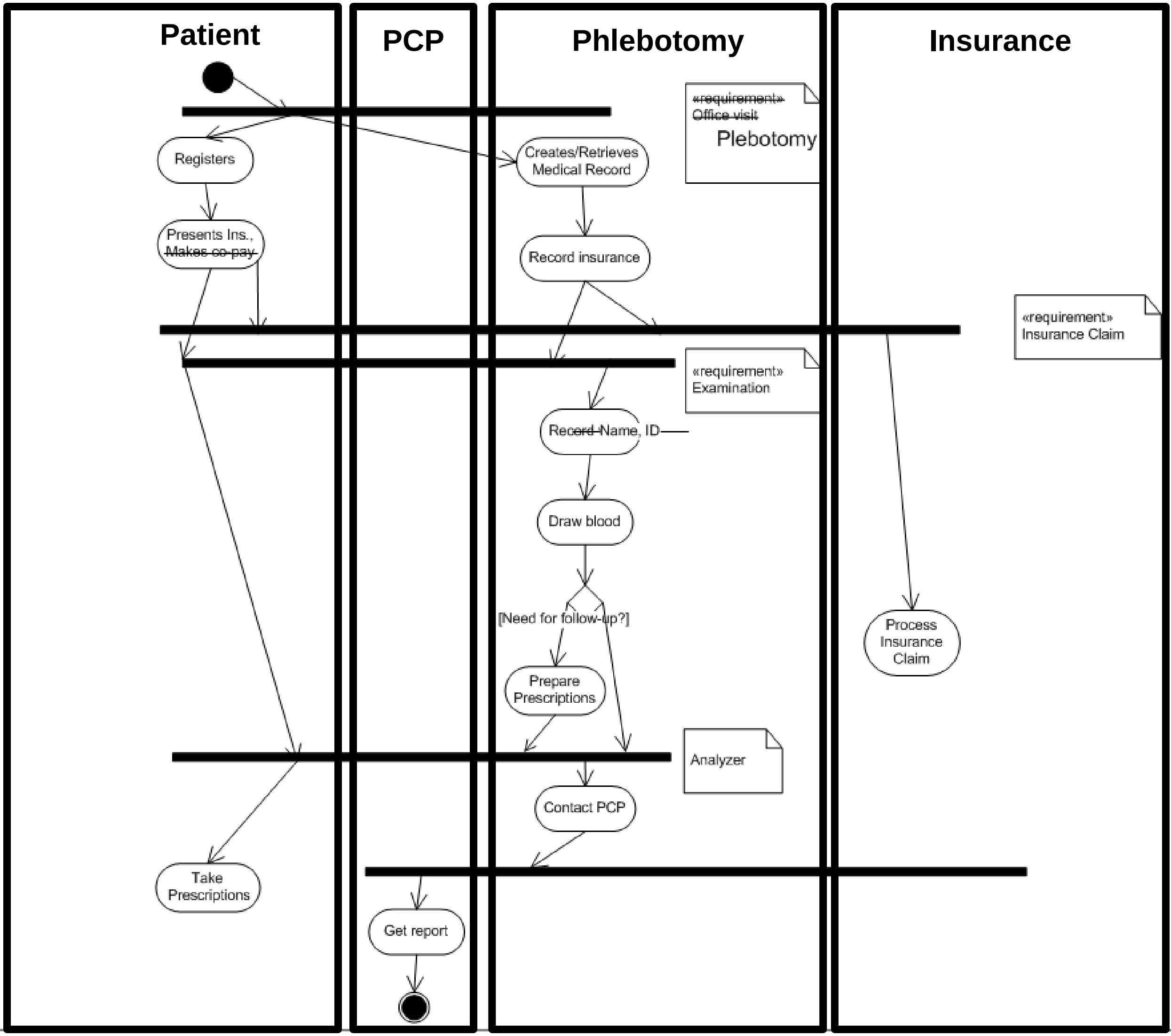
~~Assistant~~ Technician performs preliminaries and records in ~~paper chart~~ system

- ~~weight, vitals, etc.~~ name, id, etc.
- reason for visit
- Takes required blood samples

Lab sends samples to analyzer

Patient takes prescriptions and proceeds to ~~hospital radiology department~~ ENT

Activity - Labs



A Medical Story – Chapter 4 (Cardiologist)

The patient checks in

- presents insurance card
- makes co-payment
- ~~taken to exam room with medical chart~~ Presents labs report

Assistant performs preliminaries and records in paper/electronic chart

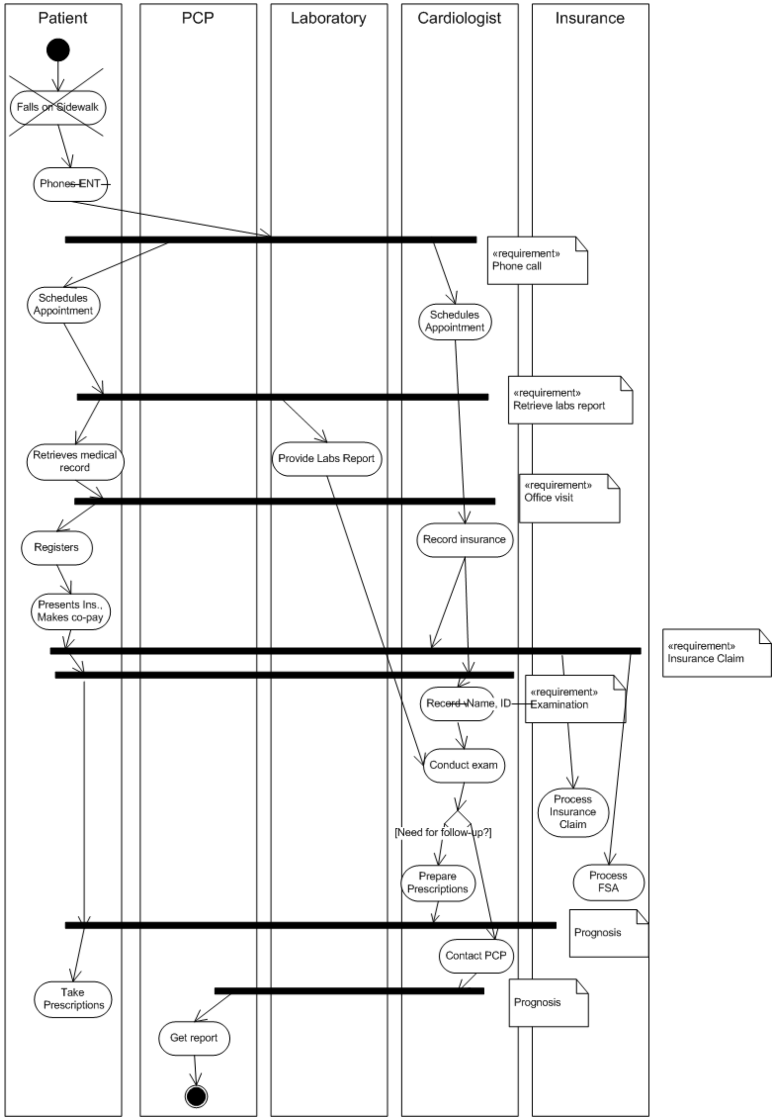
- weight, vitals, etc.
- reason for visit

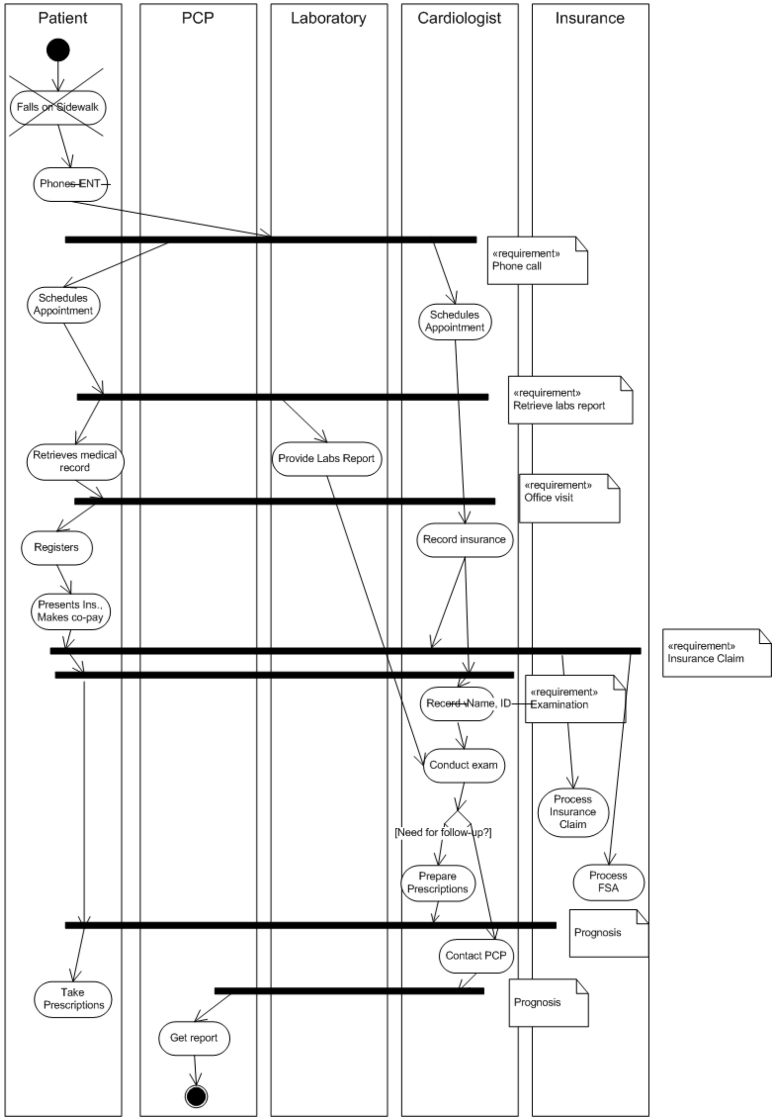
Doctor examines patient

- Conducts general examination
- ~~prescribes~~ reviews labs
- sends patient for echo cardiogram
- Records report in chart

Moderate cardiovascular risk. Further treatment required

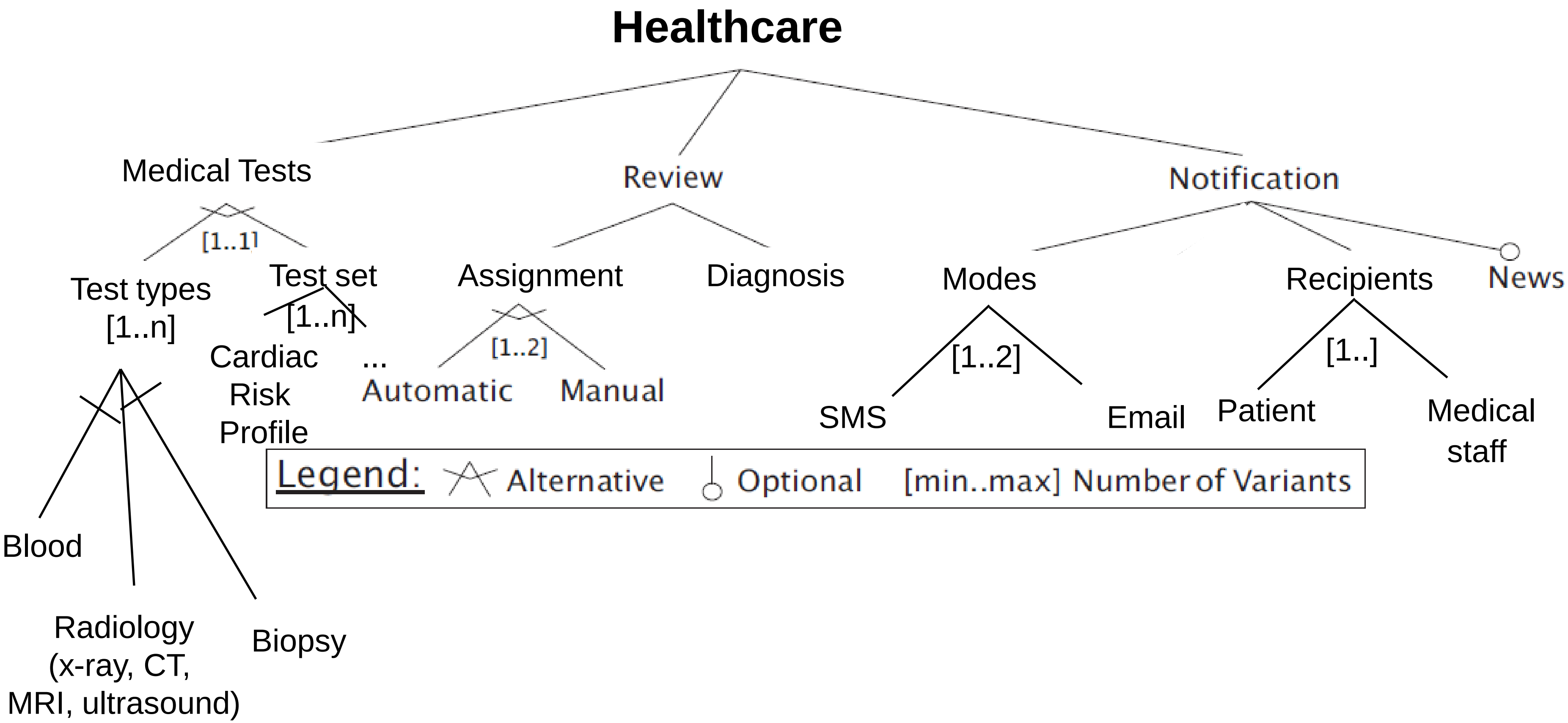
Activity - ENT



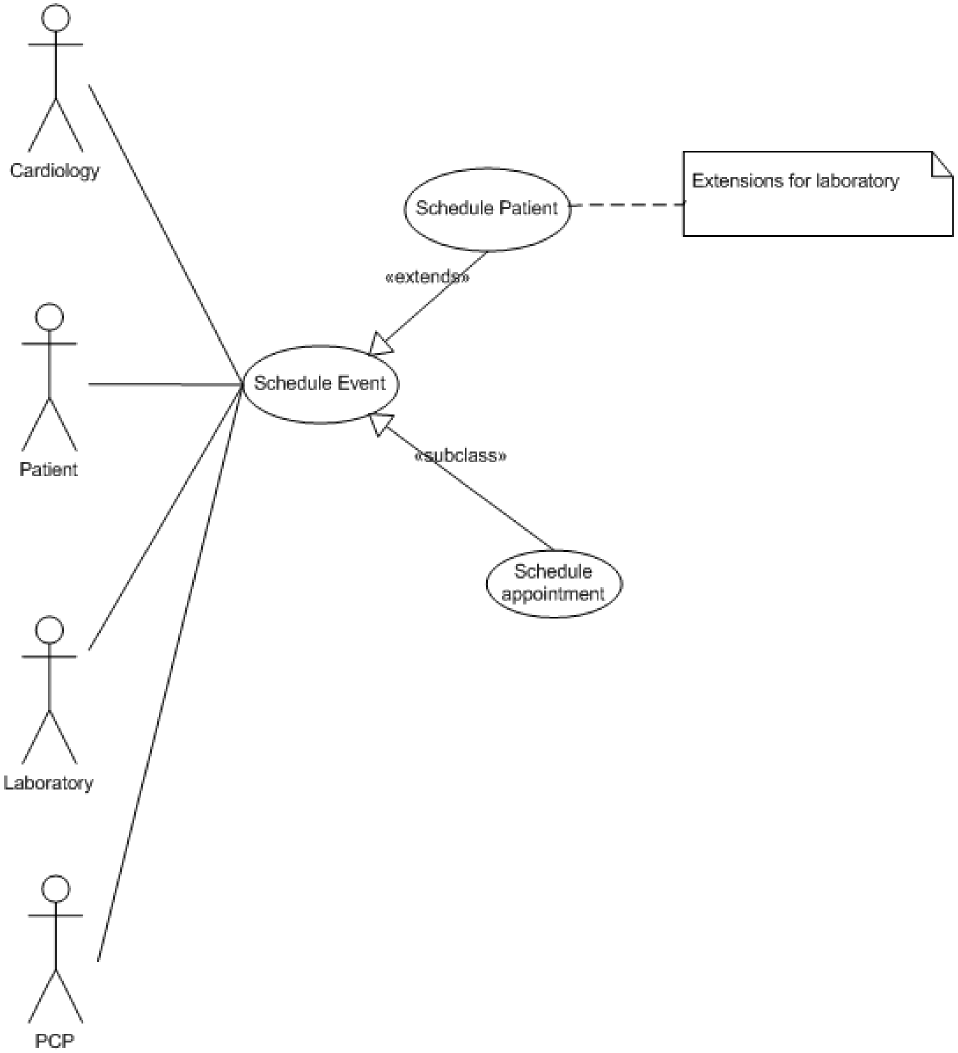


Tasks	Patterns
Scheduling (PCP & Cardio) Registering Maintaining Medical Record (all)	Processing Insurance (all) Examination (all) Managing Medical Record? Report referral (labs and Cardio)

Example Feature Model



Schedule Event Use Case



Scenario for Laboratory Schedule Patient

Actor	System
Patient requests exam	System places patient in queue for that exam category (variations for pediatric, neo-natal, etc.). Schedules technician work flow.
Option: Lab performs pre-exam activity	Updates record with pre-exam results
Technician performs exam	Updates record with exam results
	Optional: Schedule clinical consultation
Optional: Patient provides time	Send reminder to clinician to follow up with PCP. Optional: consultation.
Analysis accepts reminder. Submits analysis	Forwards report
	Optional: Schedule next reminder to patient

Pattern: Managing Medical Record

Actors – PCP, Cardiology, Labs, Medical Record system, medical information exchange

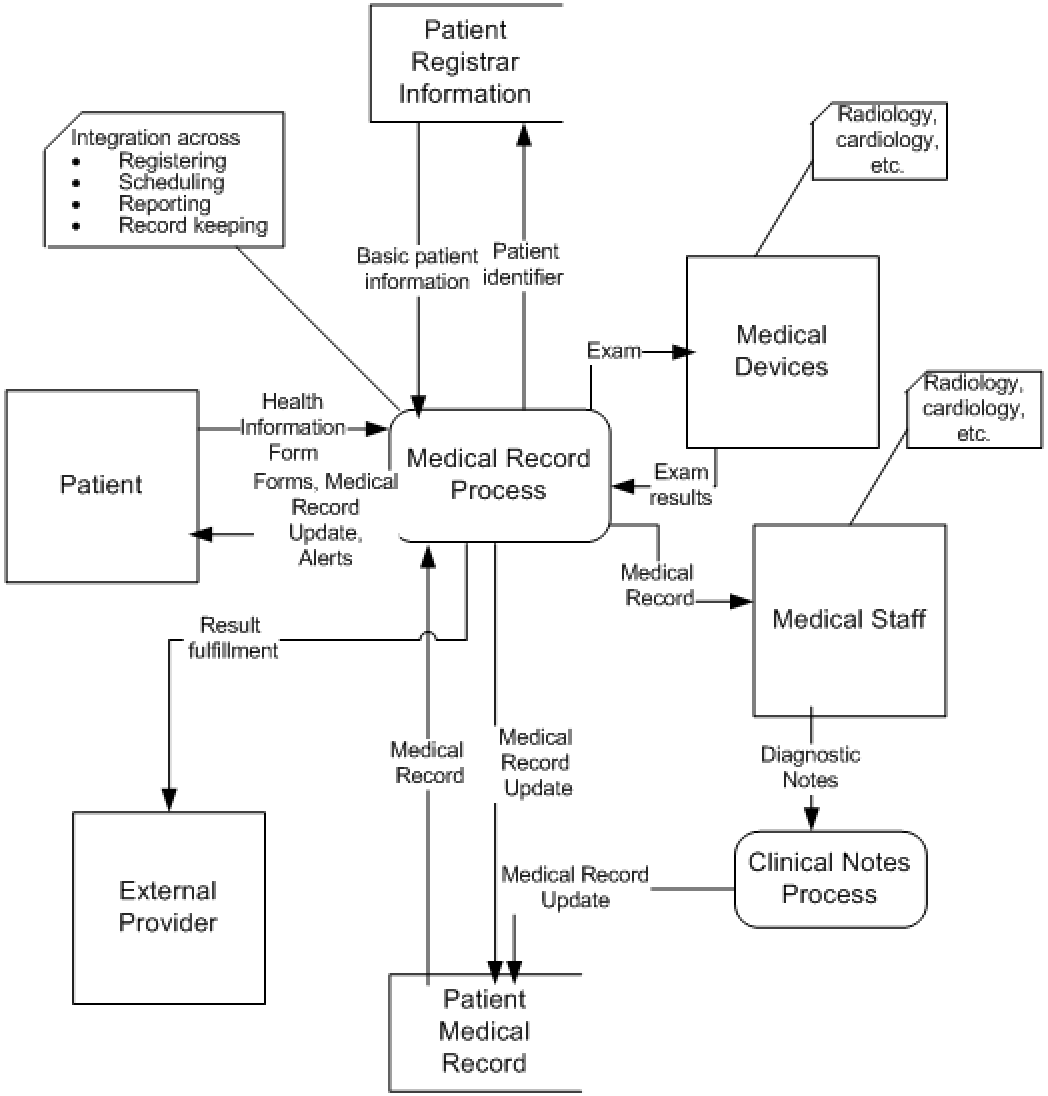
Integration across use cases:

- Registering
- Scheduling
- Reporting
- Record keeping

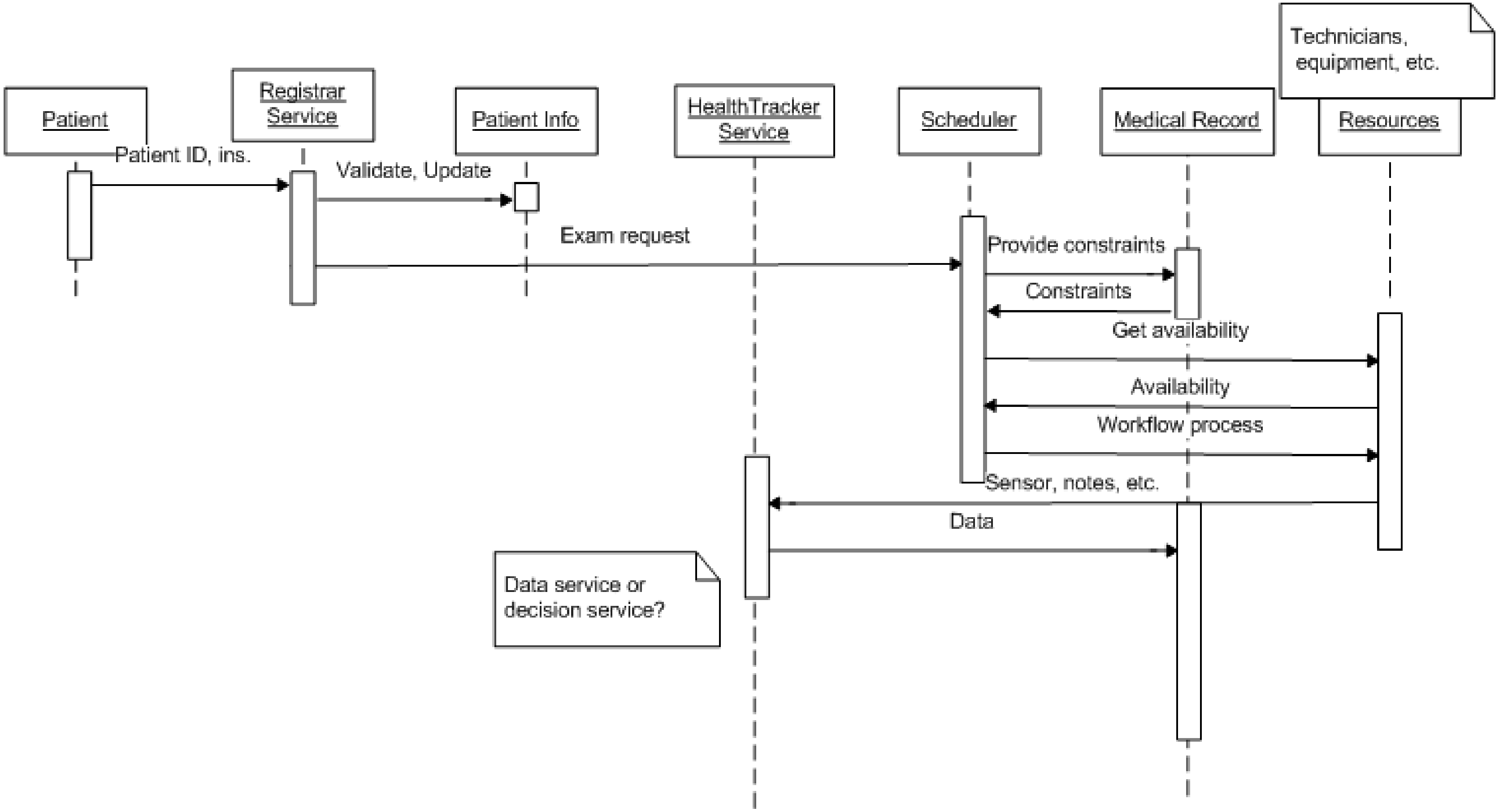
Integration may be modeled as linking use cases with extensions for variations depending on organizational constraints

Variations for medical practice area: Radiology, cardiology, etc.

Medical Record Management Context



Medical Record Workflow Sequence



Characteristics of Systematic Reuse

Not about extracting a legacy component and wrapping as a service for use in a single, new system

Systematic reuse is about:

- Creating a family of products, or software product line, whose members vary while sharing many common features
- Identifying and differentiating those features that remain constant across those products versus those that vary
- Defining service functionality and implementation characteristics within context of targeted systems
- Building variations into services and select among the variants to create a unique product
- Examples
  - Medical record management systems
  - Scheduling systems

What is a Product Line

A set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Aspects	**SPL definition element**	**Definition of a service-oriented product line**
Scope	A set of software-intensive systems	Medical information management systems
Source of variation	that share a common, managed set of features	Authentication services, medical treatment record services, physician directed services, patient management services, billing record services
Application	satisfying the specific needs of a particular market segment or mission	Electronic medical record services for the healthcare industry including hospitals, clinics, medical offices, patient home (self-directed)
Compositional elements	and that are developed from a common set of core assets	Services, scope definition, feature model, SOA-based product line architecture, etc.
Technical approach	in a prescribed way	Architecture and production plan to guide building of applications using SOA infrastructure.

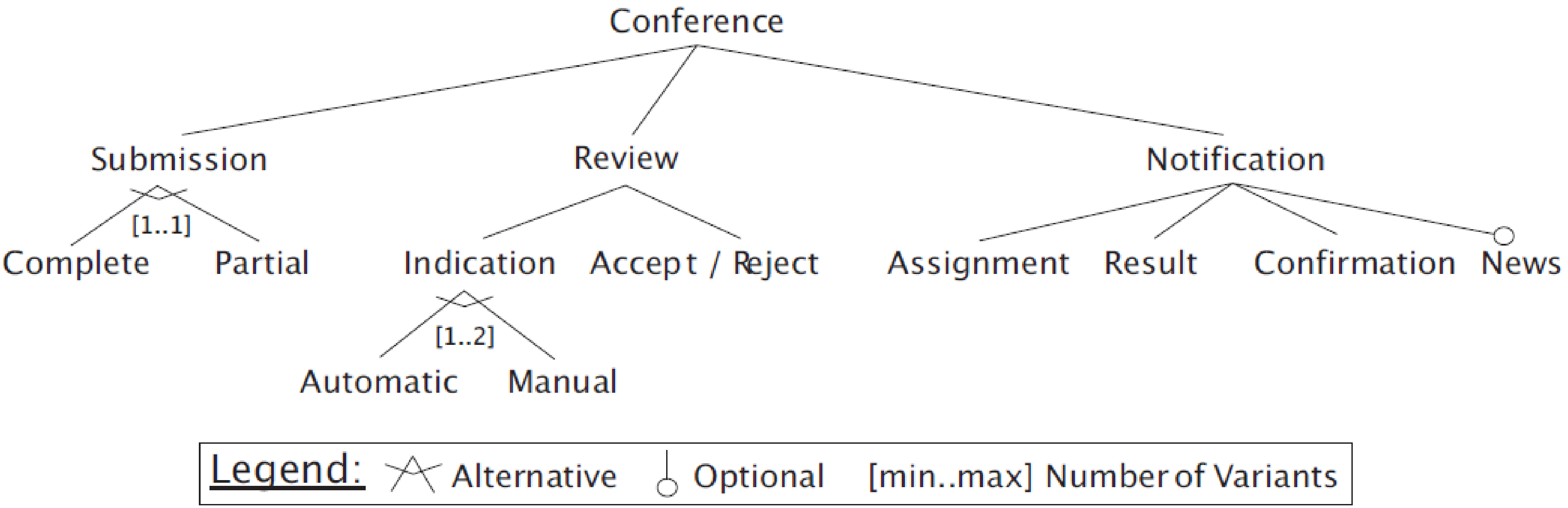
Evolutionary Approach

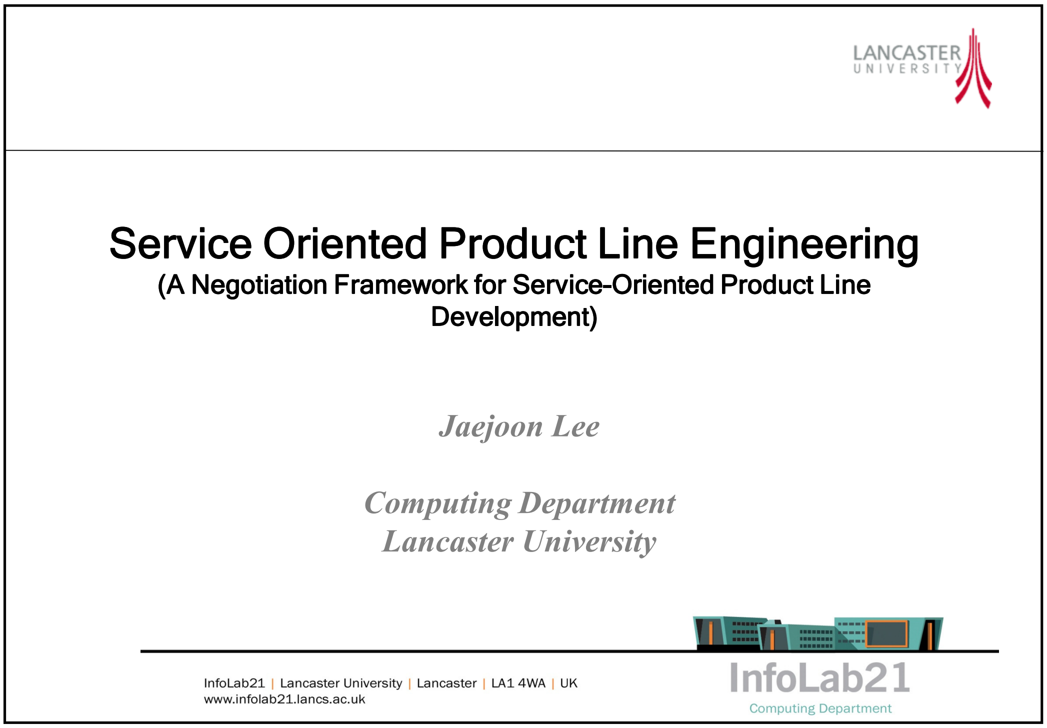
Understand potential scope of applicability of the core asset base of services

Develop the core asset base in stages while planning from the beginning to develop a product line.

- Develop part of the core asset base, including the architecture and some of the services for multiple applications
- Develop one or more applications or products.
- Develop part of the rest of the core asset base.
- Develop more products.
- Evolve more of the core asset base.

Feature Model





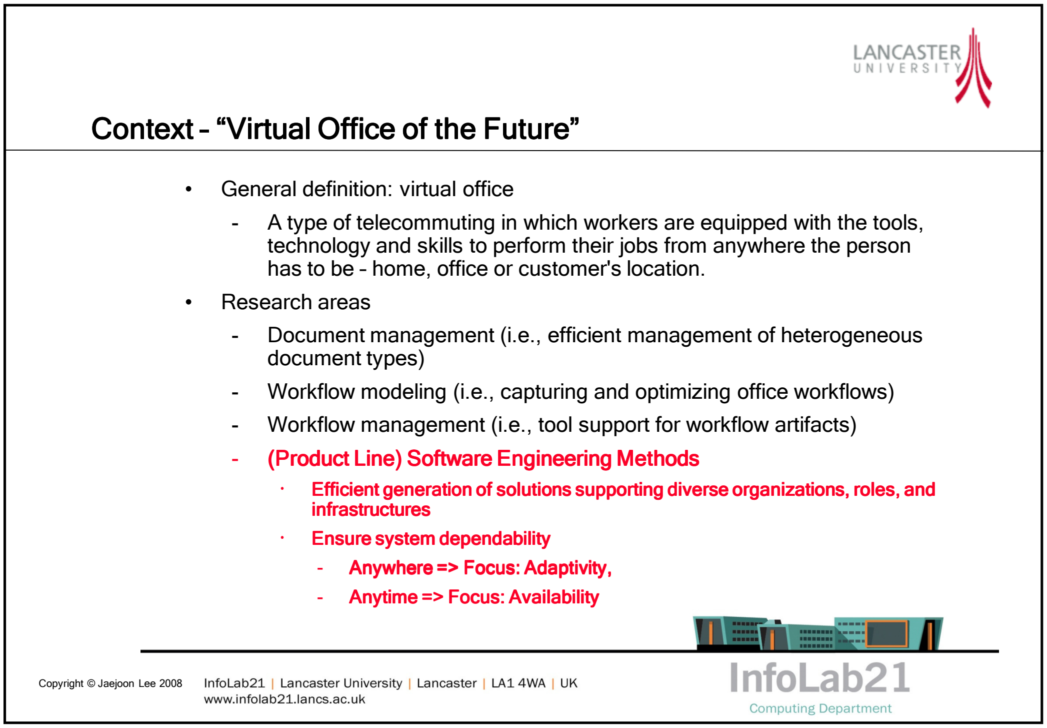
LANCASTER UNIVERSITY

Service Oriented Product Line Engineering

(A Negotiation Framework for Service-Oriented Product Line Development)

Jaejoon Lee

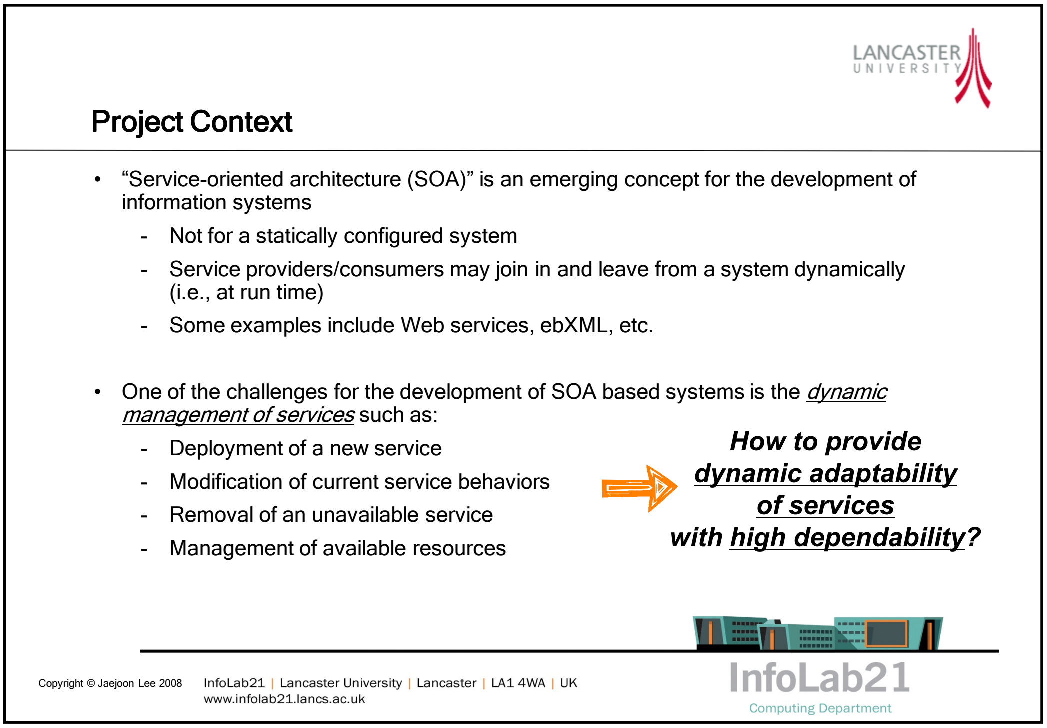
Computing Department
Lancaster University



LANCASTER UNIVERSITY

Context – “Virtual Office of the Future”

- General definition: virtual office
  - A type of telecommuting in which workers are equipped with the tools, technology and skills to perform their jobs from anywhere the person has to be - home, office or customer's location.
- Research areas
  - Document management (i.e., efficient management of heterogeneous document types)
  - Workflow modeling (i.e., capturing and optimizing office workflows)
  - Workflow management (i.e., tool support for workflow artifacts)
  - (Product Line) Software Engineering Methods
    - Efficient generation of solutions supporting diverse organizations, roles, and infrastructures
    - Ensure system dependability
      - Anywhere => Focus: Adaptivity,
      - Anytime => Focus: Availability



LANCASTER UNIVERSITY

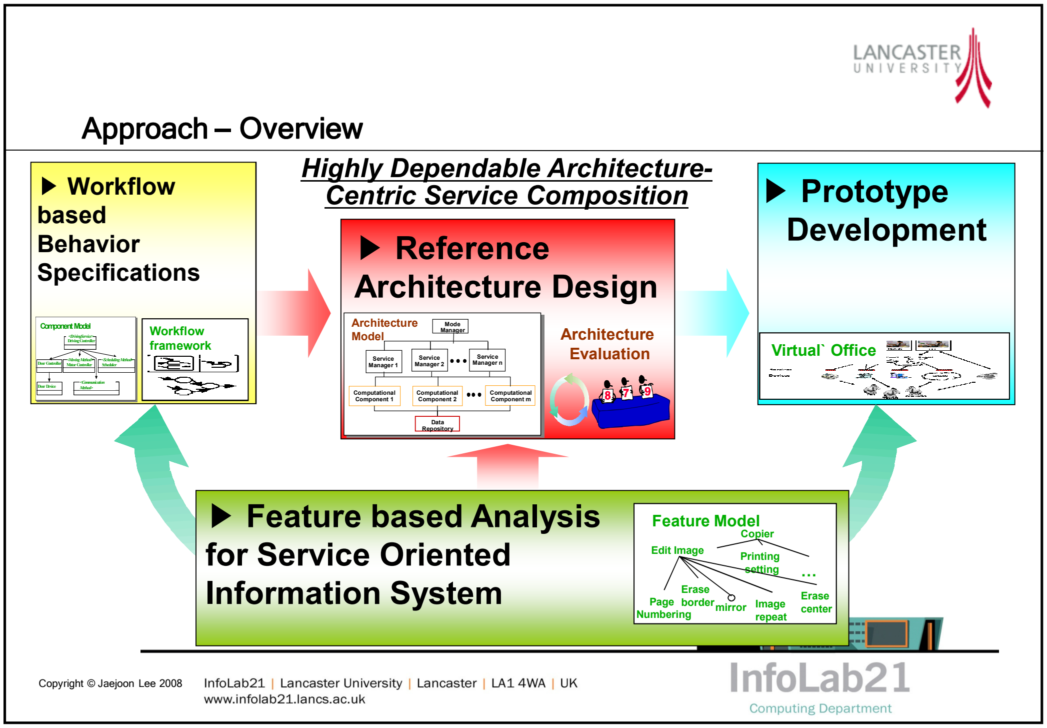
Project Context

- "Service-oriented architecture (SOA)" is an emerging concept for the development of information systems
  - Not for a statically configured system
  - Service providers/consumers may join in and leave from a system dynamically (i.e., at run time)
  - Some examples include Web services, ebXML, etc.
- One of the challenges for the development of SOA based systems is the *dynamic management of services* such as:
  - Deployment of a new service
  - Modification of current service behaviors
  - Removal of an unavailable service
  - Management of available resources

How to provide dynamic adaptability of services with high dependability?

InfoLab21 | Lancaster University | Lancaster | LA1 4WA | UK
www.infolab21.lancs.ac.uk

InfoLab21
Computing Department



LANCASTER UNIVERSITY

Approach – Overview

Highly Dependable Architecture-Centric Service Composition

▶ Workflow based Behavior Specifications

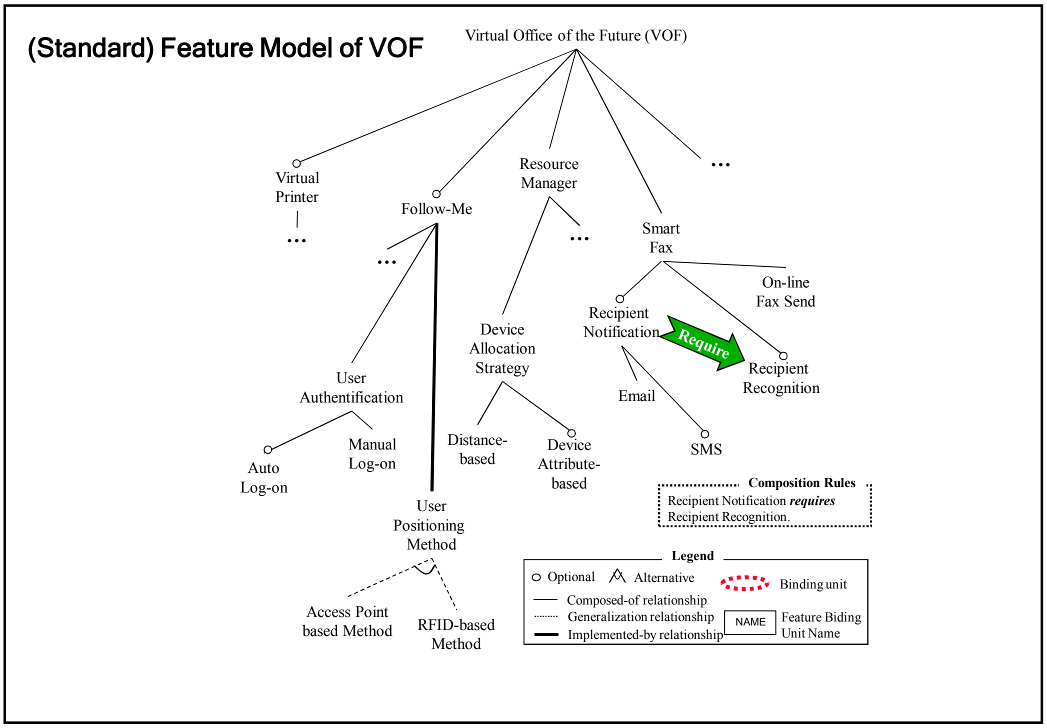
▶ Reference Architecture Design

▶ Prototype Development

▶ Feature based Analysis for Service Oriented Information System

InfoLab21 | Lancaster University | Lancaster | LA1 4WA | UK
www.infolab21.lancs.ac.uk

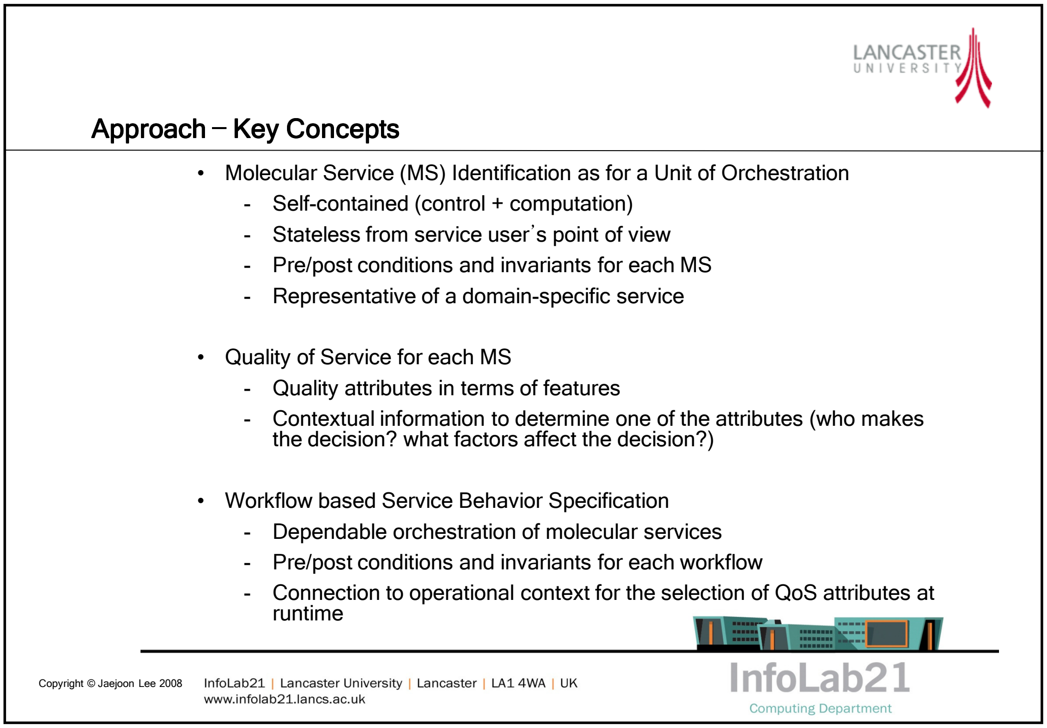
InfoLab21
Computing Department



(Standard) Feature Model of VOF

Composition Rules

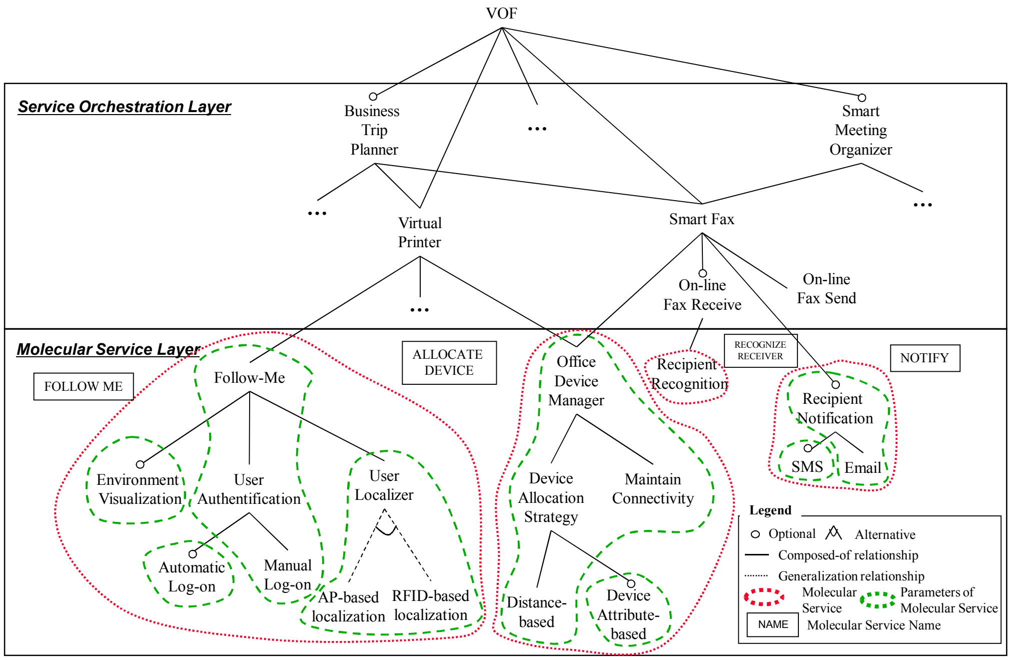
Recipient Notification ***requires*** Recipient Recognition.



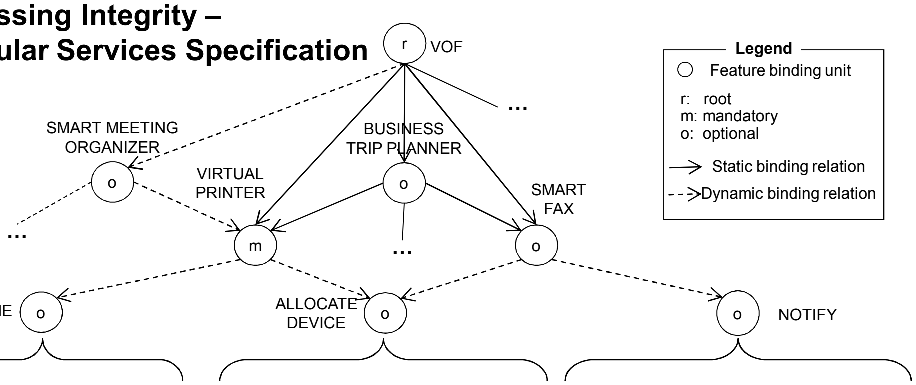
Approach – Key Concepts

- Molecular Service (MS) Identification as for a Unit of Orchestration
  - Self-contained (control + computation)
  - Stateless from service user's point of view
  - Pre/post conditions and invariants for each MS
  - Representative of a domain-specific service
- Quality of Service for each MS
  - Quality attributes in terms of features
  - Contextual information to determine one of the attributes (who makes the decision? what factors affect the decision?)
- Workflow based Service Behavior Specification
  - Dependable orchestration of molecular services
  - Pre/post conditions and invariants for each workflow
  - Connection to operational context for the selection of QoS attributes at runtime

Molecular Service Identification



Addressing Integrity –
Molecular Services Specification



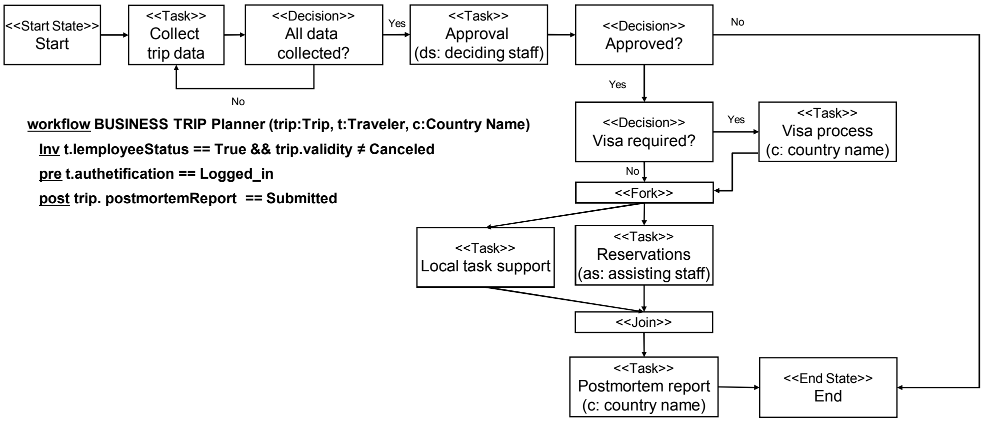
```
molecular service FOLLOW ME (user User)
 inv user.employmentStatus == true
 pre user.authentification == logged_in
 post none;
   option Environment Visualization
     binding time run time
     pre user.device == desktop ∨ notebook
     post none;
  option Automatic Log-on
    binding time run time
     pre user.rank == director ∨ manager and
        RFID bases user location method == available
     post user.access == granted ∨ rejected;
```

```
molecular service ALLOCATE DEVICE (user User)
  inv user. employmentStatus == true
  pre user.authentification == logged_in
  post user.device_allocation == success ∨ failure ;
    option Attribute based Device Allocation
      binding time installation time
      pre user.rank == director ∨ manager
      post none;
```

```
molecular service NOTIFY (sender User, receiver User)
   inv sender. employmentStatus == true
   pre sender.authentification == logged_in
        receiver.email ≠ null
   post none;
     option SMS
       binding time run time
       pre sender.cell_phone_number ≠ null  and
          sender.message ≠ null and
          receiver.cell_phone_number ≠ null
       post sender.message == null
```

Workflow Specification: Dependable Orchestration of Molecular Services

- Example of Business Trip Planner



<<Start State>> Start
<<Task>> Collect trip data
<<Decision>> All data collected?
Yes
No
<<Task>> Approval (ds: deciding staff)
<<Decision>> Approved?
No
Yes
<<Decision>> Visa required?
Yes
<<Task>> Visa process (c: country name)
No
<<Fork>>
<<Task>> Local task support
<<Task>> Reservations (as: assisting staff)
<<Join>>
<<Task>> Postmortem report (c: country name)
<<End State>> End

<u>workflow</u> BUSINESS TRIP Planner (trip:Trip, t:Traveler, c:Country Name)

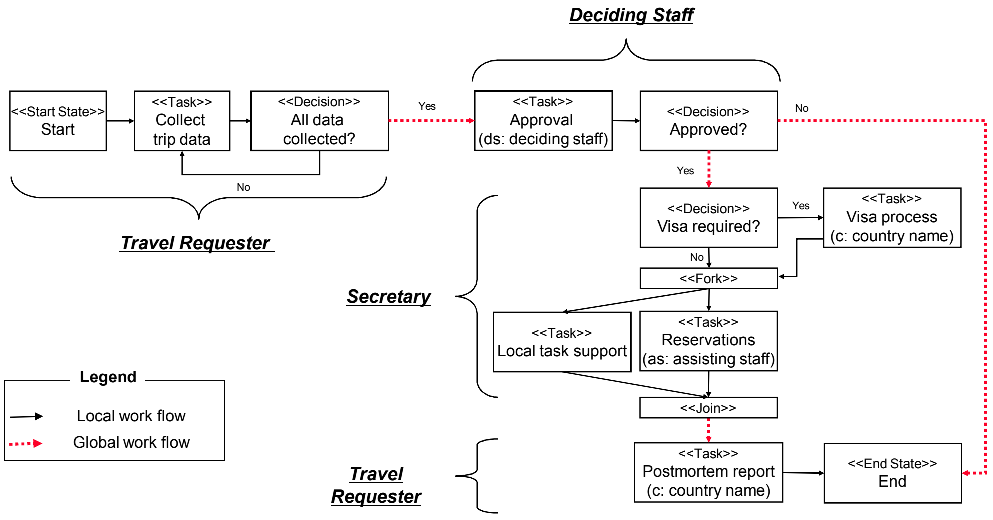
<u>Inv</u> t.lemployeeStatus == True && trip.validity ≠ Canceled

<u>pre</u> t.authetification == Logged_in

<u>post</u> trip. postmortemReport == Submitted

Identification of Localities of Tasks from a WF Specification

- Example of Business Trip Planner



Deciding Staff
<<Start State>> Start
<<Task>> Collect trip data
<<Decision>> All data collected?
Yes
No
<<Task>> Approval (ds: deciding staff)
<<Decision>> Approved?
No
Yes
Travel Requester
<<Decision>> Visa required?
Yes
<<Task>> Visa process (c: country name)
No
<<Fork>>
Secretary
<<Task>> Local task support
<<Task>> Reservations (as: assisting staff)
<<Join>>
<<Task>> Postmortem report (c: country name)
<<End State>> End
Travel Requester
Legend
Local work flow
Global work flow

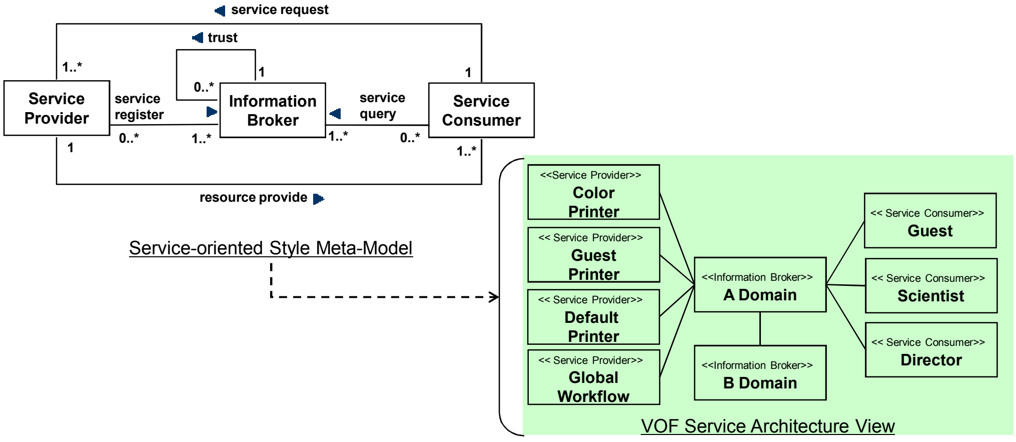
LANCASTER UNIVERSITY

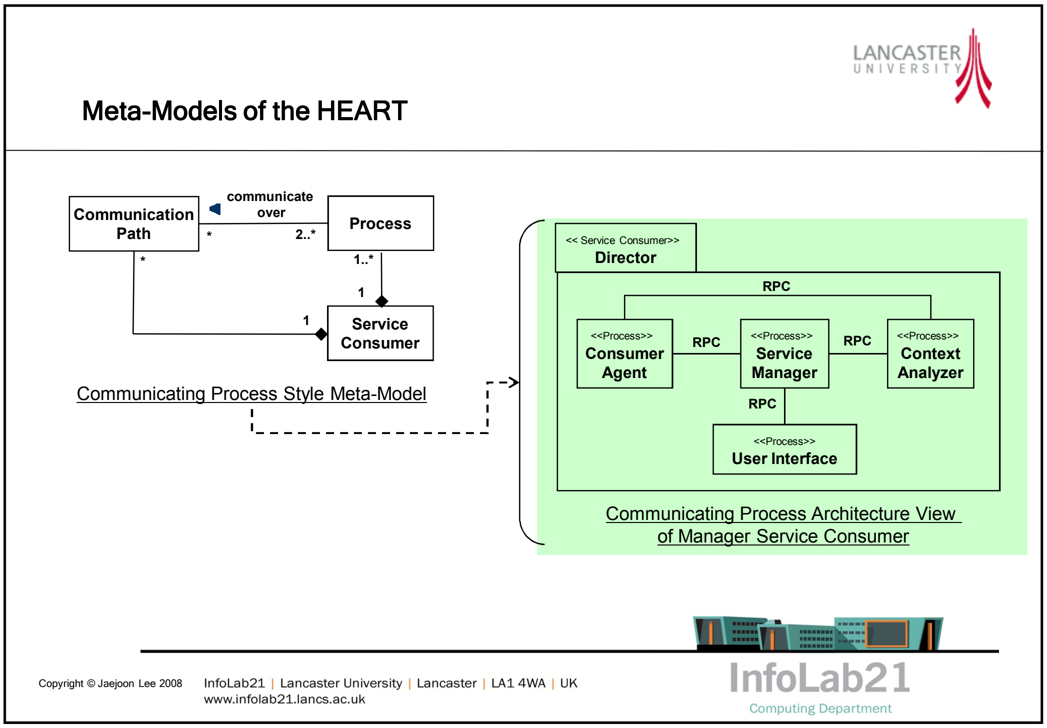
Design Goals – Product Line Architecture

- Service orientation
  - Network based service request, query, and provision
  - Scalability over the Internet
- Context awareness
  - Recognition of current operational/locational context of users
  - Maintain connectivity to service providers
- Product line variations
  - Control of product line variation before/after deployment
  - Product customization for each user
  - Dynamic product reconfiguration to provide context relevant services

LANCASTER UNIVERSITY

Architecture Style – HEART (Heterogeneous-style-based Architecture)

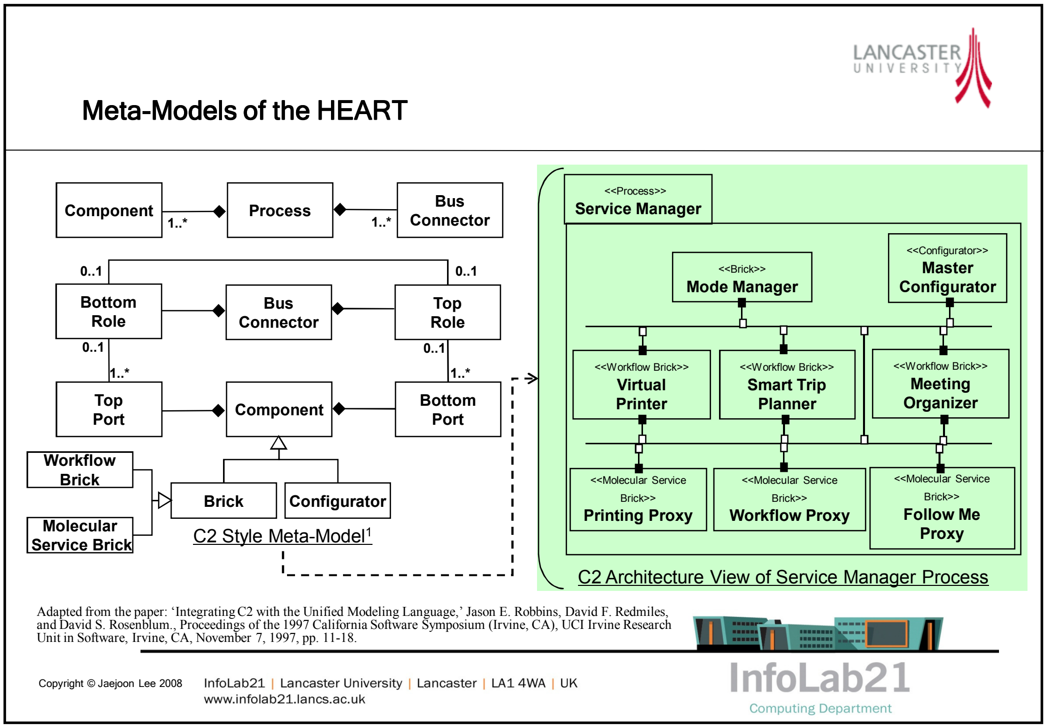

service request
trust
1..*
1
0..*
1
Service Provider
service register
Information Broker
service query
Service Consumer
0..*
1..*
1..*
0..*
1
1..*
resource provide
Service-oriented Style Meta-Model
<<Service Provider>> Color Printer
<< Service Provider>> Guest Printer
<< Service Provider>> Default Printer
<< Service Provider>> Global Workflow
<<Information Broker>> A Domain
<<Information Broker>> B Domain
<< Service Consumer>> Guest
<< Service Consumer>> Scientist
<< Service Consumer>> Director
VOF Service Architecture View



Meta-Models of the HEART

Communicating Process Style Meta-Model

Communicating Process Architecture View of Manager Service Consumer



Meta-Models of the HEART

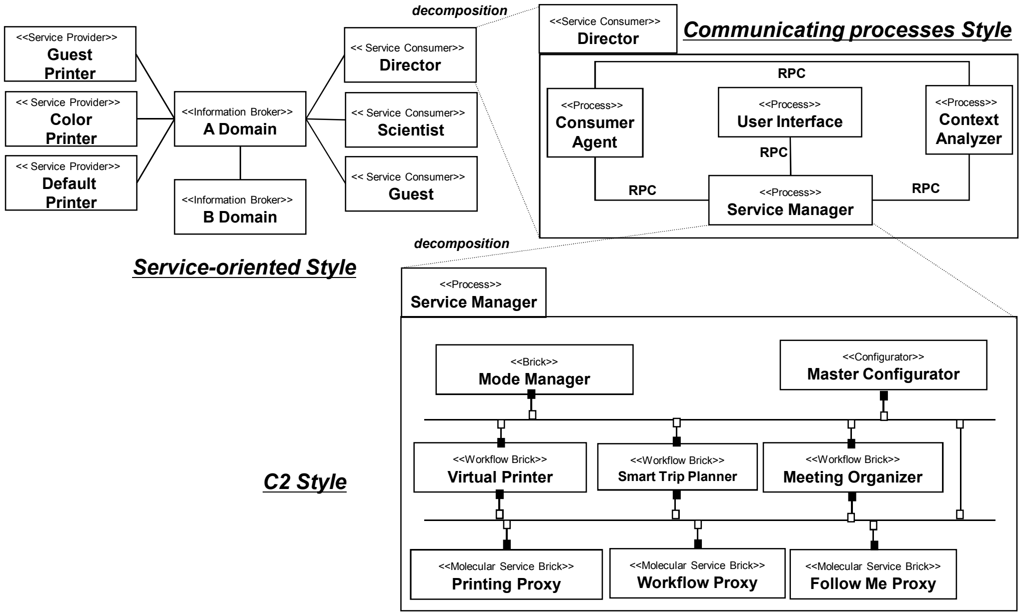
C2 Style Meta-Model[1]

C2 Architecture View of Service Manager Process

Adapted from the paper: 'Integrating C2 with the Unified Modeling Language,' Jason E. Robbins, David F. Redmiles, and David S. Rosenblum., Proceedings of the 1997 California Software Symposium (Irvine, CA), UCI Irvine Research Unit in Software, Irvine, CA, November 7, 1997, pp. 11-18.

HEART (Heterogeneous Style based Architecture) Model:

A multiple architecture style based solution for developing core assets of SO systems



decomposition

<<Service Provider>> Guest Printer

<< Service Provider>> Color Printer

<< Service Provider>> Default Printer

<<Information Broker>> A Domain

<<Information Broker>> B Domain

<< Service Consumer>> Director

<< Service Consumer>> Scientist

<< Service Consumer>> Guest

Service-oriented Style

<<Service Consumer>> Director

Communicating processes Style

RPC

<<Process>> Consumer Agent

<<Process>> User Interface

<<Process>> Context Analyzer

RPC

RPC

RPC

<<Process>> Service Manager

decomposition

<<Process>> Service Manager

<<Brick>> Mode Manager

<<Configurator>> Master Configurator

<<Workflow Brick>> Virtual Printer

<<Workflow Brick>> Smart Trip Planner

<<Workflow Brick>> Meeting Organizer

<<Molecular Service Brick>> Printing Proxy

<<Molecular Service Brick>> Workflow Proxy

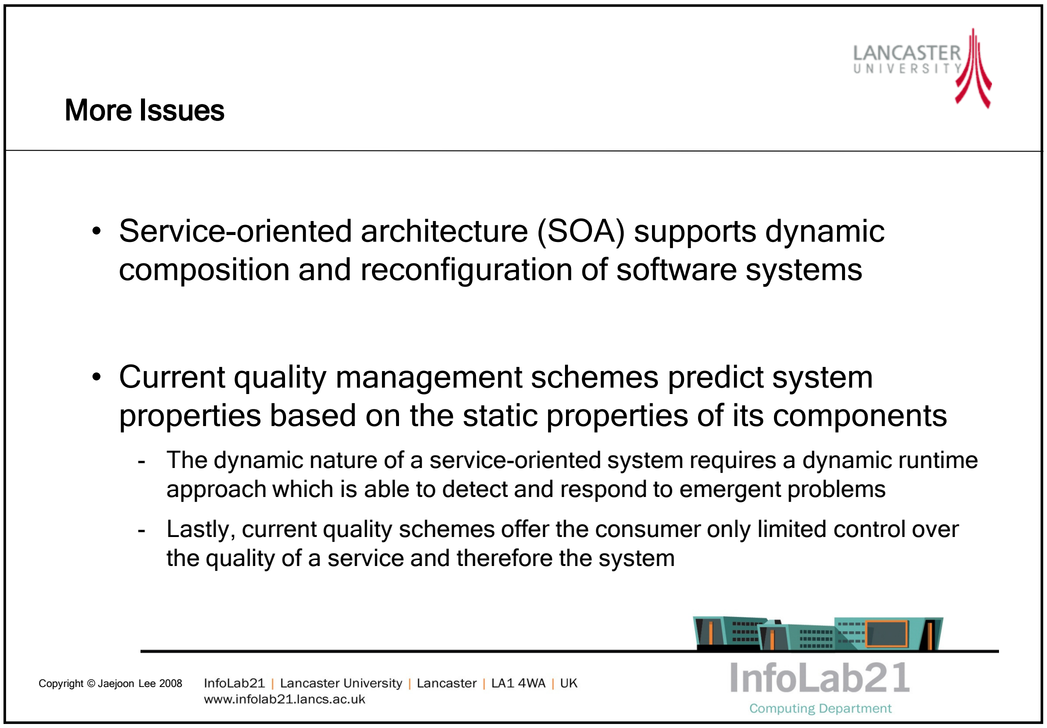
<<Molecular Service Brick>> Follow Me Proxy

C2 Style

LANCASTER UNIVERSITY

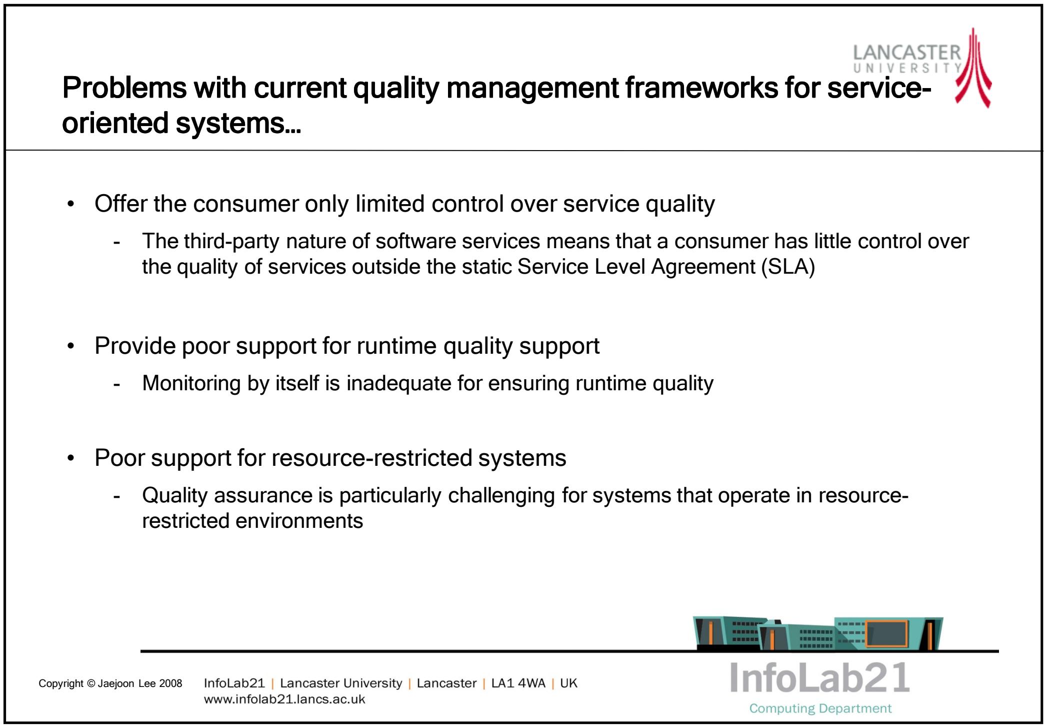
Summary of the Approach

- Feature based identification of molecular services and their quality attributes
- Extension of workflow specifications with pre/post conditions and invariants for dependable service orchestration
- Architecture model for the systematic integration of multidisciplinary design paradigms: dependability, adaptivity (dynamic variations), and service orientation
- Prototype development to demonstrate the feasibility of proposed approach



More Issues

- Service-oriented architecture (SOA) supports dynamic composition and reconfiguration of software systems
- Current quality management schemes predict system properties based on the static properties of its components
  - The dynamic nature of a service-oriented system requires a dynamic runtime approach which is able to detect and respond to emergent problems
  - Lastly, current quality schemes offer the consumer only limited control over the quality of a service and therefore the system

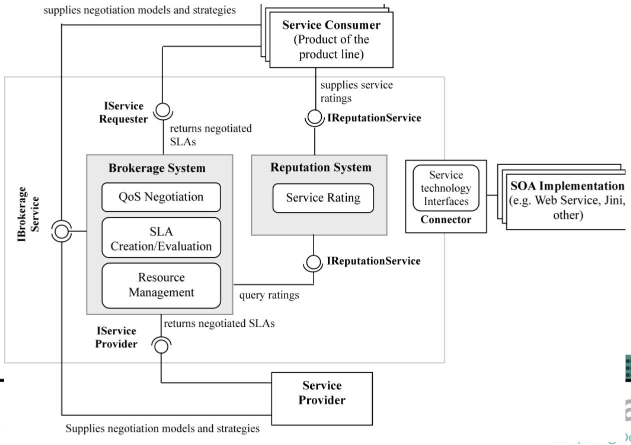


Problems with current quality management frameworks for service-oriented systems…

- Offer the consumer only limited control over service quality
  - The third-party nature of software services means that a consumer has little control over the quality of services outside the static Service Level Agreement (SLA)
- Provide poor support for runtime quality support
  - Monitoring by itself is inadequate for ensuring runtime quality
- Poor support for resource-restricted systems
  - Quality assurance is particularly challenging for systems that operate in resource-restricted environments

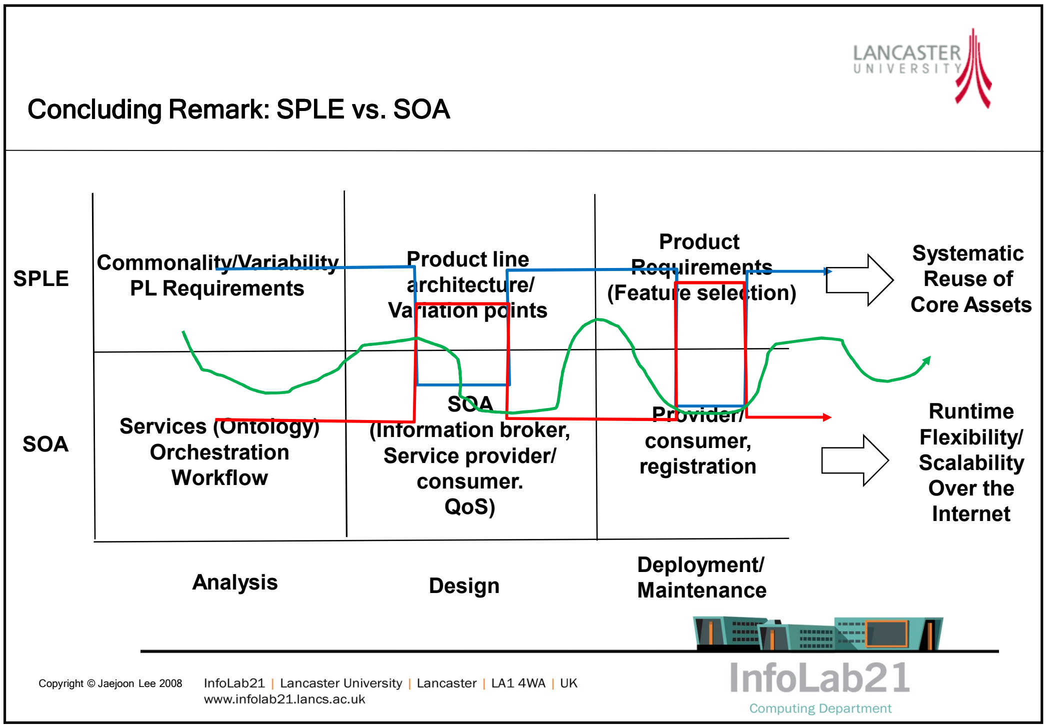
Negotiation Framework

- Provides a structural framework for:
  - Integrating different methods of negotiation and provider reputation rating
  - Supporting the requirements of automated service negotiation and renegotiation in SOA



Current Status

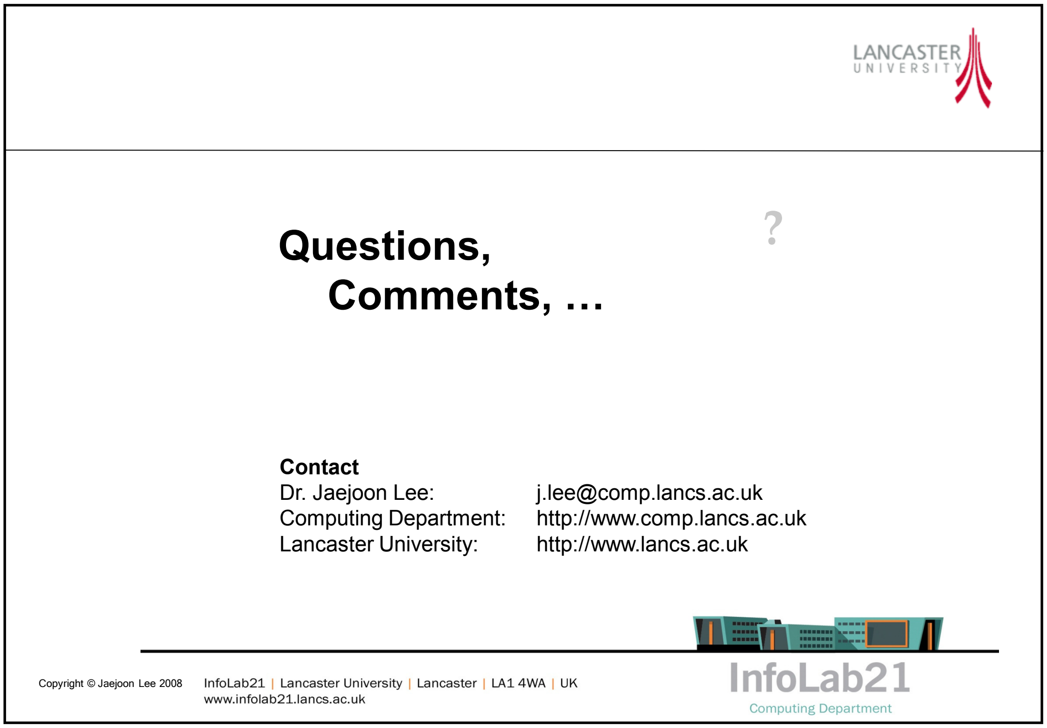
- Effective runtime quality assurance must combine monitoring with effective recovery and self-management strategies
- A consumer-centered quality assurance framework:
  - Enables consumers to negotiate service agreements which are closer to their requirements, and compensates providers accordingly
  - Allows consumers to specify quality-weighted services and to associate these with consumer strategies
- We are currently investigating improvements to the framework to support dynamic strategies, forecasting and runtime quality more efficiently in resource-constrained system environments



LANCASTER UNIVERSITY

Concluding Remark: SPLE vs. SOA

	Analysis	Design	Deployment/ Maintenance	
SPLE	Commonality/Variability PL Requirements	Product line architecture/ Variation points	Product Requirements (Feature selection)	Systematic Reuse of Core Assets
SOA	Services (Ontology) Orchestration Workflow	SOA (Information broker, Service provider/ consumer. QoS)	Provider/ consumer, registration	Runtime Flexibility/ Scalability Over the Internet



LANCASTER UNIVERSITY

**Questions,
Comments, …**

?

Contact

Dr. Jaejoon Lee:	j.lee@comp.lancs.ac.uk
Computing Department:	http://www.comp.lancs.ac.uk
Lancaster University:	http://www.lancs.ac.uk

Managing SOA System Variation through Business Process Lines and Process Oriented Development

Nicola Boffoli, Marta Cimitile, Fabrizio Maria Maggi, Giuseppe Visaggio
Department of Informatics - University of Bari - Via E. Orabona 4 - 70126 - Bari - Italy
{boffoli, cimitile, maggi, visaggio}@di.uniba.it

Abstract

Software Product Lines (SPL) and Service-Oriented Architectures (SOA) are two emerging approaches to the software development currently receiving great attention both in research and in practice.

Our work suggests an approach to transfer the main peculiarities of the SPL (i.e. asset reuse and variation mechanisms) to the SOA systems development, in order to realize a SOA systems line. In this way we provide a method to easily adapt a SOA application to different customer needs in changeable environments.

All this is realized using the Business Process Lines (BPL) concept together with the Process Oriented Development (POD) paradigm. A BPL realizes process models suitable to different customers or market segments needs. The POD paradigm allows to transform a process model into a SOA system.

Finally we show an application of our proposal in a research project that involve several industrial and academic organization. In the project a set of BPL is realized and implemented using the MIT process handbook.

1. Introduction

Software Product Lines (SPL) [1] and Service Oriented Architectures (SOA) [2] aim to develop software systems through two common perspectives: software reuse and flexibility [3]. Using these approaches, enterprises can implement software systems for different customers reusing software resources rather than developing the same software capabilities again. In this way they gain in productivity, software quality and time to market.

A SPL is a set of software-intensive systems sharing common features. In particular a SPL aims to satisfy the specific needs of a market segment using a common set of core assets in a prescribed way.

In general, the SPL paradigm is characterized by two different concepts:

- <u>*Asset Reuse*</u>: management of the “Core Asset” i.e. collection, organization and systematic refinement of the invariant or variant assets representing respectively the SPL *Commonality* and *Variability*.
- <u>*Variation Mechanisms*</u>: automatic building of the products based on the systematic reuse of the “Core Assets”. Each asset is a software component with fixed specifications allowing to:
  - *Configure* the products through asset integration procedures;
  - *Specialize* the assets through the specification of their parametric parts.

A SOA is a software architecture able to orchestrate web services to guarantee the integration of heterogeneous systems in a business process. A SOA is made up of components and interconnections stressing interoperability and location transparency.

Early research works, concerning the comparison between SPL and SOA, are just appearing in the software engineering research community [4]. In particular a crucial research question is: how can SOA systems benefit from SPL good practices (i.e. reuse and variation management approaches)? Or in other words *what* is the *core asset* and *how* can we implement the *variation mechanisms* in the SOA system context?

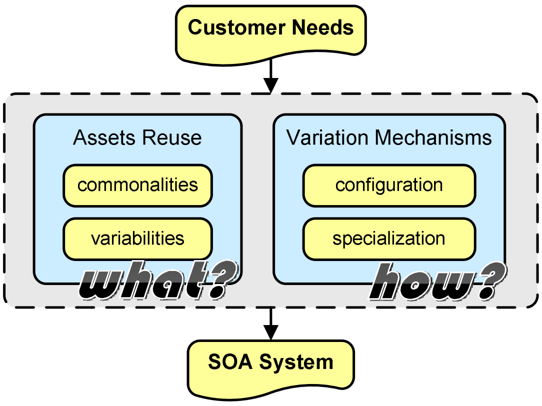


Figure 1: SPL concepts in SOA context

In this paper we answer these questions through the approach illustrated in Figure 2. In particular, we propose transferring advantages from SPL to SOA operating on business processes and using two specific instruments:

1. <u>*Business Process Line*</u> *(BPL)*. A BPL according to the SPL practices is able to model an appropriate business process, *process variant*, suitable for specific customer needs.

2. *Process Oriented Development (POD)*. POD is able to transform a process variant into an executable SOA system through successive transformations aimed at making the process model understandable by an execution engine.

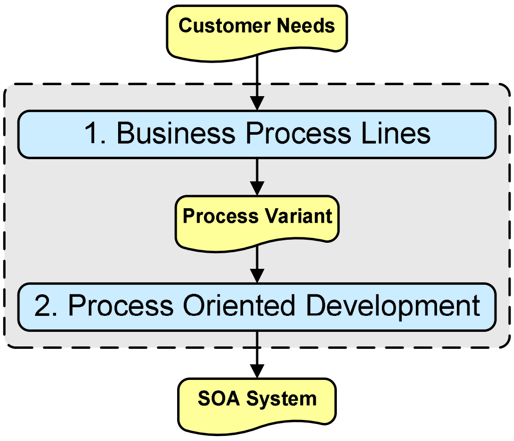


Figure 2: Schema of the Proposal

The resulting SOA system line automates the underlying business processes; so, if we adapt the business processes to new customer needs using the underlying BPL and then we generate from it the corresponding SOA system, it will result in its turn suitable to the specific customer requirements.

Moreover since the SOA system results from the translation of the former process models, it will be in compliance with the underlying business processes without misalignments between models and their implementation.

The remainder of the paper is structured as follows: section 2 presents the overview of the proposal and discusses in detail the BPL and POD approaches; in section 3 is described the application of the approach in a research project; section 4 completes the paper providing some conclusive insights and final remarks showing prospective works.

2. Proposed Approach

The proposed approach is synthesized in Figure 2. It consists in two main phases using respectively the Business Process Lines concept and the Process Oriented Development paradigm.

In the first phase, starting from the current customer needs, a BPL allows to realize a process variant specific for the given requirements. In the second phase, starting from this model, POD allows to automate this model and transforms it into a SOA system. These phases are detailed in the following paragraphs.

2.1 Business Process Lines (BPL)

In [5, 6, 7] the authors transfer the main SPL concept to the business processes field.

According to these works a BPL could be considered a set of similar business processes sharing a common part (commonality) and characterized by a variant part (variability) depending on the specific context where the process will be applied (more details in [8]). In particular a BPL works integrating a set of process assets, i.e. atomic reusable parts of a business process (one or more activities with their IN/OUT). Commonality is a set of invariant assets and variability is a set of variant assets selected according to a fixed context profile.

Commonality and variability are then integrated in order to obtain a process variant to be applied in the given context. The assets integration rules are driven by their IN/OUT artifacts allowing to establish the succession of the process assets: the outputs of the previous asset are the inputs of the successive one.

The details of this phase are represented in Figure3.

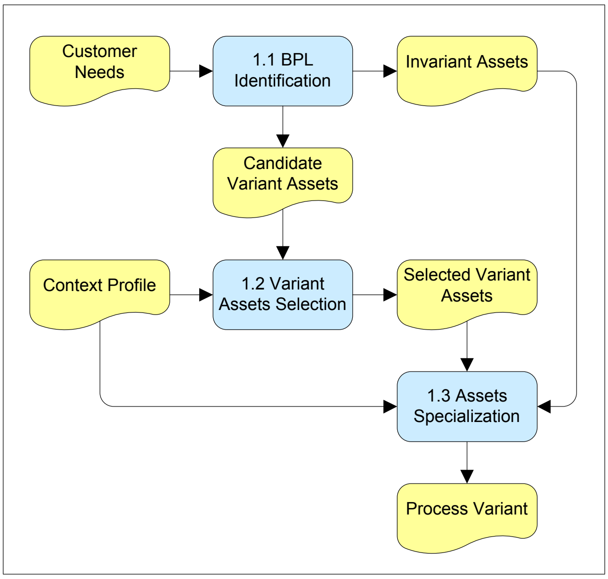


Figure 3: Business Process Lines Approach

When a BPL is selected the invariant assets and all the candidate variant assets are specified. The BPL is selected on the basis of the customer specific needs: for example if we are interested to the process “Selling” we will use a “Selling BPL”. The invariant assets of the “Selling BPL” could be for example “Obtain Order”, “Deliver product” and “Receive Payment”. Each of these process assets is composed by the basic activities, inputs and outputs necessary within the “Selling” process. Afterwards to identify a specific process variant, the variant assets have to be selected among the candidate ones on the basis of a specific context. For example if we want to sell via electronic store an asset “Organize the web site” has to be selected. Process assets (variant or invariant) have to be then specialized on the basis of the context itself. The specialization aims to specify the behavior of each process asset using specializing actions. In particular an asset could be specialized adding IN/OUT artifacts to an activity, specializing an artifact, specializing an activity, adding an attribute to an artifact (size, compilation guideline, quality standards etc.), adding an attribute to an activity (required skills, tools, hardware or software resources etc.). For example if we want to sell via electronic store, the activity “Receive Payment” could be specialized in “Receive online Payment”. Finally the process assets selected and specialized are integrated into the required process variant.

2.1.1 BPL Logical Model

To choose the suitable process variant of a BPL to be applied in a specific context profile, we need a function associating to a specific context profile the process variant specific for the given context:

$$f: CP \rightarrow S \qquad (1)$$

where

- CP is the set of all the possible context profiles. An element $cp \in CP$ is represented as a vector of instantiated diversity factors DF_i $i=1, \dots, r$. Each DF_i is a factor characterizing a particular aspect of the environment and has a definition domain $[DF_i]=\{df_{i1}, df_{i2}, \dots, df_{iq}\}$ where each df_{ij} $j=1,...q$ is an instance of DF_i. So we can say that the set CP is: $CP= [DF_1] \times [DF_2] \times \dots\dots \times [DF_r]$.
- S is the set of all the possible process variants of the considered BPL.

f can be detailed in this way:

$$f\,(cp)= \Phi(\sigma(cp, K), \sigma(cp, \chi(cp))) \qquad (2)$$

If A is the set of all the process assets associated to the BPL, in (2):

- $K = \{ia_1,ia_2.....ia_n\} \subseteq A$ is the set of invariant assets realizing the commonality;
- χ: $CP \rightarrow CVA=A-K$ is the function referred to the activity 1.2 in Figure 3 associating to each fixed $cp \in CP$ the set of variant assets $\{va_1,va_2.....va_m\} \subseteq CVA$ realizing the process variability according to the fixed context cp. CVA=A-K is the set of the candidate variant assets associated to the BPL;
- σ is the function referred to the activity 1.3 in Figure 3 including the assets specialization rules on the basis of the context profile cp. It associates to a set of assets another set of assets specialized according to the fixed context profile. In particular $\sigma(cp, \{asset_1,asset_2.....asset_p\}) = \{\sigma_1(cp,asset_1), \sigma_2(cp,asset_2),\dots, \sigma_n(cp,asset_p)\}$, where $\sigma_1, \sigma_2,..., \sigma_p$ are transformations specializing respectively the assets $asset_1$, $asset_2$,..., $asset_p$ according to the context profile cp.
- Φ includes the integration rules useful to compose the commonality to the variability specialized according to the context profile cp.

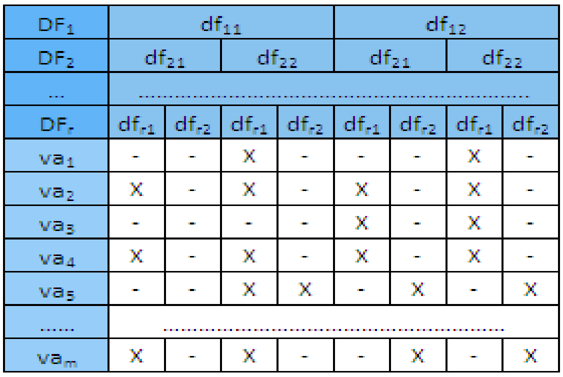
So, $f(cp) = \Phi(\sigma(cp, K), \sigma(cp, \chi(cp))) =$
$\Phi(\sigma(cp, \{ia_1,ia_2\dots,ia_n\}), \sigma(cp, \{va_1,va_2..,va_m\})) =$
$\Phi(\{\sigma_1(cp, ia_1), \sigma_2(cp, ia_2), ..., \sigma_n(cp,ia_n)\},$
$\{\sigma_{n+1}(cp, va_1), ..., \sigma_{n+m}(cp,va_m)\})$ identifies the process variant suitable for the context profile cp.

2.1.2 BPL Operative Model

The functions χ and σ defined in the logical model could be implemented through a decision tables system (more details in [9, 10, 11, 12]).

A *Variability Selection table* implements the function χ of the logical model. It allows to select the suitable variant assets composing the variability, characteristic of a specified context profile. So this kind of decision table is structured as follows (Figure4):

- the CONDITION quadrant contains the diversity factors DF_i i=1,...r driving the variant assets selection;
- the CONDITIONAL STATE quadrant contains the possible value of each diversity factor: $[DF_i]=\{df_{i1}, df_{i2}, \dots, df_{iq}\}$;
- the ACTION quadrant contains all the candidate variant assets ($\in$CVA) that can be selected to realize the process variability;
- the RULE quadrant identifies the relationships between each context profile and the variant assets to realize the corresponding process variability.

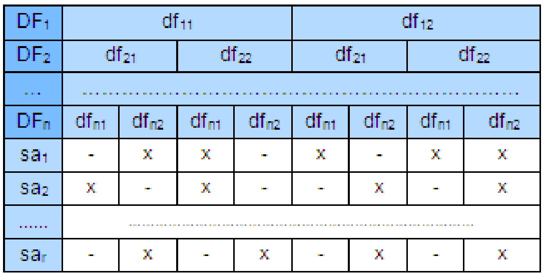


DF_1	df_{11}				df_{12}			
DF_2	df_{21}		df_{22}		df_{21}		df_{22}	
…	………………………………………………………………							
DF_r	df_{r1}	df_{r2}	df_{r1}	df_{r2}	df_{r1}	df_{r2}	df_{r1}	df_{r2}
va_1	-	-	X	-	-	-	X	-
va_2	X	-	X	-	X	-	X	-
va_3	-	-	-	-	X	-	X	-
va_4	X	-	X	-	X	-	X	-
va_5	-	-	X	X	-	X	-	X
……	…………………………………………………………							
va_m	X	-	X	-	-	X	-	X

Figure 4: Variability Selection table

An *Asset Specialization* table implements a function σ_i of the logical model. It allows to specialize a process asset (variant or invariant) on the basis of specified context profile, executing a set of specializing actions. So this kind of decision table is structured as follows (Figure5):

- the CONDITION quadrant contains the diversity factors DF_i i=1,...r driving the asset specialization;
- the CONDITIONAL STATE quadrant contains the possible values of each diversity factor: $[DF_i]=\{df_{i1}, df_{i2}, \ldots, df_{iq}\}$;
- the ACTION quadrant contains the actions to specialize the asset according to the specified context profile;
- the RULE quadrant identifies the relationships between each context profile and the specializing actions to be applied.



DF_1	df_{11}				df_{12}			
DF_2	df_{21}		df_{22}		df_{21}		df_{22}	
…	…………………………………………………							
DF_n	df_{n1}	df_{n2}	df_{n1}	df_{n2}	df_{n1}	df_{n2}	df_{n1}	df_{n2}
sa_1	-	x	x	-	x	-	x	x
sa_2	x	-	x	-	-	x	-	x
……	…………………………………………………							
sa_r	-	x	-	x	-	x	-	x

Figure 5: Asset Specialization table

2.2 Process Oriented Development (POD)

Starting from the process variant identified through a BPL, a SOA system can be implemented using the POD paradigm. In particular the process model is made understandable by an execution engine through successive transformations. In this way the POD allows to implement in a SOA system the changeable process requirements captured using a BPL, so a SOA systems line is actually realized. Referring to Figure 2, POD implements a SOA system starting from the process variant that is the output of the BPL.

Figure 6 illustrates in details the sequence of activities to realize this implementation. The abstract process model, formally described through a Process Modeling Language, is enriched with implementative details realizing a new intermediate model called Detailed Process Model (DPM). Finally a BPEL workflow is created as the last and most specific model.

Referring to Figure 6, in the *Specification* activity the process variant is translated into the corresponding DPM. As described above, the DPM will be realized enriching the process variant through implementative details making it understandable by an automatic translator. In particular a process model describes how activities are carried out exchanging artifacts. So in the DPM each artifact is mapped with a BPEL variable. Each variable has a specific type. This variable type should be traceable with the characteristic of the artifact. The information about an artifact in a process model are not enough to translate the artifact in a BPEL variable, that's why this association requires the pre-specification defined in the DPM. Moreover the DPM contains the mapping between the activities of the process model and the different services implementing them.

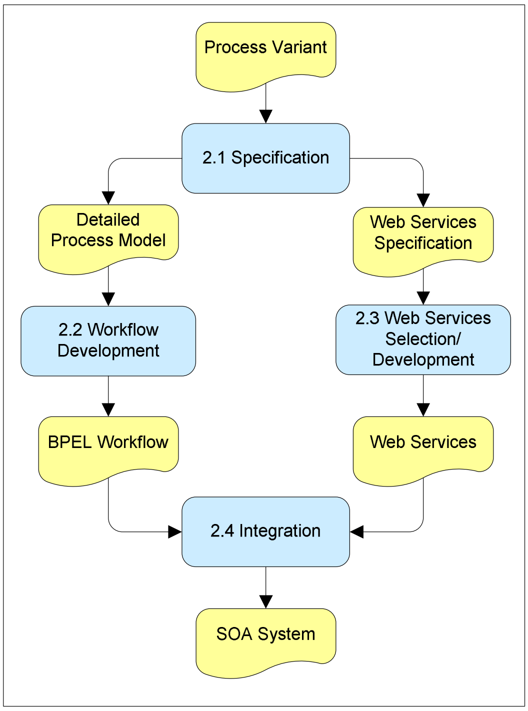


Figure 6: Process Oriented Development Approach

We want to highlight that real processes could provide the occurrence of human activities as well. So to implement the process activities executed by human agent, an extension of BPEL, BPEL4People, is used. In particular in the DPM the URL of the services carrying out the automatic activities and the e-mails of the agents carrying out the human activities should be specified. In this way the BPEL workflow will invoke the suitable software application to execute the automatic activities and will inform by e-mail the suitable agent to execute the human activities. The Web Services Specification produced by the *Specification* phase is useful to identify the services implementing the automatic activities (*Web Services Selection/Development*).

Starting from the information specified in the DPM the translator is able to generate an executable BPEL workflow (*Workflow Development*) implementing the underlying business process using the specified web services (*Integration*). For reasons of brevity we don't explain in details the translation algorithm but we can sketch the mapping between the elements of a business process and the corresponding BPEL objects. Table 1 shows the mapping between the process elements and the BPEL objects translating them. In the workflow BPEL the connectors linking all these elements are translated using the BPEL tag <link>.

Table 1: DPM-BPEL mapping

Process Element	**BPEL object**
Start Node	<Receive>
End Node	<Reply>
Activity	<Invoke>
Decision Node	It could be translated as a tag BPEL <if>, <while> or <repeat until> on the basis of the specific control flow

In particular an activity is translated using the BPEL tag <Invoke>. In a process model activities could be atomic or composite. The process model could be seen as a tree where atomic activities are the leaf-nodes. The translation algorithm starting from the leaf-nodes of the tree cover recursively the whole process model. Each atomic activity corresponds to a web service invocation. Each composite activity corresponds to the invocation of a BPEL workflow orchestrating the web services implementing its sub-activities that could be atomic or in their turn composite.

3. Case Study: a Selling SOA systems line

The proposed approach has been adopted in a research project. In Puglia some industrial and research organizations are working on it. They are collaborating to implement SOA systems so that business processes of different customers in the local agricultural and food market can be automated. Within this project there are different providers offering a number of software services that could be reused and adapted in each different business contexts according to the customer needs. That's why the project represents a field of interest for the proposed approach application.

The application of the proposed approach in the project has been realized through three main steps:

- BPL definition
- BPL application
- POD application

The BPL definition step consists in the extraction of a set of BPL aiming to model different kinds of business processes.

The BPL application consists to apply, starting from the defined BPL, the approach proposed in Figure 3.

Finally the POD application consists in the application of the approach proposed in Figure 6 to the process variant obtained in the BPL application step.

3.1 BPL Definition

Starting from the analysis of MIT library [13] we have achieved the information useful to obtain a BPL Library (we will deepen these outcomes in a future work). The MIT library represents a collection of more than 5000 business activities related in several business processes. These activities and processes have been used to obtain a set of BPL allowing to model several kinds of business processes.

In particular we describe in this session, the definition of a Selling BPL obtained using the MIT Selling processes.

Table 2: Invariant assets

ia_1	*Obtain order*
ia_2	*Deliver product or service*
ia_3	*Receive payment*

Starting from MIT Selling processes we have identified, through their common parts, the invariant assets realizing the Selling BPL commonality (Table 2). Each process asset is represented as one or more activities and their IN/OUT: for instance the asset ia_1="Obtain Order" is represented as in Figure 7.

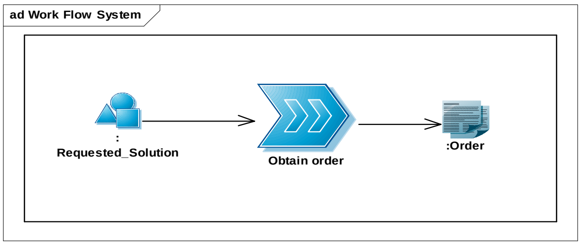


Figure 7: "Obtain Order" asset

Afterwards using the information about the different application contexts where the MIT processes are applied we have been able to define a number of diversity factors and their possible values (Table 3) to specify the conditions of the variability selection table.

Table 3: Diversity factors

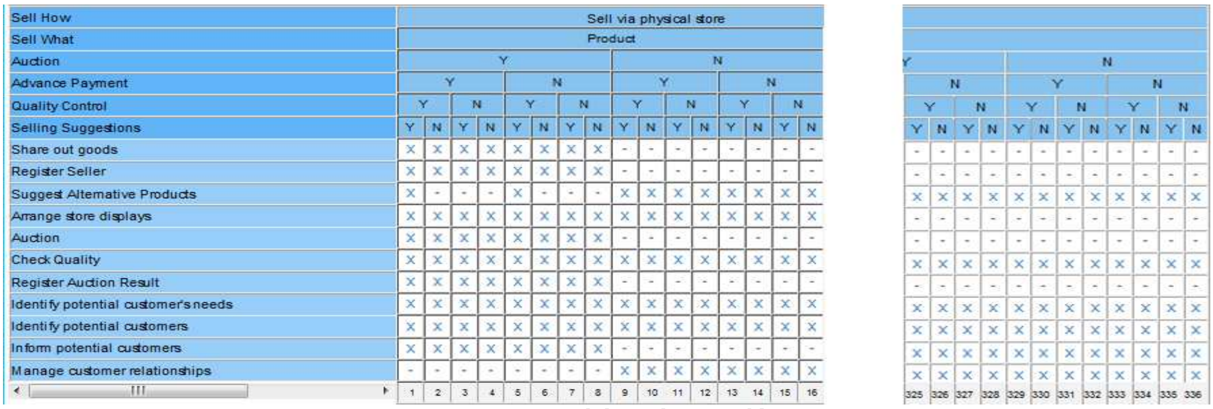
Diversity Factors	**Values**
Sell How	*Sell via physical store* *Sell via electronic store* *Sell via face-to-face* *Sell via direct mail* *Sell via email/fax* *Sell via television direct response marketing* *Sell via telemarketing*
Sell What	*Service, Product, Process*
Auction	*Y, N*
Advance Payment	*Y, N*
Quality control	*Y, N*
Selling Suggestions	*Y, N*

Finally using the variable parts of the MIT Selling processes, we have identified the candidate variant assets (Table 4) to specify the actions of the variability selection table.

Using this information, we have been able to realize the variability selection table (Figure 8). It encloses the rules to associate to each possible context profile the related variant assets. These have to be composed with the BPL invariant assets to obtain the process variant specific for the specified context profile.

Table 4: Candidate variant assets

va_1	*Share out goods*
va_2	*Register Seller s*
va_3	*Register Alternative Products*
va_4	*Arrange store displays*
va_5	*Auction*
va_6	*Check quality*
va_7	*Register Auction Result*
va_8	*Identify potential customers need*
va_9	*Identify potential customers*
va_{10}	*Inform potential customers*
va_{11}	*Manage customer relationships*

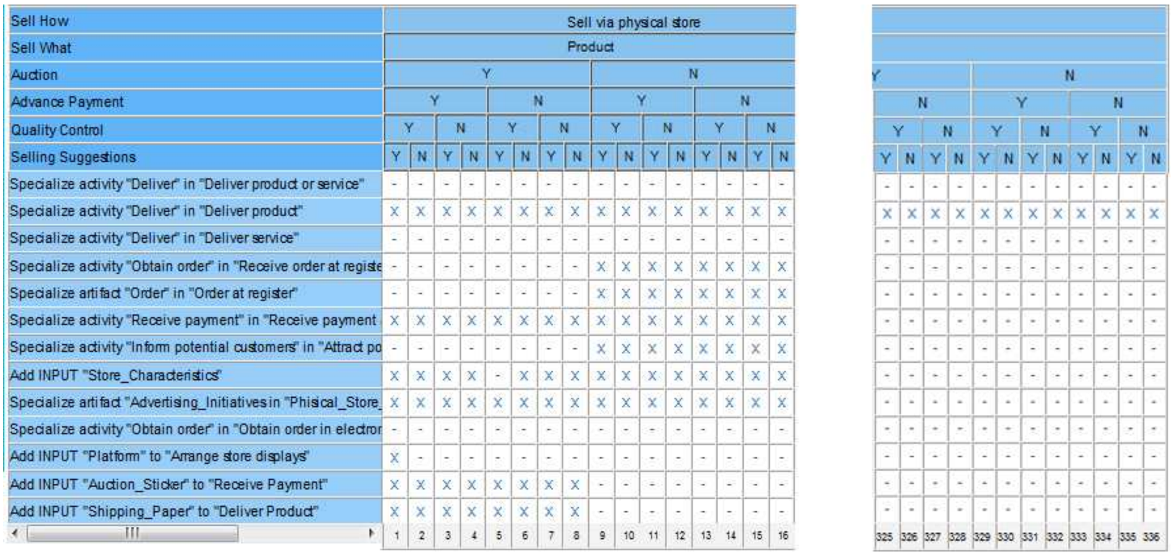


Sell How	Sell via physical store															
Sell What	Product															
Auction	Y								N							
Advance Payment	Y				N				Y				N			
Quality Control	Y		N		Y		N		Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Share out goods	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Register Seller	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Suggest Alternative Products	X	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X
Arrange store displays	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Auction	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Check Quality	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Register Auction Result	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Identify potential customer's needs	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Identify potential customers	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Inform potential customers	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Manage customer relationships	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Sell How												
Sell What												
Auction	Y				N							
Advance Payment	N				Y				N			
Quality Control	Y		N		Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Share out goods	-	-	-	-	-	-	-	-	-	-	-	-
Register Seller	-	-	-	-	-	-	-	-	-	-	-	-
Suggest Alternative Products	X	X	X	X	X	X	X	X	X	X	X	X
Arrange store displays	-	-	-	-	-	-	-	-	-	-	-	-
Auction	-	-	-	-	-	-	-	-	-	-	-	-
Check Quality	X	X	X	X	X	X	X	X	X	X	X	X
Register Auction Result	-	-	-	-	-	-	-	-	-	-	-	-
Identify potential customer's needs	X	X	X	X	X	X	X	X	X	X	X	X
Identify potential customers	X	X	X	X	X	X	X	X	X	X	X	X
Inform potential customers	X	X	X	X	X	X	X	X	X	X	X	X
Manage customer relationships	X	X	X	X	X	X	X	X	X	X	X	X
	325	326	327	328	329	330	331	332	333	334	335	336

Figure 8: Variability selection table

For instance, considering the column 5 of the table, we have that for the context profile cp*= (Sell via physical store, Product, Y, N, Y, Y) the variant assets to select are: va_1="Share out goods", va_2="Register Seller", va_3="Suggest Alternative Products", va_4="Arrange store displays", va_5="Auction", va_6="Check Quality", va_7="Register Auction Result", va_8="Identify potential customer needs", va_9="Identify potential customers", va_{10}="Inform potential customers". So to obtain the process variant specific for the given context profile we have to compose these assets with the invariant assets of the commonality.



Sell How	Sell via physical store															
Sell What	Product															
Auction	Y								N							
Advance Payment	Y				N				Y				N			
Quality Control	Y		N		Y		N		Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Specialize activity "Deliver" in "Deliver product or service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Deliver" in "Deliver product"	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Deliver" in "Deliver service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Receive order at registe	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X
Specialize artifact "Order" in "Order at register"	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X
Specialize activity "Receive payment" in "Receive payment	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Inform potential customers" in "Attract po	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X
Add INPUT "Store_Characteristics"	X	X	X	X	-	X	X	X	X	X	X	X	X	X	X	X
Specialize artifact "Advertising_Initiatives in "Phisical_Store_	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Obtain order" in "Obtain order in electror	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Platform" to "Arrange store displays"	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Auction_Sticker" to "Receive Payment"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Add INPUT "Shipping_Paper" to "Deliver Product"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Sell How												
Sell What												
Auction	Y				N							
Advance Payment	N				Y				N			
Quality Control	Y		N		Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Specialize activity "Deliver" in "Deliver product or service"	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Deliver" in "Deliver product"	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Deliver" in "Deliver service"	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Receive order at registe	-	-	-	-	-	-	-	-	-	-	-	-
Specialize artifact "Order" in "Order at register"	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Receive payment" in "Receive payment	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Inform potential customers" in "Attract po	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Store_Characteristics"	-	-	-	-	-	-	-	-	-	-	-	-
Specialize artifact "Advertising_Initiatives in "Phisical_Store_	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Obtain order in electror	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Platform" to "Arrange store displays"	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Auction_Sticker" to "Receive Payment"	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Shipping_Paper" to "Deliver Product"	-	-	-	-	-	-	-	-	-	-	-	-
	325	326	327	328	329	330	331	332	333	334	335	336

Figure 9: Assets Specialization table

Moreover to define the asset specialization tables for each process asset we have identified the specializing actions corresponding to each possible context profile. Since the conditions are the same for each process asset, we can incorporate all the asset specialization tables corresponding to the functions σ_i in only one table (corresponding to the function σ of the logical model). The resulting table (Figure 9) encloses the rules to associate to each context profile, the actions to be executed to specialize the behavior of all the assets related to the considered BPL.

For instance, according to the context profile cp* considered before (column 5), we need to specialize the activities: "Deliver" in "Deliver product", "Receive Payment" in "Receive Payment at register". Moreover we have to specialize the artifact "Advertising initiatives" in "Physical Store advertising". Finally we have to add the input "Auction Sticker" and "Shipping paper" respectively to "Receive Payment" and "Deliver Product activities".

3.2 BPL Application

The obtained BPL could be used to model different process variants of the Selling process.

During the research project this BPL has been used to model different process variants to fulfill different customers involved in the project. In particular, in this work we describe the application of the Selling BPL to automate the Selling process of a fish consortium in Puglia. The consortium was interested to sell via auction the fish caught by the ship-owner members boats. In particular the organization needed to model its Selling process and automate it in a SOA system integrating the services of different providers.

In this scenario the BPL has allowed to model the customer Selling process easily.

According to the first activity of Figure 3, for the customer was necessary to model a Selling process. In this case we have been able to use the BPL defined in the paragraph 3.1. For this BPL, the invariant assets are listed in table 1 and the candidate variant assets are listed in table 3.

Afterwards the variability selection table has been executed. Customer requirements have allowed to specify the context profile cp* corresponding to column 5 in Figure 6. In this way we have obtained the corresponding list of variant assets.

Finally the assets specialization activity has been performed using the Specialization Table shown in Figure 7. We have identified the list of specialization actions (see column 5 of the Specialization table). So each asset has been specialized through a set of specialization actions. Finally, all the specialized assets have been integrated considering their IN/OUT.

The obtained process model is shown in Figure 10.

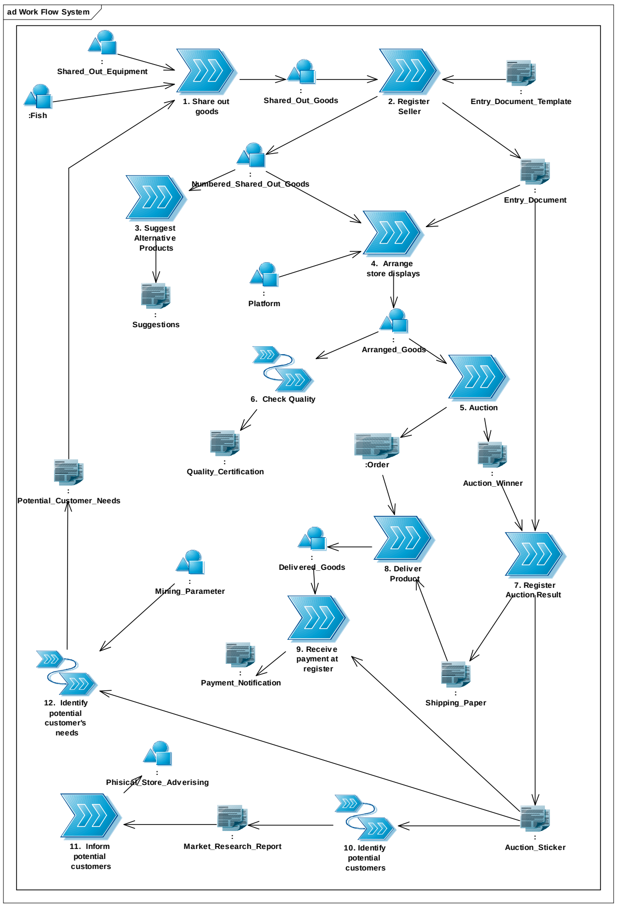


Figure10: Fish Consortium Selling process

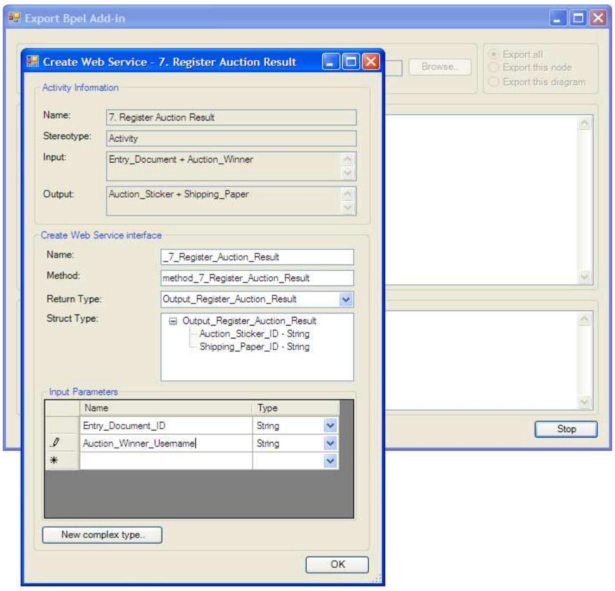
3.3 POD Application

According to the POD paradigm (Figure 6) we have realized the DPM starting from the process variant identified through the BPL. This new model has been obtained enriching the process model with implementative details. For this purpose we have implemented an application, ExportBPEL to support the DPM generation. This application has been realized as add-in of Enterprise Architect (EA), a graphical UML design and business analysis tool for modeling, documenting, building and maintaining object-oriented software systems.

In Figure 11 a screenshot of the application is shown. Here the implementation details related to the activity *7. Register_Auction_Result* are defined. In particular the developer has to specify the variable types implementing the input and output artifacts of the activity and the method implementing the activity itself. In this way all the specifications to realize the BPEL workflow are identified. In this case the activity must be implemented through the method *method_7_Register_Auction_Result* providing two input variables: *Entry_Document_ID* (String) and *Auction_Winner_Username* (String). The output variable must be a complex type composed by *Auction_Sticker_ID* (String) and *Shipping_Paper_ID* (String).

On the other hand Software Specification to realize the web services implementing the different activity have been realized. The different services provider have used these specifications to develop the suitable web services.

Starting from the DPM previously created ExportBPEL allows to realize automatically the Workflow BPEL implementing the underlying business process. Finally the BPEL workflow has been completed specifying the URL of the web services implemented by the different providers.



Name: 7. Register Auction Result
Stereotype: Activity
Input: Entry_Document + Auction_Winner
Output: Auction_Sticker + Shipping_Paper

Name: _7_Register_Auction_Result
Method: method_7_Register_Auction_Result
Return Type: Output_Register_Auction_Result
Struct Type: Output_Register_Auction_Result
- Auction_Sticker_ID - String
- Shipping_Paper_ID - String

Name	Type
Entry_Document_ID	String
Auction_Winner_Username	String

Figure11: Export BPEL application

4. Conclusions

This work represents a contribution to transfer the good practices of SPL (asset reuse and variation mechanisms) to SOA system development. The adoption of BPL and POD approaches permits to select the suitable process variant and to implement the appropriate SOA system for one or more customers.

In particular in this work we refer to the automation of the Selling process in a fish consortium. In this scenario we have evaluated some advantages deriving from the adoption of the proposed approach:

- it facilitates the selection of the suitable process model according to the customer requirements reducing its time and effort of 80%;
- it facilitates the implementation of the selected business process reducing its time and effort of 30%.

These values are qualitative evaluations on the basis of the feedbacks of the experts involved in the project. They compare time and effort for the modeling and implementation steps using our approach with the corresponding data obtained in similar cases performed with traditional approaches.

The described research project is not yet concluded. BPL and POD will apply in other business processes in the same research project to confirm and deepen these results.

References

[1] P. Clements and L. Northrop. “Software Product Lines: Practices and Patterns.” SEI Series in Software Engineering. Addison–Wesley, August 2001.

[2] T. Erl. “Service-Oriented Architecture: A Field Guide to Integrating Xml and Web Services”. Prentice Hall, 2004

[3] C. Wienands. “Studying the Common Problems with Service-Oriented Architecture and Software Product Lines” Service-Oriented Architecture (SOA) & Web Services Conference. Atlanta, GA, October 2006

[4] S.Cohen, R.Krut. “Proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines”, SPECIAL REPORT CMU/SEI-2008-SR-006, May 2008

[5] A. Schnieders and F. Puhlmann. “Variability mechanisms in e-business process families”. 9th International Conference on Business Information Systems, Klagenfurt (Austria), June 2006.

[6] J. Bayer, M. Kose, A. Ocampo. “Improving the Development of e-Business Systems by Introducing Process-Based Software Product Lines”. 7th International Conference on Product Focused Software Process Improvement (PROFES), Amsterdam (Holland), June 2006.

[7] A. Schnieders. “Variability Mechanism Centric Process Family Architectures”. 13th Annual IEEE International Conference on the Engineering of Computer Based Systems ECBS 2006, pp. 289-298, IEEE Computer Society Press, 2006.

[8] N. Boffoli, M. Cimitile, F.M. Maggi. “Managing Business Process Flexibility and Reuse through Business Process Lines”, 4th Conference on Software and Data Technologies (ICSOFT), July 2009.

[9] J. Vanthienen, C. Mues, G. Wets, K. Delaere. “A tool-supported approach to inter-tabular verification, Expert Systems with Applications”, 15, pp. 277-285, 1998

[10] R. Maes, J.E.M. Van Dijk. “On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems”, The Computer Journal, 31(6), 1988

[11] T.B. Ho, D. Cheung, and H. Liu. “Advances in Knowledge Discovery and Data Mining”, 9th Pacific-Asia Conference, Vietnam, 2005.

[12] A. Bar-Or, D. Keren, A. Schuster, R. Wolff. “Hierarchical Decision Tree Induction in Distributed Genomic Databases”, IEEE Transactions on Knowledge and Data Engineering, Vol.17, 2005.

[13] T.W. Malone, K. Crowston, G.A. Herman, Organizing Business Knowledge-The MIT Process Handbook, MIT Press Cambridge, 2003.

Invited Speaker

Dr. Nicola Boffoli

Software Engineering LABoratory (SERLAB)

University of Bari, Italy

- Research Interests:
  - *software processes, business processes, software product lines, service oriented architectures*
- Presentation Title
  - *Managing SOA System Variation through Business Process Lines and Process Oriented Development*

SOAPL 2009
S.Francisco, 25th August

Managing SOA System Variation through Business Process Lines and Process Oriented Development

Nicola Boffoli, Marta Cimitile,
Fabrizio Maria Maggi, Giuseppe Visaggio
SERLAB - Department of Informatics
University of Bari - Italy
{boffoli, cimitile, maggi, visaggio}@di.uniba.it

SERLAB
SOFTWARE ENGINEERING RESEARCH LABORATORY
c/o DIPARTIMENTO DI INFORMATICA
VIA ORABONA, 4 - 70126 - BARI
TEL: + 39. 080.5443279 FAX: + 39.080.5442536

Outline

SPL & SOA

- Peculiarities and Comparison
- Research Question

Proposal

- Business Process Lines
- Process Oriented Development
- Case Study: a Selling SOA-System Line

SPL & SOA

⇨ Two common perspectives

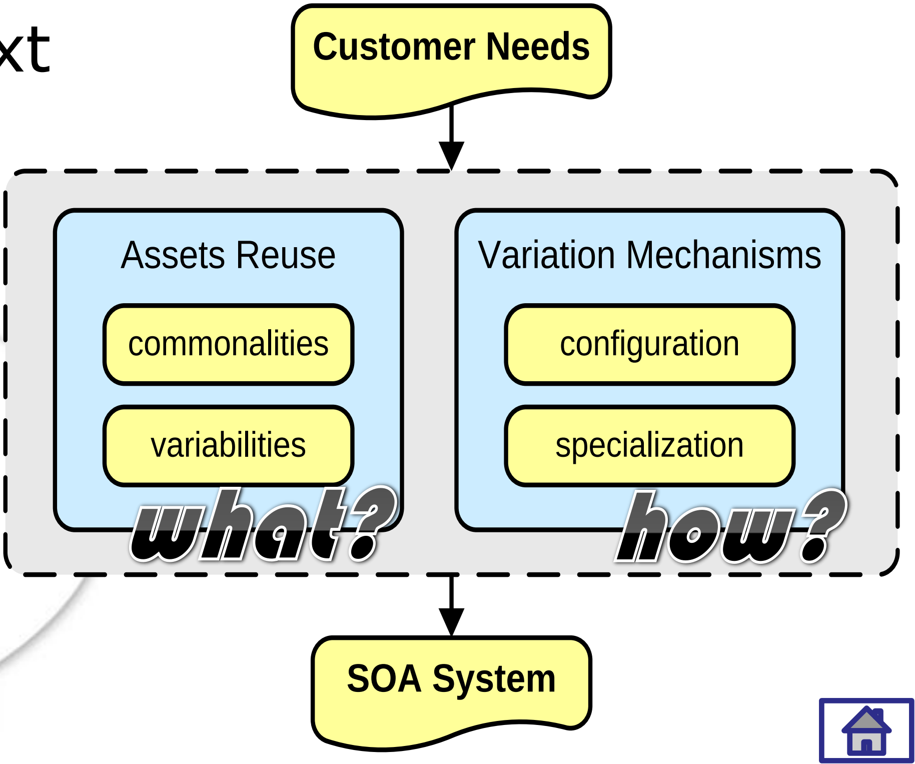
- ❑ Software reuse and flexibility
  - implementing new software systems reusing existing software resources
  - allowing to adapt the systems to the different customers of a whole market segment

⇨ *However…*

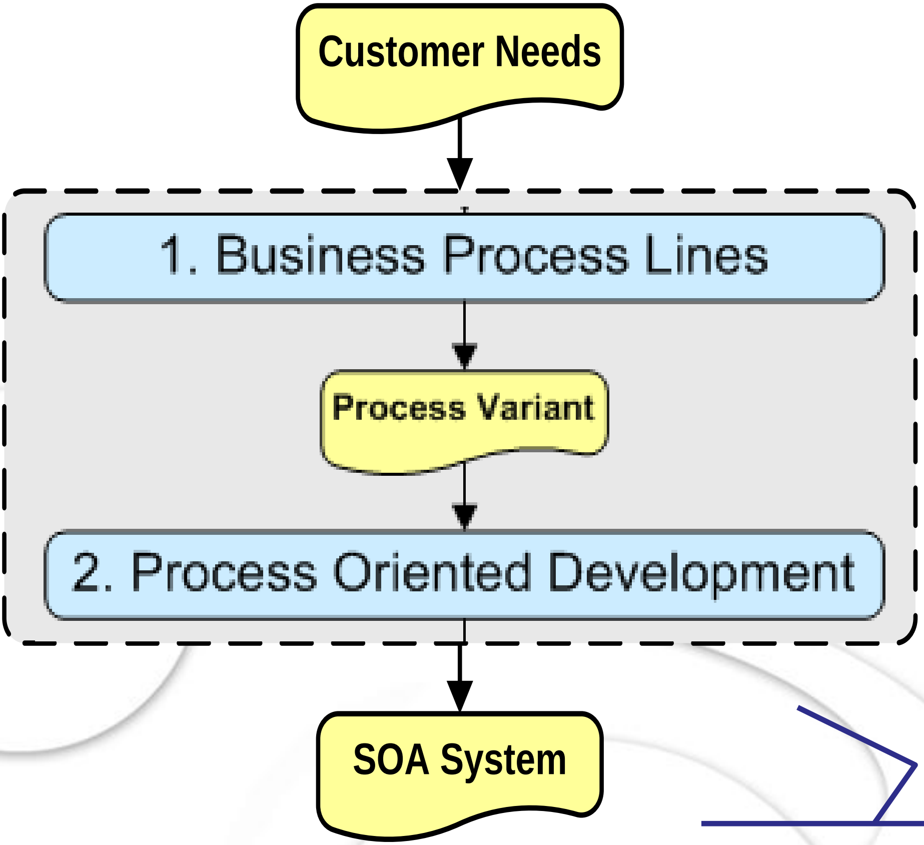
- ❑ SPL focuses on the commonality and variability to build a set of software products
- ❑ SOA allows to compose, orchestrate and maintain solutions based on services, implementing business processes

Research Question

- How can SOA systems benefit from SPL good practices?
  - reuse and variation management approaches
- In the SOA system context
  - *what* is the core asset?
  - *how* can we implement the variation mechanisms?



Our Proposal



according to the SPL practices is able to model the ***process variant****, suitable for specific customer needs*

is able to transform a process variant into an executable ***SOA system***

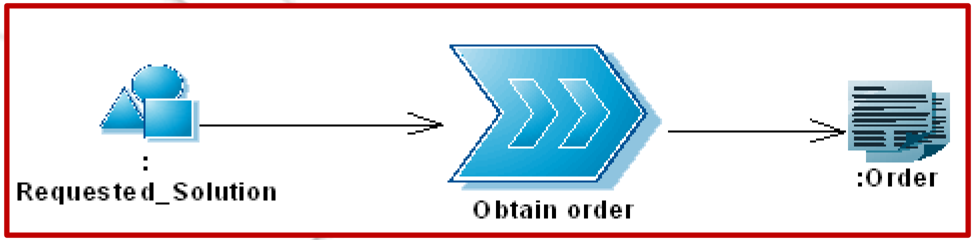
Case Study*: a selling SOA system line*

Business Process Lines (1/3)

- A BPL is a set of similar business processes
  - sharing a common part (commonality)
  - characterized by a variant part (variability) depending on the specific context where the process will be applied

- A BPL works integrating a set of process assets
  - i.e. atomic reusable parts of a business process (one or more activities with their IN/OUT)

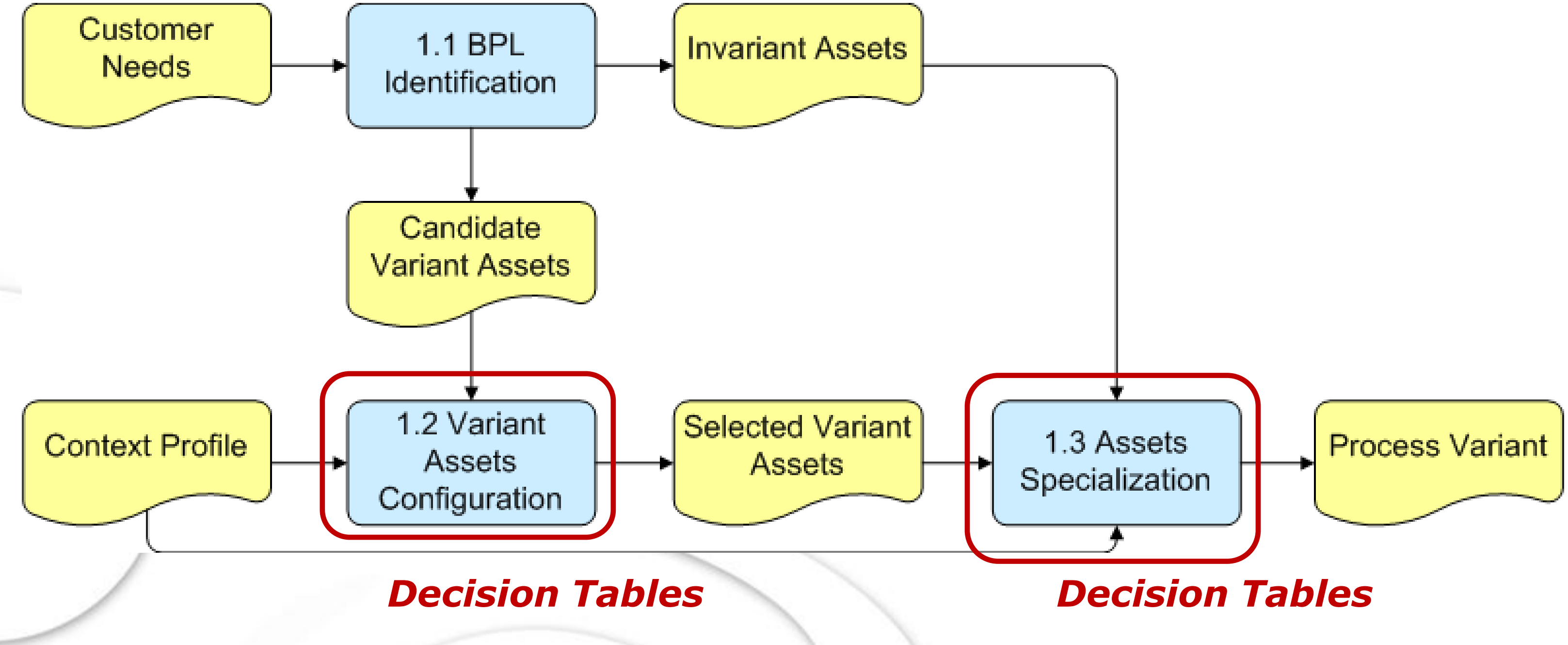
PROCESS ASSET



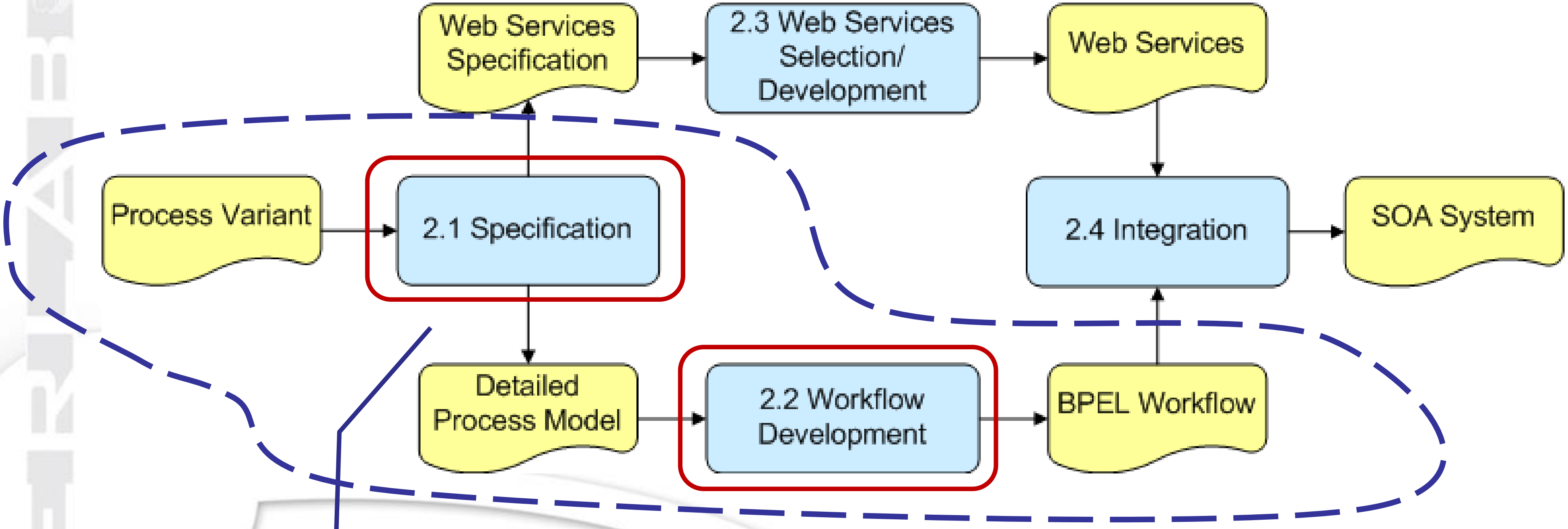
Business Process Lines (2/3)

- A BPL consists in:
  - a set of invariant assets (commonality)
  - a set of variant assets (variability)
    - each process of the BPL requires a different subset of the variant assets chosen according to the specific context
  - a set of rules: to build automatically the appropriate process model (the “process variant”)
    - Variability Selection
      - *Variant assets are selected among the candidate ones on the basis of the specific context*
    - Assets Specialization
      - *Each asset (variant or invariant) is specialized modifying its characteristics according to the specified context*

Business Process Lines (3/3)



Process Oriented Development



Implementative details,
mapping between:

- *process artifacts – BPEL variables*
- *process activities – serivices or BPEL WkF*

Process Element	**BPEL object**
Start Node	<Receive>
End Node	<Reply>
Activity	<Invoke>
Decision Node	It could be translated as a tag BPEL <if>, <while> or <repeat until> on the basis of the specific control flow

- Case Study -
a Selling SOA System Line

Case Study: overview

- Our proposal has been adopted in a research project
  - collaboration with industrial organizations from *Puglia*
  - SOA systems implementation to automate business processes in the field of *local agricultural and food*

Case Study: overview

⇨ 3 main steps

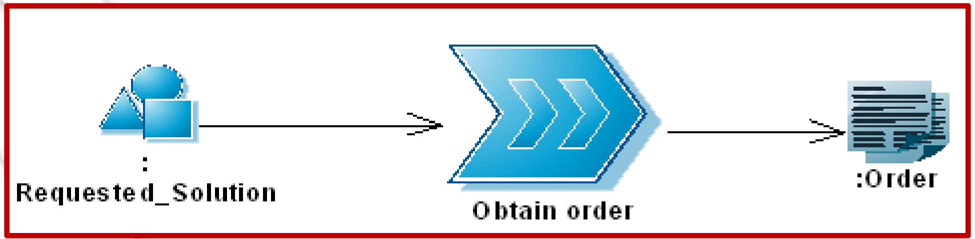
1. *BPL definition*
2. *BPL application*
3. *POD application*

Step1: BPL Definition (1/4)

- Analysis of MIT library (2003, Malone et Al.)
  - information useful to obtain *a set of BPL*
    (allowing to model several kinds of business processes)
  - in this session the definition of a *Selling BPL*
    (obtained using the MIT Selling processes)

- Results

Invariant Assets	
ia$_1$	*Obtain order*
ia$_2$	*Deliver product or service*
ia$_3$	*Receive payment*

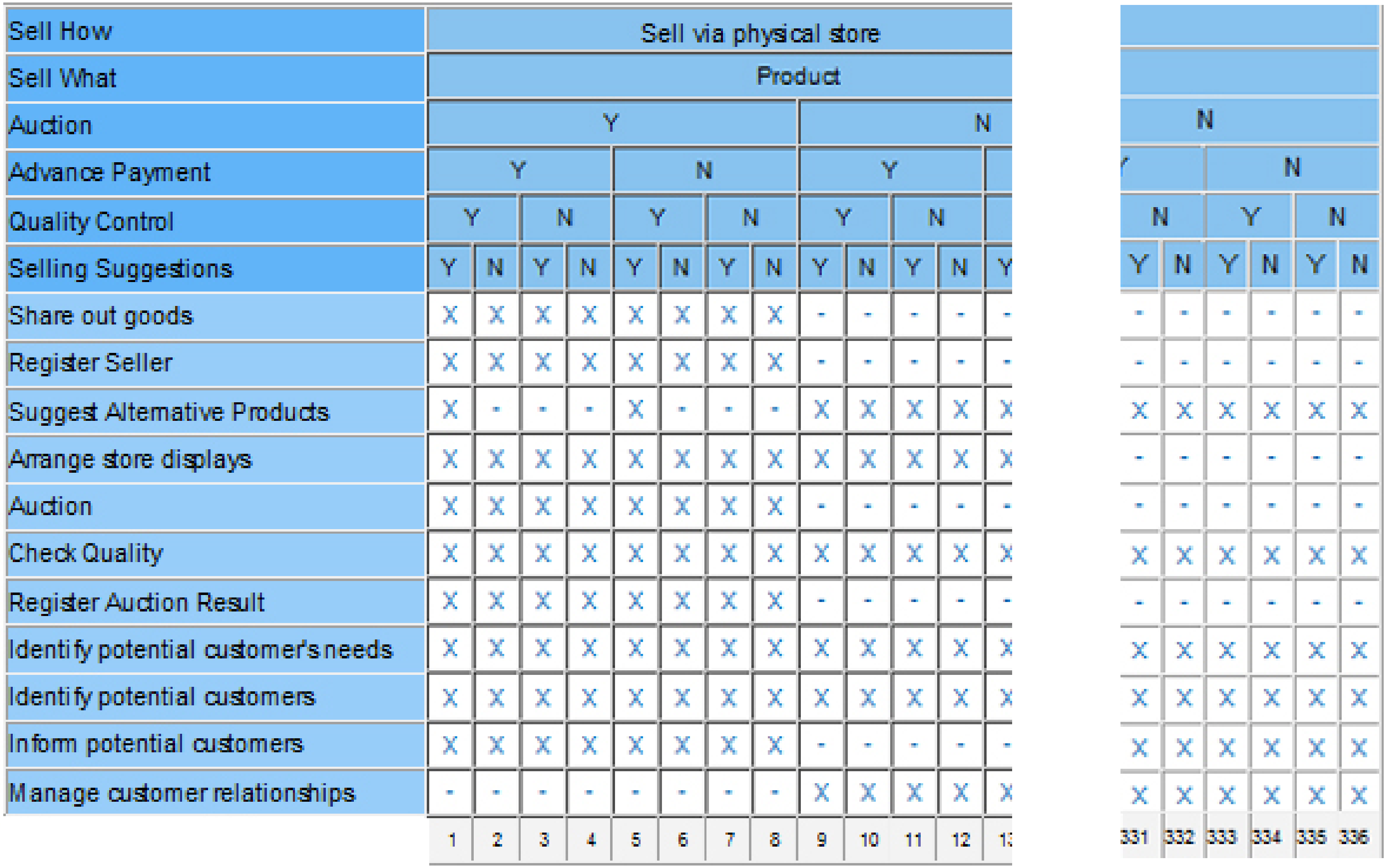


Step1: BPL Definition (2/4)

Diversity Factors	*Values*
Sell How	*Sell via physical store* *Sell via electronic store* *Sell via face-to-face* *Sell via direct mail* *Sell via email/fax* *Sell via television direct response marketing* *Sell via telemarketing*
Sell What	*Service, Product, Process*
Auction	*Y, N*
Advance Payment	*Y, N*
Quality control	*Y, N*
Selling Suggestions	*Y, N*

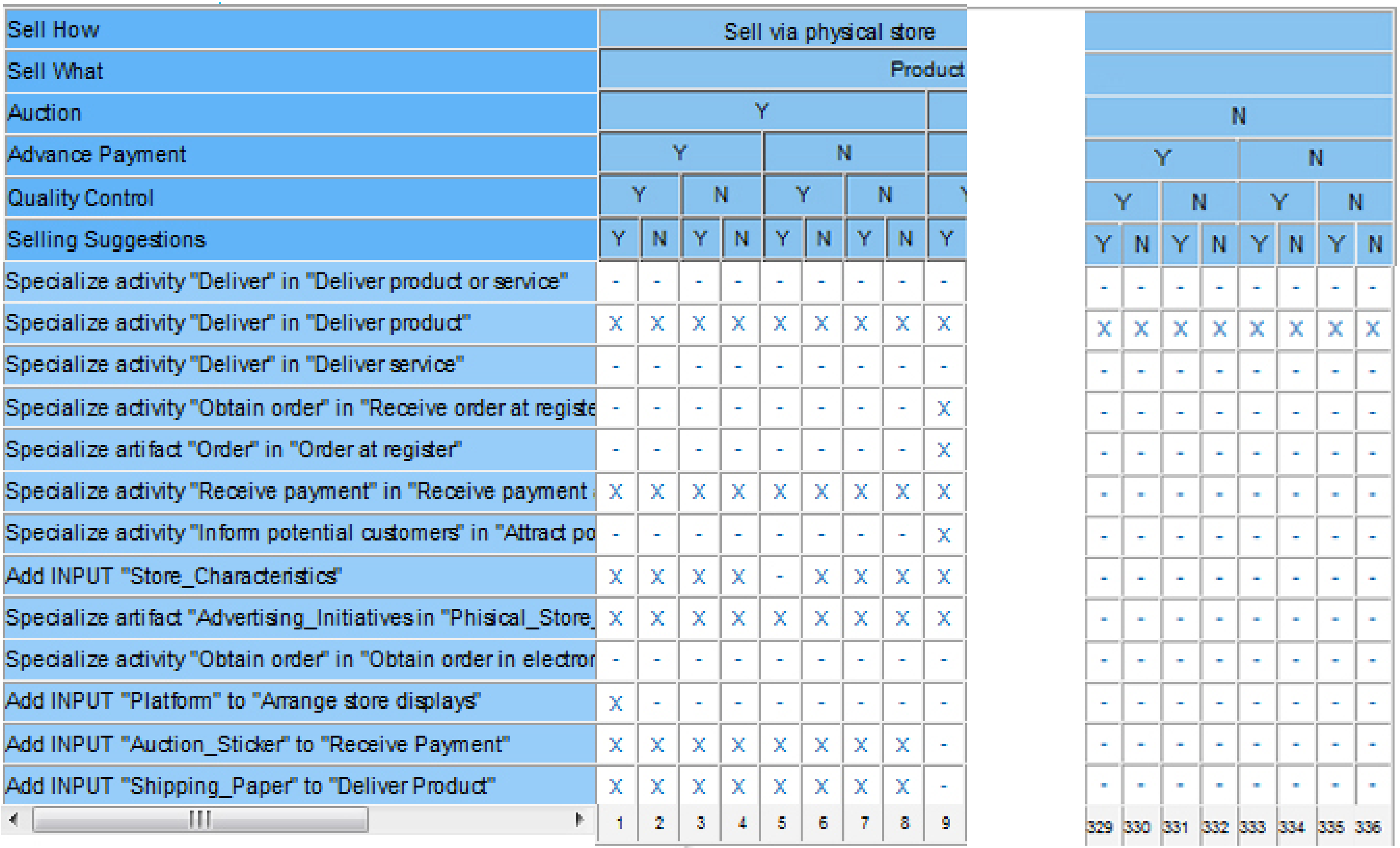
Candidate Variant Asset	
va$_1$	*Share out goods*
va$_2$	*Register Sellers*
va$_3$	*Register Alternative Products*
va$_4$	*Arrange store displays*
va$_5$	*Auction*
va$_6$	*Check quality*
va$_7$	*Register Auction Result*
va$_8$	*Identify potential customers need*
va$_9$	*Identify potential customers*
va$_{10}$	*Inform potential customers*
va$_{11}$	*Manage customer relationships*

Step1: BPL Definition (3/4)



Sell How	Sell via physical store																		
Sell What	Product																		
Auction	Y								N					N					
Advance Payment	Y				N				Y					Y		N			
Quality Control	Y		N		Y		N		Y		N			N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	Y	N	Y	N	Y	N
Share out goods	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Register Seller	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Suggest Alternative Products	X	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	X	X	X
Arrange store displays	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-
Auction	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Check Quality	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Register Auction Result	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Identify potential customer's needs	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Identify potential customers	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Inform potential customers	X	X	X	X	X	X	X	X	-	-	-	-	-	X	X	X	X	X	X
Manage customer relationships	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	12	13	331	332	333	334	335	336

Step1: BPL Definition (4/4)



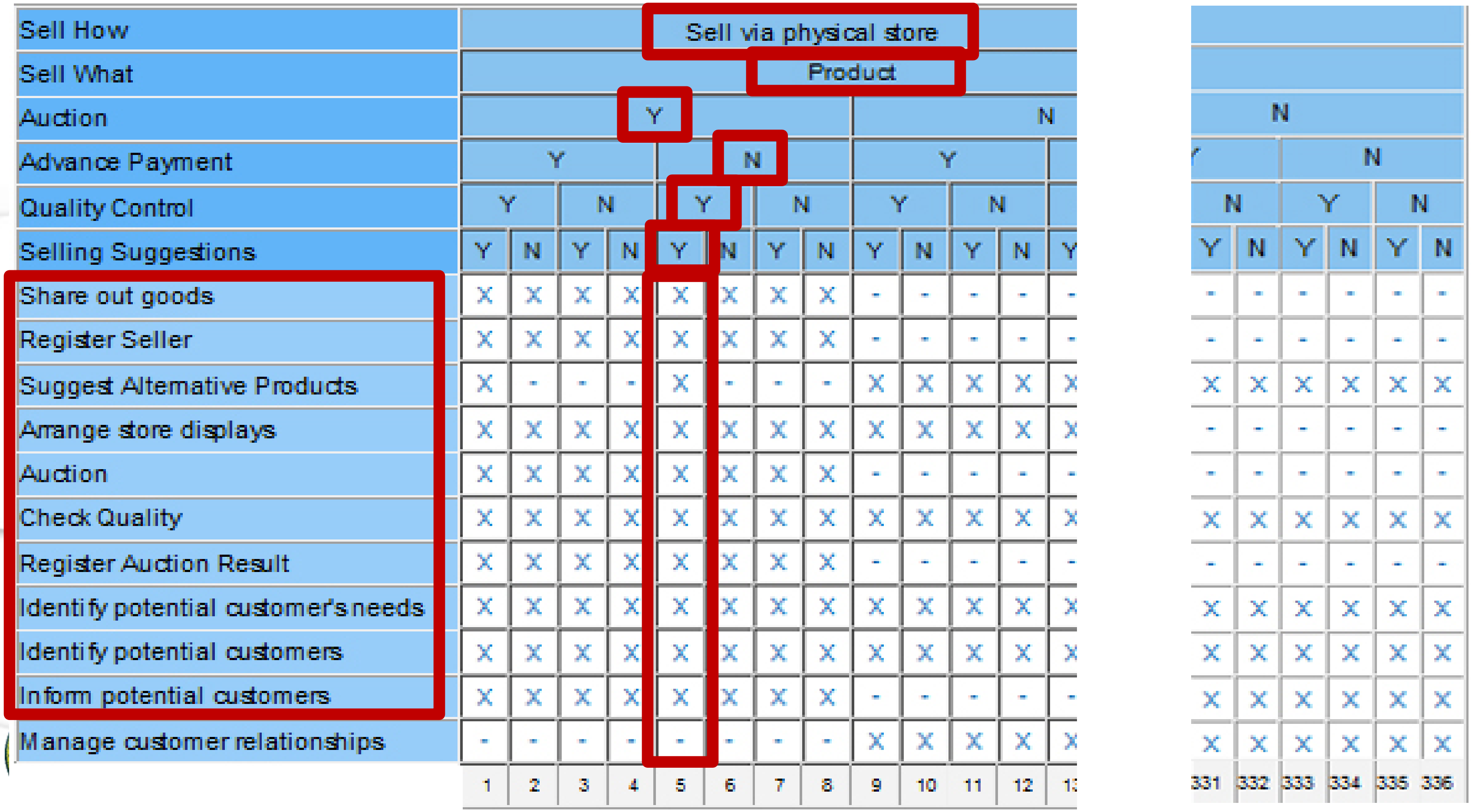
Sell How	Sell via physical store																
Sell What	Product																
Auction	Y									N							
Advance Payment	Y				N					Y				N			
Quality Control	Y		N		Y		N		Y	Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	Y	N	Y	N	Y	N	Y	N
Specialize activity "Deliver" in "Deliver product or service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Deliver" in "Deliver product"	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Deliver" in "Deliver service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Receive order at registe	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Specialize artifact "Order" in "Order at register"	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Specialize activity "Receive payment" in "Receive payment	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Specialize activity "Inform potential customers" in "Attract po	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Add INPUT "Store_Characteristics"	X	X	X	X	-	X	X	X	X	-	-	-	-	-	-	-	-
Specialize artifact "Advertising_Initiatives in "Phisical_Store_	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Obtain order in electror	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Platform" to "Arrange store displays"	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Auction_Sticker" to "Receive Payment"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-
Add INPUT "Shipping_Paper" to "Deliver Product"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-
	1	2	3	4	5	6	7	8	9	329	330	331	332	333	334	335	336

Step2: BPL Application (1/3)

- The obtained BPL could be used to model different process variants of the Selling process
- In this work
  - we describe the application of the Selling BPL to automate the selling process of *a fish consortium* in Manfredonia (Puglia)

Step2: BPL Application (2/3)

⇨ Context Profile =("Sell via physical store", "Product", "Y", "N", "Y", "Y")

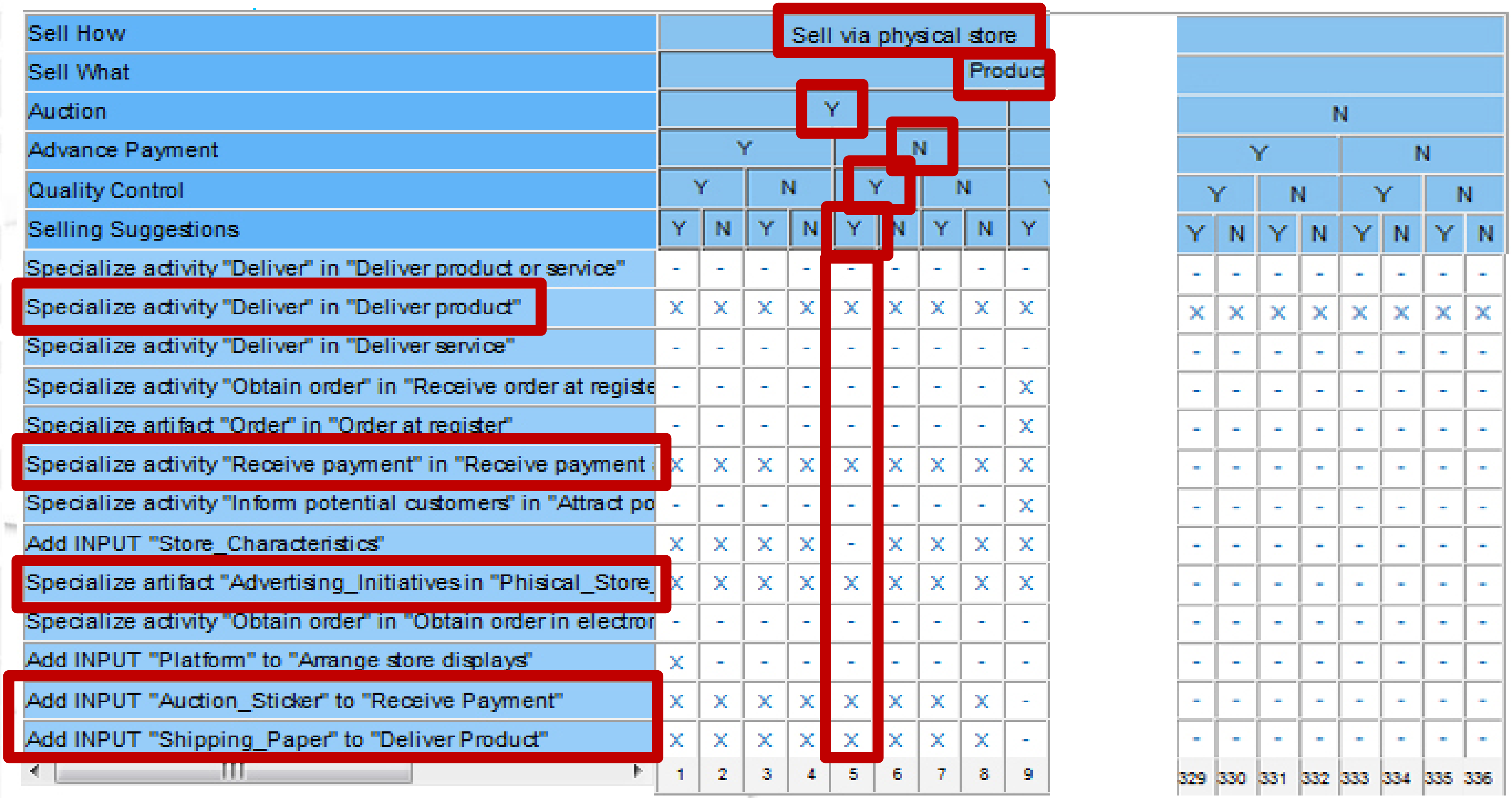


Sell How	Sell via physical store																		
Sell What	Product																		
Auction	Y								N					N					
Advance Payment	Y				N				Y					Y		N			
Quality Control	Y		N		Y		N		Y		N			N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	Y	N	Y	N	Y	N
Share out goods	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Register Seller	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Suggest Alternative Products	X	-	-	-	X	-	-	-	X	X	X	X	X	X	X	X	X	X	X
Arrange store displays	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-
Auction	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Check Quality	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Register Auction Result	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
Identify potential customer's needs	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Identify potential customers	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Inform potential customers	X	X	X	X	X	X	X	X	-	-	-	-	-	X	X	X	X	X	X
Manage customer relationships	-	-	-	-	-	-	-	-	X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	12	13	331	332	333	334	335	336

Step2: BPL Application (3/3)

Fish Consortium Selling Process

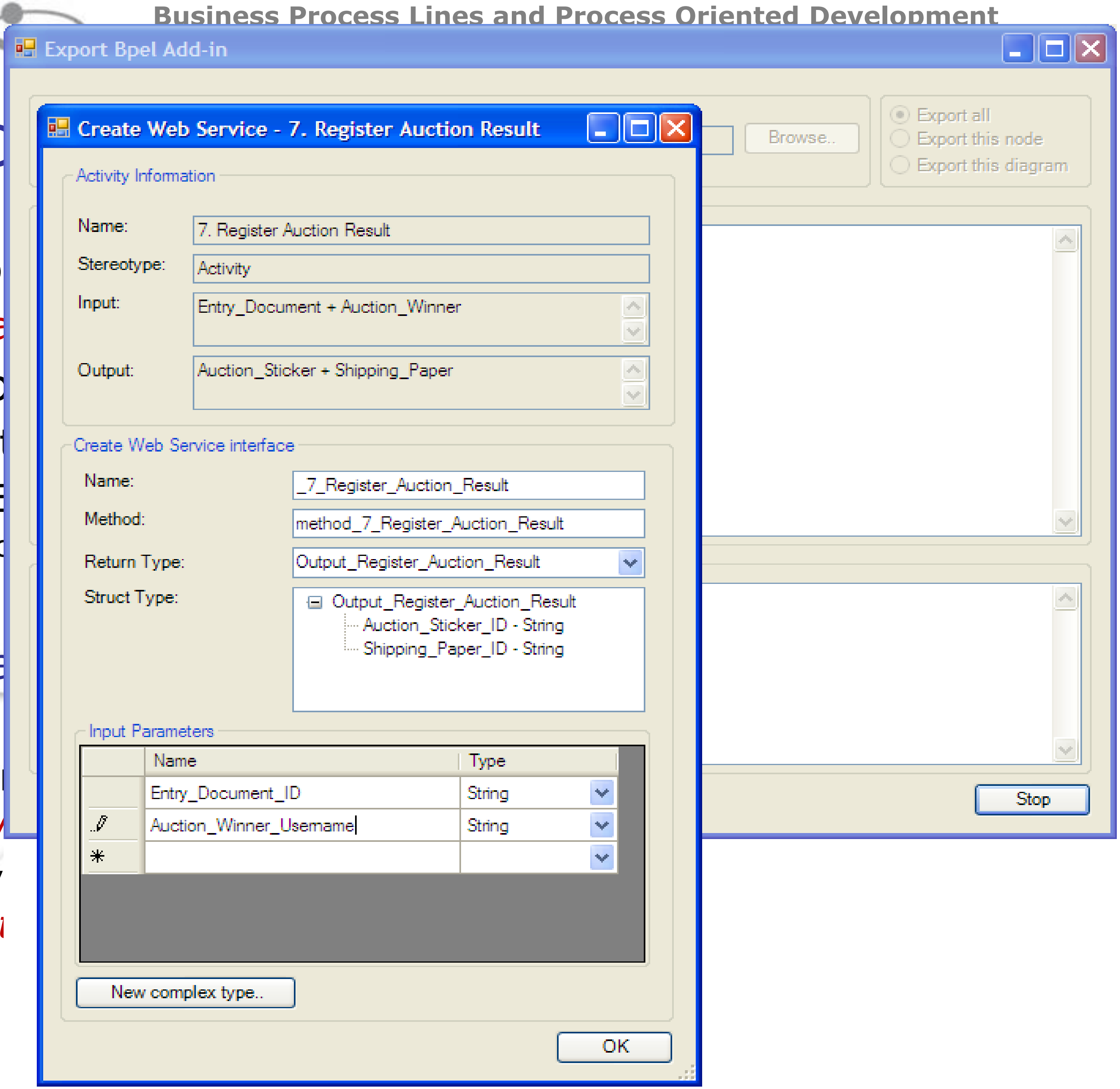
⇨ Context Profile =("Sell via physical store", "Product", "Y", "N", "Y", "Y")



Sell How	Sell via physical store																
Sell What	Produc																
Auction	Y									N							
Advance Payment	Y				N					Y				N			
Quality Control	Y		N		Y		N		Y	Y		N		Y		N	
Selling Suggestions	Y	N	Y	N	Y	N	Y	N	Y	Y	N	Y	N	Y	N	Y	N
Specialize activity "Deliver" in "Deliver product or service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Deliver" in "Deliver product"	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Specialize activity "Deliver" in "Deliver service"	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Receive order at registe	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Specialize artifact "Order" in "Order at register"	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Specialize activity "Receive payment" in "Receive payment	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Specialize activity "Inform potential customers" in "Attract po	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
Add INPUT "Store_Characteristics"	X	X	X	X	-	X	X	X	X	-	-	-	-	-	-	-	-
Specialize artifact "Advertising_Initiatives in "Phisical_Store_	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
Specialize activity "Obtain order" in "Obtain order in electror	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Platform" to "Arrange store displays"	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Add INPUT "Auction_Sticker" to "Receive Payment"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-
Add INPUT "Shipping_Paper" to "Deliver Product"	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-	-
	1	2	3	4	5	6	7	8	9	329	330	331	332	333	334	335	336

Step3: POD

- ⇨ According to
  - ❑ Process Va
- ⇨ For this purp
  - ❑ an applicat
  - ❑ add-in of E
    - a graphic
      software
- ⇨ Activity7: *Re*
  - ❑ 1 method:
  - ❑ 2 input var
    Auction_W
  - ❑ 1 output v
    Auction_S



Export Bpel Add-in

Browse..

● Export all
○ Export this node
○ Export this diagram

Stop

Create Web Service - 7. Register Auction Result

Activity Information

Name: 7. Register Auction Result

Stereotype: Activity

Input: Entry_Document + Auction_Winner

Output: Auction_Sticker + Shipping_Paper

Create Web Service interface

Name: _7_Register_Auction_Result

Method: method_7_Register_Auction_Result

Return Type: Output_Register_Auction_Result

Struct Type:
⊟ Output_Register_Auction_Result
  Auction_Sticker_ID - String
  Shipping_Paper_ID - String

Input Parameters

	Name	Type
	Entry_Document_ID	String
✎	Auction_Winner_Username	String
✱		

New complex type..

OK

DIB

Conclusions and Future Works (1/2)

- ⇨ This work proposes to apply the good practices of SPL to SOA, the authors introduce
  - ❑ the BPL permits to select the suitable process variant
  - ❑ the POD permits to implement the appropriate SOA

- ⇨ Case study: selling process in a fish consortium
  - ❑ - 80% in selection of the suitable process model according to the customer requirements
  - ❑ - 30% in implementation of the selected business
  - ❑ These values are qualitative evaluations on the basis of the feedbacks of the experts involved in the project

Conclusions and Future Works (2/2)

⇨ Variation Mechanisms

- *business processes modeling*: the BPL support the most of typical variation of SOA system
- *services development*: the variations are addressable to the traditional SPL approaches

⇨ Future Works

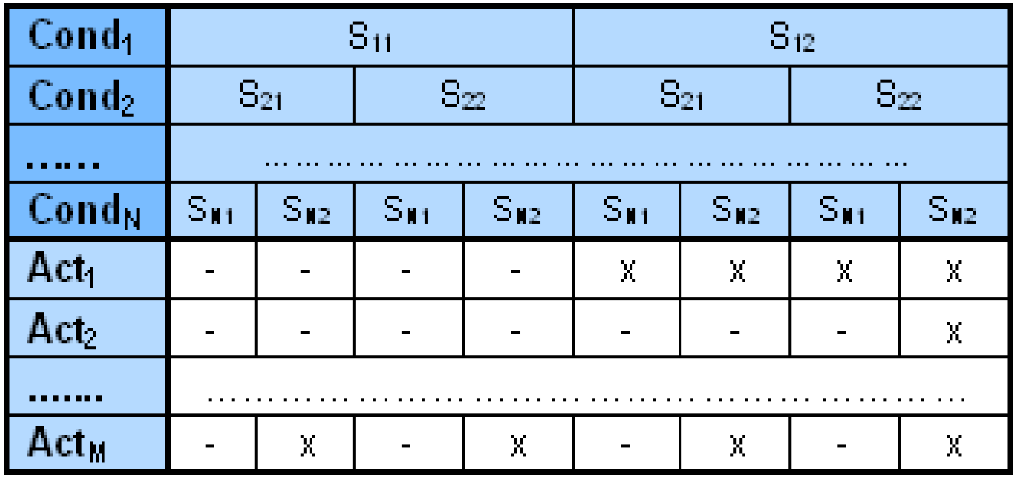
- BPL perspective
  - tailoring/adopting specific SPL techniques: features model, aspects, …
- SPL perspective
  - SOA system is a software product ➔ SPL generating SOA product too

Thank You!

Questions?

Decision Table Formalism

- ⇨ A decision table (DT) is divided in four quadrants: conditions (Cond), conditional states (S), actions (Act) and rules (x)
- ⇨ The table is defined so that each combination of conditions and conditional states corresponds to a set of actions to carry out

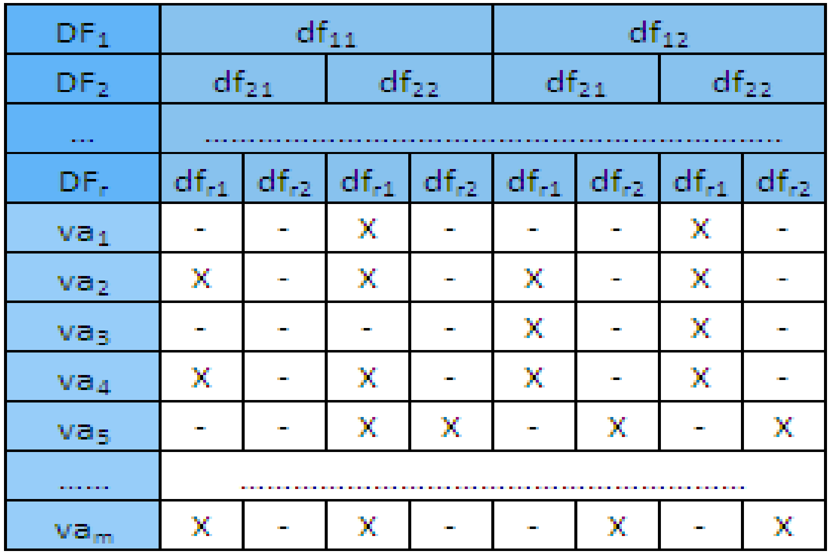


$Cond_1$	S_{11}				S_{12}			
$Cond_2$	S_{21}		S_{22}		S_{21}		S_{22}	
……	… …							
$Cond_N$	S_{N1}	S_{N2}	S_{N1}	S_{N2}	S_{N1}	S_{N2}	S_{N1}	S_{N2}
Act_1	-	-	-	-	x	x	x	x
Act_2	-	-	-	-	-	-	-	x
…….	…………………………………………………………………………							
Act_M	-	x	-	x	-	x	-	x

- Compact overview
- Modular knowledge organization
- Evaluation of consistency, completeness and redundancy

Variability Selection DT

- For each BPL a Variability Selection DT is built to select the variant assets characteristic of a specific context among the candidate variant assets
  - the CONDITION quadrant contains the diversity factors DF_i i=1,...r driving the variant assets selection
  - the CONDITIONAL STATE quadrant contains the possible values of each factor: $[DF_i]=\{df_{i1}, df_{i2}, \ldots, df_{iq}\}$
  - the ACTION quadrant contains all the candidate variant assets that can be selected to realize the process variability
  - the RULE quadrant identifies the relationships between each context profile and the variant assets



DF_1	df_{11}				df_{12}			
DF_2	df_{21}		df_{22}		df_{21}		df_{22}	
...	....							
DF_r	df_{r1}	df_{r2}	df_{r1}	df_{r2}	df_{r1}	df_{r2}	df_{r1}	df_{r2}
va_1	-	-	X	-	-	-	X	-
va_2	X	-	X	-	X	-	X	-
va_3	-	-	-	-	X	-	X	-
va_4	X	-	X	-	X	-	X	-
va_5	-	-	X	X	-	X	-	X
......	....							
va_m	X	-	X	-	-	X	-	X

Asset Specialization DT

- For each asset, variant or invariant, an Asset Specialization DT is built as follows
  - the CONDITION quadrant contains the diversity factors DF_i $i=1,\dots r$ driving the asset specialization
  - the CONDITIONAL STATE quadrant contains the possible values of each diversity factor: $[DF_i]=\{df_{i1}, df_{i2}, \dots, df_{iq}\}$
  - the ACTION quadrant contains the specializing actions to characterize the asset according to the specified context profile
  - the RULE quadrant identifies the relationships between each context profile and the specializing actions to be applied

DF_1	df_{11}				df_{12}			
DF_2	df_{21}		df_{22}		df_{21}		df_{22}	
...	....							
DF_n	df_{n1}	df_{n2}	df_{n1}	df_{n2}	df_{n1}	df_{n2}	df_{n1}	df_{n2}
sa_1	-	x	x	-	x	-	x	x
sa_2	x	-	x	-	-	x	-	x
......	....							
sa_r	-	x	-	x	-	x	-	x

Towards an Approach for Service-Oriented Product Line Architectures

Flávio Mota Medeiros[1,2] Eduardo Santana de Almeida[2,3]
Silvio Romero de Lemos Meira[1,2,3]
Federal University of Pernambuco (UFPE)[1]
Reuse in Software Engineering (RiSE)[2]
Recife Center for Advanced Studies and Systems (C.E.S.A.R.)[3]
{fmm2,srlm}@cin.ufpe.br esa@rise.com.br

Abstract

Service-Oriented Architecture (SOA) has appeared as an emergent approach for developing distributed applications as a set of self-contained and business-aligned services. SOA aids solving integration and interoperability problems and provides a better Information Technology (IT) and business alignment, giving more flexibility for the enterprises. However, SOA does not provide support for high customization and systematic planned reuse to develop applications that fit customer individual needs. In this paper, we propose an approach in which SOA applications are developed as Software Product Lines (SPLs). Thus, the term Service-Oriented Product Line is used for service-oriented applications that share common parts and vary in a regular and identifiable manner. In this context, high customization and systematic planned reuse are achieved through managed variability and the use of a two life-cycle model as in SPL engineering: core assets and product development. We conclude the paper with an initial case study in the conference management domain explaining the steps of our approach.

1. Introduction

In software development, there is an essential need to reduce costs, effort, and time to market of software products [1]. It is crucial to develop flexible systems able to adapt to market changes quickly [2]. In addition, there are lots of different technologies appearing, and enterprises need to integrate their software investments (legacy systems) with these new technologies [3]. However, the complexity and size of systems are increasing, and products must fit customer or market segment needs [4].

In this context, SOA is an emergent approach to solve integration and interoperability problems [5, 6], align IT and business goals, and increase business flexibility [2]. However, SOA lacks on support for high customization and systematic planned reuse. In other words, despite of the natural way of achieving customization in service-oriented applications, changing service order or even the participants of service compositions, services are not designed with variability to be highly customizable and reusable in specific contexts. In addition, service artifacts, e.g., specifications and models, are not designed with variability as well. Hence, these artifacts cannot be easily reused by a family of service-oriented applications [7].

Thus, SPL engineering, which has the principles of variability, customization and systematic planned reuse in its heart, can be used to aid SOA to achieve these benefits. In this path, service-oriented applications that support a particular set of business processes can be developed as SPLs [8, 9]. The motivation for it is to achieve desired benefits such as productivity gains, decreased development costs and effort, improved time to market, applications customized to specific customers or market segment needs, and competitive advantage [4, 10].

In this paper, we propose an approach for service-oriented product line architectures that combines SPL and SOA concepts and techniques to achieve high customization, systematic planned reuse and the desired benefits mentioned before.

Hence, the concept of managed variability and systematic planned reuse were introduced into service-oriented development activities. In order to deal with these concepts, the development process was divided in two life cycles as in SPL engineering [4, 11]. The first, *core assets development*, produces generic artifacts with variability to establish a production capability for applications. The second, *product development*, resolves the variation points of the generic artifacts produced in core asset development and creates applications customized to specific customers. Management at the technical and organizational levels during core assets and product development must be strongly committed to the success of the product line [12].

The reminder of this paper is organized as follows. Section 2 presents an overview of the approach for service-oriented product line architectures, and Section 3 describes its inputs, outputs and activities in details. A case study on the conference management domain is presented in Section 4. Related work is discussed in Section 5, and, Section 6 presents some concluding remarks and directions for future work.

2. Approach Overview

In this section, an overview of the approach for service-oriented product line architectures is presented. It is a top-down approach for the systematic identification, and documentation of service-oriented core assets supporting the non-opportunistic reuse of SOA.

The approach is based on the architectural style shown in Figure 1. This architectural style was adapted from [13, 14], which present a complete list of layers commonly used in SOA development. As mentioned, the architectural style is divided into layers, each of them with specific purposes as described next.

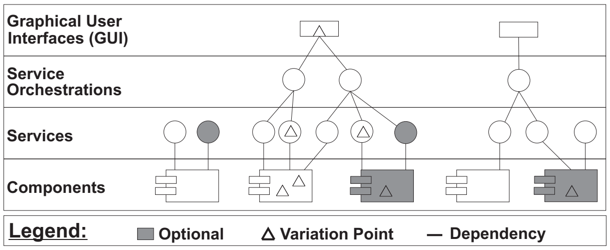


Figure 1. Architectural Style.

The *interface layer* is composed of Graphical User Interfaces (GUI) components. This layer may be used only by service-oriented product lines that require visual interfaces to interact with services and service orchestrations. The *orchestration service layer* consists of composite services, which implement coarse-grained business activities, or even an entire business process, that need the participation and interaction of several fine-grained services. The *service layer* is composed of self-contained and business-aligned services, which implement fine-grained business activities. Finally, the *component layer*, which consists of a set of components that provide functionality for the services exposed in the service layer and maintain their Quality of Service (QoS).

Note that the architectural elements (components, services, service orchestrations and user interface components) of these layers are developed with variability, and they can be mandatory, optional or alternative.

As mentioned previously, the approach is divided in two life cycles as in software product line engineering [4, 11]: *core assets* and *product development*. The *core assets development* aims to provide guidelines and steps to identify, document and implement generic architectural elements with variability. During *product development*, these architectural elements are specialized to a particular context according to specific customer requirements or market segments needs.

In this paper, we focus on the core assets development. In particular, on the design of domain specific architectures for service-oriented product lines. Thus, we provide guidelines and steps for the identification and documentation of components, services, service orchestrations and their flows using a top-down approach. In other words, the identification of architectural elements from existing legacy systems, the bottom-up approach, is not considered in this work. The following section presents the inputs, outputs and activities of the approach for service-oriented product line architectures in more details.

3. The Approach

The approach for service-oriented product line architectures starts with an identification phase. It receives the *feature model* and the *business process models* as mandatory inputs, and produces a list of possible components, service candidates and service orchestration candidates for the product line architecture. Thus, these architectural elements can be reused in all products of the line. This phase is separated in *component identification* and *service identification* activities.

Subsequently, there is a *variability analysis* activity. It receives the list of components and services identified previously, and defines and documents key architectural decisions regarding variability. In this activity, it is defined how the variability will be implemented within the services and components.

Architecture specification activity concludes the approach. In this activity, the architecture is documented using different views in order to represent the concerns of the different stakeholders involved in the project [15].

Figure 2 shows the inputs, outputs and activities of the approach for service-oriented product line architectures.

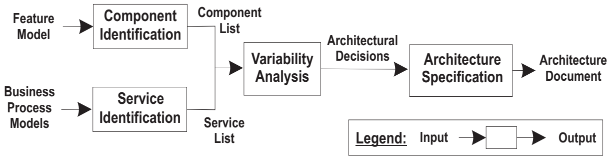


Figure 2. Activities of the Approach.

The next sections present the activities of the approach in more details. An initial case study clarifying and explaining these activities with examples is presented in Section 4.

3.1 Component Identification

In this activity, the components of the service-oriented product line will be identified. We consider a software component as a self-contained artifact with well-defined interfaces and subject to third-party compositions [16].

This activity starts with an analysis of the feature model to identify architectural component candidates. The purpose of this activity is to put features into modules (components) in order to design an architecture where components can be added or removed to generate customized products. Each of the modules identified in this activity will be an architectural component candidate for the service-oriented product line architecture.

In order to clarify this activity, we will use an alternative feature with two variants as example. In this case, each variant will be placed in a different component. Thus, the behavior of each variant can be put in a product by adding or removing one of the components. Since the features are alternative, only one of the components will be present in a product. However, in some cases, depending on the variability granularity, it may be appropriated to put both features in a unique component and add internal variability. This issue will be discussed in variability analysis activity in Section 3.3.

The components identified in this activity will maintain the quality of the services in the product line. Thus, identify components considering quality attributes, e.g., modifiability and reusability, is appropriated. However, some quality attributes, e.g., security and performance, will be responsibility of the service platform selected as well [17].

Existing software components can be considered for integration in this activity to increase reuse. The next section presents the service identification activity, which provides some guidelines and steps to identify service and service orchestration candidates.

3.2 Service Identification

The identification of service candidates is a challenging task of service-oriented computing [18, 19]. In the context of service-oriented product lines, service identification activity is even harder due to concerns with commonalities and variability.

In the service identification activity, a set of service and service orchestration candidates that support the business processes are identified. Thus, as the services are supposed to support the business processes, it is reasonable to identify them from the business process models [3, 5, 20].

This activity starts with an analysis of the business process models. In this analysis, the processes themselves, their sub-processes and business process activities are considered as service or service orchestration candidates, it depends on their granularity. Concurrently, key business entities are identified, and service candidates are created to implement their life cycle methods, e.g., create, delete, update and retrieve [3]. Finally, service candidates are defined to implement utility functionalities that support the services and service orchestrations identified previously, e.g., logging, monitoring and data transformation, when necessary.

We present a top-down approach for service identification, but it does not exclude existing services to be considered for integration during this activity. The service identification activity provides a service portfolio with all the service candidates identified as output. The next section presents the variability analysis activity.

3.3 Variability Analysis

According to [21], variability is the ability to change or customize software systems. Improving variability in a system implies making it easier to do certain kinds of customizations. Moreover, it is possible to anticipate some types of variability and construct a system in such a way that it is prepared for inserting predetermined changes.

At this point, the possible components, and the service and service orchestration candidates of the service-oriented product line have been identified. During the variability analysis activity, it is defined and documented essential architectural decisions about how the variability presented in the feature model and business processes will be implemented within services and components.

The variability analysis activity starts with an analysis of the component and service candidates identified. The similarities and differences among services should be analyzed with the purpose of reduce the number of service candidates. The similarity analysis consists of comparing the functionality of services in order to join similar services that implement fine-grained variability, e.g., variability that can be implemented by changing a class attribute or method. In this case, services will be joined in a single service with internal variability. The same analysis is realized among the component candidates. At this point, the services and components are no longer candidates anymore.

Subsequently, it is analyzed how the variation points will be implemented within the components. Component-Based Development (CBD) can be used as an implementation technique, i.e., each variant is implemented in a different component. Alternatively, well known variability implementation techniques can be used to implement component internal variability, e.g., aspect-oriented programming, conditional compilation, configuration files and design patterns [22]. The same thing occurs with the services. In this case, service orientation can be used as a technique to implement variability, i.e., each variant can be implemented in a service. It is the way the current service-oriented applications are customized, changing service order or even the participants of service compositions to implement variability. However, depending on the variability granularity it may be insufficient. A variation point can be implemented changing a class attribute, or a class, a method or even an entire component or service. Thus, in some cases it is necessary to introduce service internal variability.

In order to implement service internal variability, i.e., a unique service that can be customized to different purposes, the service interface, in some cases, must reflect the underlying variability the service contains in its components and classes. Thus, conditional compilation and parameterization can be used with the purpose of change service interfaces or modify the service behavior according to specific customer requirements. The use of code transformation tools is used in [17] to implement service interface variability.

Variability analysis activity produces as output a set of architectural decisions regarding variability that will be specified during architecture specification activity, which is presented in the next section.

3.4 Architecture Specification

In the architecture specification activity, the components, services, service orchestrations and their flows will be specified, i.e., the architecture will be specified. In this activity, the models and specification are produced with variability as all the artifacts of core assets development. Architecture specification requires notations with support for variability representation.

Software architectures are complex entities that cannot be represented in a simple one-dimensional fashion [15]. Since there are different stakeholders involved in a project with particular concerns about the system, it is important to represent the architecture upon different views.

During architecture specification, the first step is the definition of component and service interfaces. Subsequently, different architectural views can be produced: structural view, layer view, interaction view, dependency view, concurrency view and physical view. Each view is described in detail next.

The *structural view* represents the architecture static structure. This view shows the components, services and service orchestrations of the architecture. The *layer view* presents the services organized in their layers. The *interaction view* shows how the services and components communicate to realize a specific functionality. The *dependency view* presents dependence information among services and components. The *concurrency view* shows parallel communication among services and components, but it can be represented in the interaction view as well. Finally, the *physical view* shows how the services and components are distributed and the protocol of communication.

Some UML diagrams with stereotypes and variability extensions, such as [23, 24], can be used to create these views. As examples, the component diagram can be used to represent the structural view and dependency view of components, the interaction and concurrency view can be represented with sequence diagrams, and the dependency view of services can be created with interfaces, stereotypes and dependency arrows in class diagrams.

The next section presents a case study on the conference management domain using the approach.

4. Case Study

In this section, we introduce an initial case study on the conference management domain in order to clarify and explain our approach. The case study consists of a service-oriented product line that intends to produce customized service-oriented applications for the management of different conferences.

Part of the feature model of the service-oriented product line is presented in Figure 3, and its features are described next.

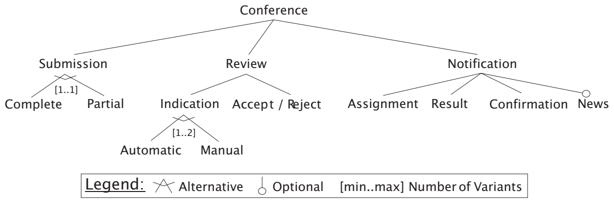


Figure 3. Feature Model.

- Submission: authors can submit their complete papers or, first submit the abstract, followed by the complete version. Complete and partial submissions are alternative features.

- Review: the indication of papers to reviewers can be made automatically and/or manually. Reviewers can also accept or reject paper indications. Automatic and manual indications are not exclusive, they can work together.

- Notification: the system can send information to reviewers about paper assignments. It can send acceptance or rejection (result) information to authors. It can also send event news, e.g., deadlines, and confirmation messages, e.g., paper or review submitted, to authors and reviewers. Event news notification is an optional feature. Assignments, confirmation and result notifications are mandatory.

Applying the technique for component identification, we finish with the following component candidates: *complete submission*, *partial submission*, *review management*, *automatic indication*, *manual indication*, and *assignment, result, confirmation and news* notification components. The complete and partial submission components were separated because they are alternative features, and only one of them will be bound to an application. The same thing for the automatic and manual indication components, which are an alternative non-exclusive choice, only one, or both will be present in an application. The variants and mandatory sub-features of the notification feature were also put each one in a different component. There are other components, e.g., access control, user management that were excluded from the paper due to space limitations.

Figure 4 shows the simplified paper submission business process. It starts with two optional activities, authors submit the abstract of the paper and receive a confirmation. Afterward, the authors submit the complete version of the paper and receive a confirmation again. Finally, after the reviews finish, the authors receive the result (acceptance and rejection) messages.

Figure 5 shows the simplified review business process. First, the system indicates papers to reviewers automatically and/or manually (chair indication). The reviewers receive the notification about the papers to review. They can reject or accept the reviews, and next, they receive a confirmation about the action they have performed. Finally, the reviewers submit their reviews and receive a confirmation again.

From these business processes, the following service candidates were identified: *abstract submission*, *paper submission*, *review management*, *notification* and orchestration services (*submit process, review process*) for the whole processes. The components of access control and user management mentioned above do not need to be exposed as services because they do not bring any business value to these business processes.

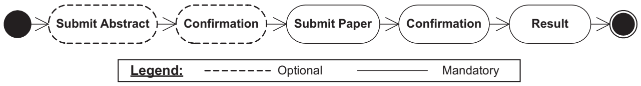


Figure 4. Submission Business Process.

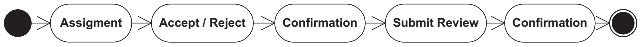


Figure 5. Review Business Process.

After the identification of the components and services candidates, we try to reduce the number of candidates defining how the variability will be implemented. For instance, the abstract submission and paper submission service candidates can be reduced to only one service. However, variability is introduced to the service interface in order to reflect that the service operation submit paper abstract is optional. The automatic and manual indication components, which assign papers to reviewers can be implemented in a unique component, but the variability should be introduced internally using SPL variability techniques, e.g., design patterns. In the case of the notification feature, its sub-features (assignment, results, confirmation and news) were put all in a unique component with internal variability because the variability granularity of these sub-features was low. We also use only one service with internal variability to exposed the notification component, however, this service also required interface variability in order to reflect that the news notification feature is optional.

During architecture specification, the architectural views are created. Figure 6 shows a dependency view of the orchestration service for the submission business process. As it can be seen, the submission service contains a variable operation in order to reflect the variability implemented in the partial and complete submission components. The same thing for the notification service, which has an optional operation as well to reflect that the news feature is optional. As another architectural view example, Figure 7 shows the interaction view of the submission process. The steps related with partial submission (submit abstract and its confirmation message) will be removed of the documentation when the feature complete submission is selected.

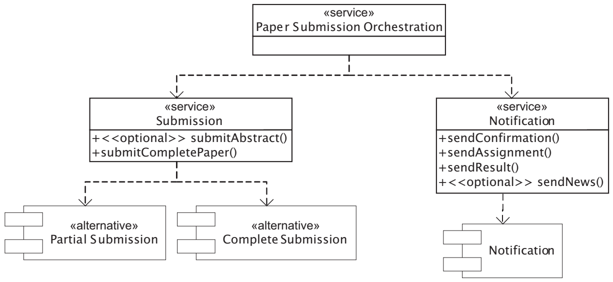


Figure 6. Dependency View.

5. Related Work

Two different approaches for business process modeling based on product line principles exploiting commonalities and variability through domain engineering are presented in [8, 9]. Both works realize processes able to adapt themselves to different customers or market segment needs. Thus, the resulting SOA systems that automate them will be

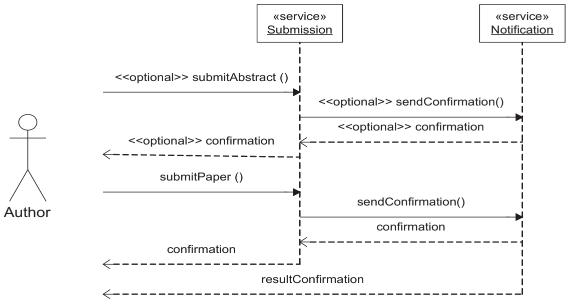


Figure 7. Interaction View.

suitable to different customer needs as the underlying processes. However, none of them concerns the identification of services candidates or gives information about the components that realize the service implementation. In addition, the work is not concerned with architecture specification and documentation, and focus on web service technologies such as BPEL. We solve these gaps in our work providing information on how to identify and specify services and components regarding variability issues. Moreover, our approach does not focus on any specific technology.

An approach for developing service-oriented product lines is presented in [18, 25]. It proposes a service identification method based on the feature binding analysis technique [26]. However, it does not consider the business processes and it may identify service candidates that are not aligned with the business goals. The service identification technique of our approach is based on the techniques used in service-oriented development [3, 20]. Thus, we identify services from an analysis of the business processes.

In [17], a development process for web services is proposed. It analyzes a particular software product line development process and compares it with the service-oriented product line process proposed. It concludes the paper with an example for a service-oriented product line web store that basically uses a code tranformation tool to implement service interface variability. In our work, we suggest some techniques for the implementation of service interface variability, not only the use of code transformation tools, but also well known techniques used in SPL, e.g., conditional compilation and parametrization.

6. Conclusions and Future Work

This work presents a contribution to the combination of SOA and SPL concepts. In particular, how these concepts can be used together to achieve desired benefits such as improved reuse, decreased development costs and time to market, and production of flexible applications customized to specific customers or market segment needs.

In order to achieve these goals, we presented an approach for service-oriented product line architectures that introduces the concepts of managed variability into service-oriented world and uses a two life-cycle model as in SPL engineering, however, only core assets development is considered in this work. These concepts were introduced in order to provide support for high customization and systematic planned reuse during service-oriented development. In this context, services are developed to be reused in specific contexts and service-oriented applications can be developed rapidly and customized according to specific customer requirements. We also present a case study on the conference management domain clarifying and explaining the activities of the approach.

As a future work, we are planning to apply this service-oriented product line architecture approach to others domains and validate the real benefits of the combination of SOA and SPL that we have used in this work. In addition, we are performing a case study using different technologies and techniques for service internal variability implementation in order to identify the real differences, if they exists, from object-oriented and component-based variability implementation.

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[1]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and Brazilian Agency (CNPq process number 475743/2007-5).

References

[1] F. J. v. d. Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] S. Carter, *The New Language of Business: SOA & Web 2.0*. IBM Press, 2007.

[3] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for developing service-oriented solutions," *IBM System Journal*, vol. 47, no. 3, pp. 377–396, 2008.

[4] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[5] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.

[1] INES - http://www.ines.org.br

[6] M. P. Papazoglou and W.-J. V. D. Heuvel, "Service-oriented design and development methodology," *International Journal of Web Engineering and Technology (IJWET)*, vol. 2, no. 4, pp. 412–442, 2006.

[7] A. Helferich, G. Herzwurm, and S. Jesse, "Software product lines and service-oriented architecture: A systematic comparison of two concepts," in *SPLC '07: 11th International Software Product Line Conference*, IEEE Computer Society, 2007.

[8] N. Boffoli, D. Caivano, D. Castelluccia, F. M. Maggi, and G. Visaggio, "Business process lines to develop service-oriented architectures through the software product lines paradigm," in *SPLC '08: 12th International Software Product Line Conference*, pp. 143–147, 2008.

[9] E. Ye, M. Moon, Y. Kim, and K. Yeom, "An approach to designing service-oriented product-line architecture for business process families," in *ICACT '07: 9th International conference on Advanced Computing Technologies*, pp. 999–1002, 2007.

[10] S. Cohen and R. Krut, eds., *Proceedings of the First Workshop on Service-Oriented Architectures and Software Product Lines, 11th International Software Product Line Conference*, 2007.

[11] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[12] L. Northrop, "Sei's software product line tenets," *IEEE Software*, vol. 19, pp. 32–40, July 2002.

[13] A. Arsanjani, "Service-oriented modeling and architecture," tech. rep., Service-Oriented Architecture and Web services Center of Excellence, IBM, 2004.

[14] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, "S3: A service-oriented reference architecture," *IT Professional*, vol. 9, no. 3, pp. 10–17, 2007.

[15] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practices*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[16] C. Szyperski, "Component technology: what, where, and how?," in *ICSE '03: 25th International Conference on Software Engineering*, pp. 684–693, IEEE Computer Society, 2003.

[17] S. Günther and T. Berger, "Service-oriented product lines: Towards a development process and feature management model for web services," in *SPLC '08: 12th International Software Product Line Conference*, pp. 131–136, 2008.

[18] J. Lee, D. Muthig, and M. Naab, "An approach for developing service oriented product lines," in *SPLC '08: 12th International Software Product Line Conference*, pp. 275–284, IEEE Computer Society, 2008.

[19] D. Kang, C. yang Song, and D.-K. Baik, "A method of service identification for product line," in *ICCIT '08: 3rd International Conference on Convergence and Hybrid Information Technology*, vol. 2, pp. 1040–1045, 2008.

[20] A. Erradi, S. Anand, and N. Kulkarni, "Soaf: An architectural framework for service definition and realization," in *SCC '06: Proceedings of the IEEE International Conference on Services Computing*, pp. 151–158, IEEE Computer Society, 2006.

[21] J. V. Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *WICSA '01: 2nd Working IEEE/IFIP Conference on Software Architecture*, p. 45, 2001.

[22] C. Gacek and M. Anastasopoules, "Implementing product line variabilities," *SSR '01: Symposium on Software Reusability*, vol. 26, no. 3, pp. 109–117, 2001.

[23] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley, 2004.

[24] M. Razavian and R. Khosravi, "Modeling variability in business process models using uml," in *ITNG '08: 5th International Conference on Information Technology - New Generations*, pp. 82–87, 2008.

[25] J. Lee, D. Muthig, and M. Naab, "Identifying and specifying reusable services of service centric systems through product line technology," in *SPLC '07: 11th International Software Product Line Conference*, IEEE Computer Society, 2007.

[26] J. Lee and K. C. Kang, "Feature binding analysis for product line component development," in *PFE '03: 5th International Workshop on Software Product-Family Engineering*, pp. 250–260, 2003.

RiSE

REUSE IN SOFTWARE ENGINEERING

DO MORE

www.rise.com.br

Towards an Approach for Service-Oriented Product Line Architectures

Flávio Mota Medeiros

Eduardo Santana de Almeida

Silvio Romero de Lemos Meira

Outline

- Introduction
- Variability Levels and Mechanisms
- Case Study
- Reuse Across Product Lines
- Conclusions and Future Work

Introduction

- Service-Oriented Architecture (SOA) [1]
  - » Develop distributed applications
  - » Self-contained, reusable and loosely coupled
- SOA Highlights [2]
  - » Align IT and business goals
  - » Increase reusability
  - » Flexibility to change (business agility)

General Scenario

- Systematic planned reuse
  - SOA reuse is not planned and systematic as in SPL [3]
  - Ad-hoc reuse strategy
- Artifacts reuse
  - Services are the reusable entities [4]
  - Diagrams and business process models
- High customization
  - Through variability [5]

Research Goal

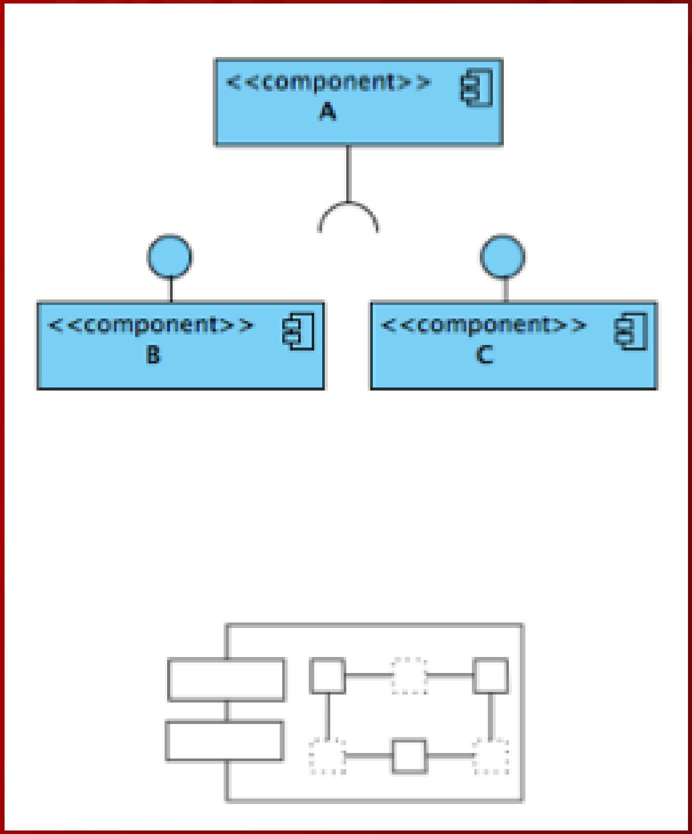
- Develop SPL with SOA
  - » Explore commonality and variability
  - » Domain-specific service reuse
  - » Artifacts with variability
- Raise reuse level
  - » Increase productivity
  - » Decrease development costs and time
- Service-oriented applications
  - » Systematic planned reuse
  - » High customization
  - » Customers or market segments needs

Outline

- Introduction
- Variability Levels and Mechanisms
- Case Study
- Reuse Across Product Lines
- Conclusions and Future Work

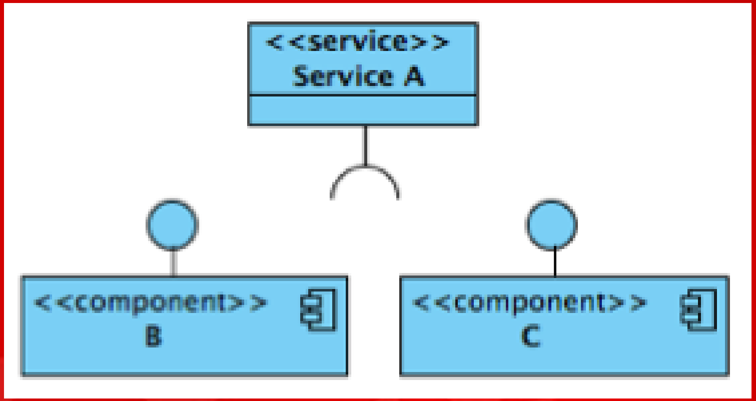
Variability Levels [6] [7]

- Configuration variability
  - » Select services or components
  - » Architecture variability
  - » High granularity
    - – Different classes
- Customization variability
  - » Introduce internal variability
  - » Customize component and services
  - » Low granularity
    - – Class attributes or methods



Variability Mechanisms

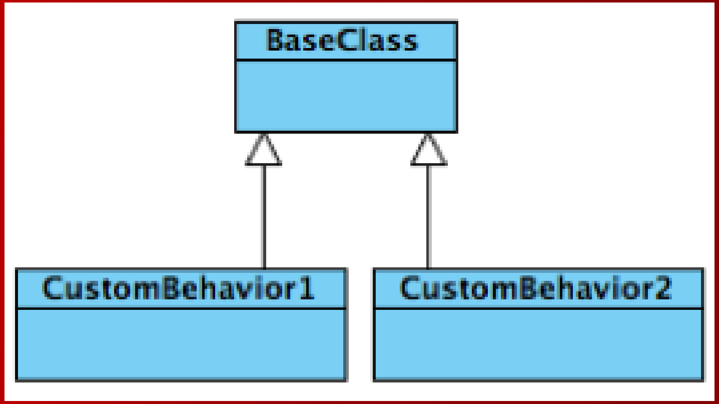
- Configuration Variability [8] [9]
  - » Dependency injection
  - » Parameters
  - » Configuration files
  - » Aspect orientation



```
If ( condition ) {
    //Binding Component B
    //Call service B
} else {
    //Binding Component C
    //Call service C
}
```

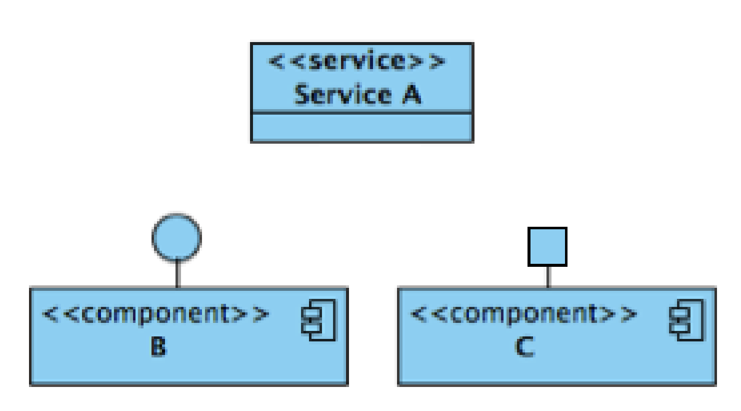
Variability Mechanisms

- Customization variability [10]
  - » Design patterns
  - » Configuration files
  - » Parameters
  - » Aspect orientation



Variability Mechanisms

- Configuration and customization variability together
  - » Select component B or C
  - » Component B and C with different interfaces
  - » Service A exposes different interfaces
  - » Aspect orientation



```
public aspect ComponentB {
      public void ServiceA.operationB(){
         // Implementation
      }
}


public aspect ComponentC {
      public void ServiceA.operationC(){
         // Implementation
      }
}
```

Outline

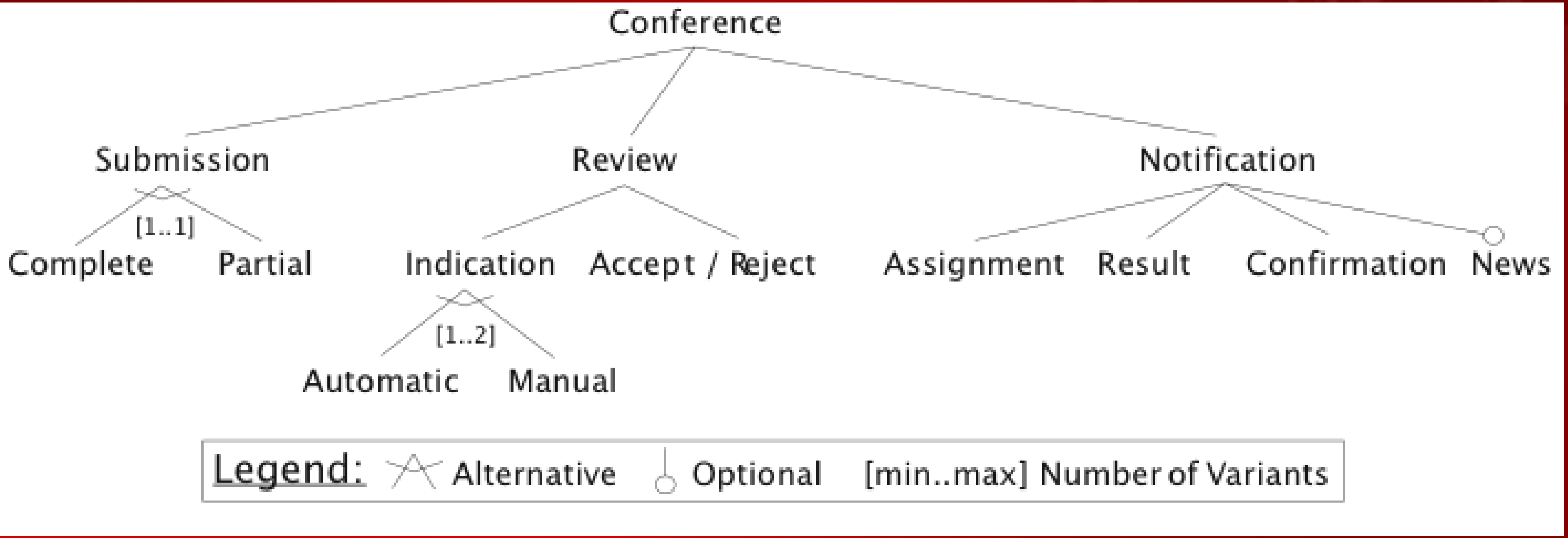
- Introduction
- Variability Levels and Mechanisms
- Case Study
- Reuse Across Product Lines
- Conclusions and Future Work

Case Study

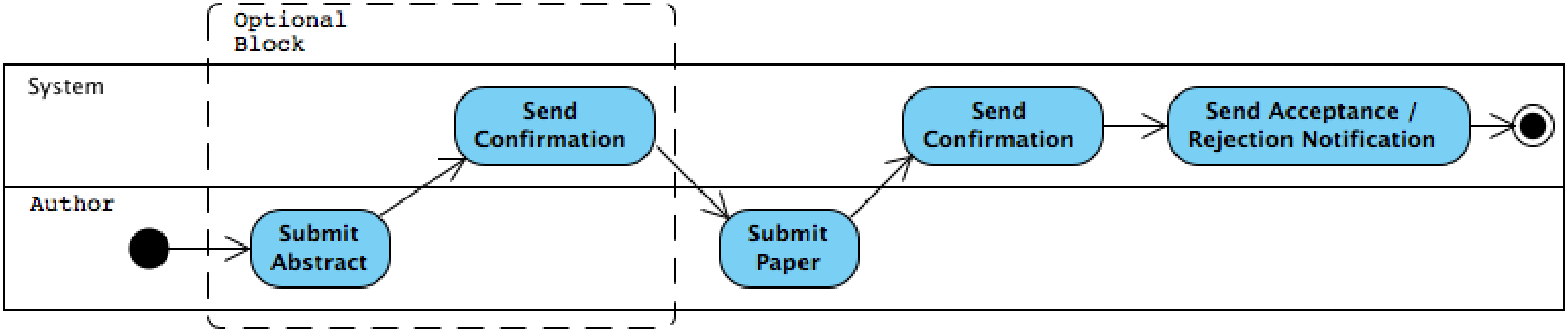
- Conference management
  - Service-oriented product line
  - Explore commonality and variability
    - Cyber chair, easy chair, IS Technology journal, etc
  - Customized according to the requirements

- SPL project
  - Full product line version
  - Reduced scope in the context of SOAPL

Scope

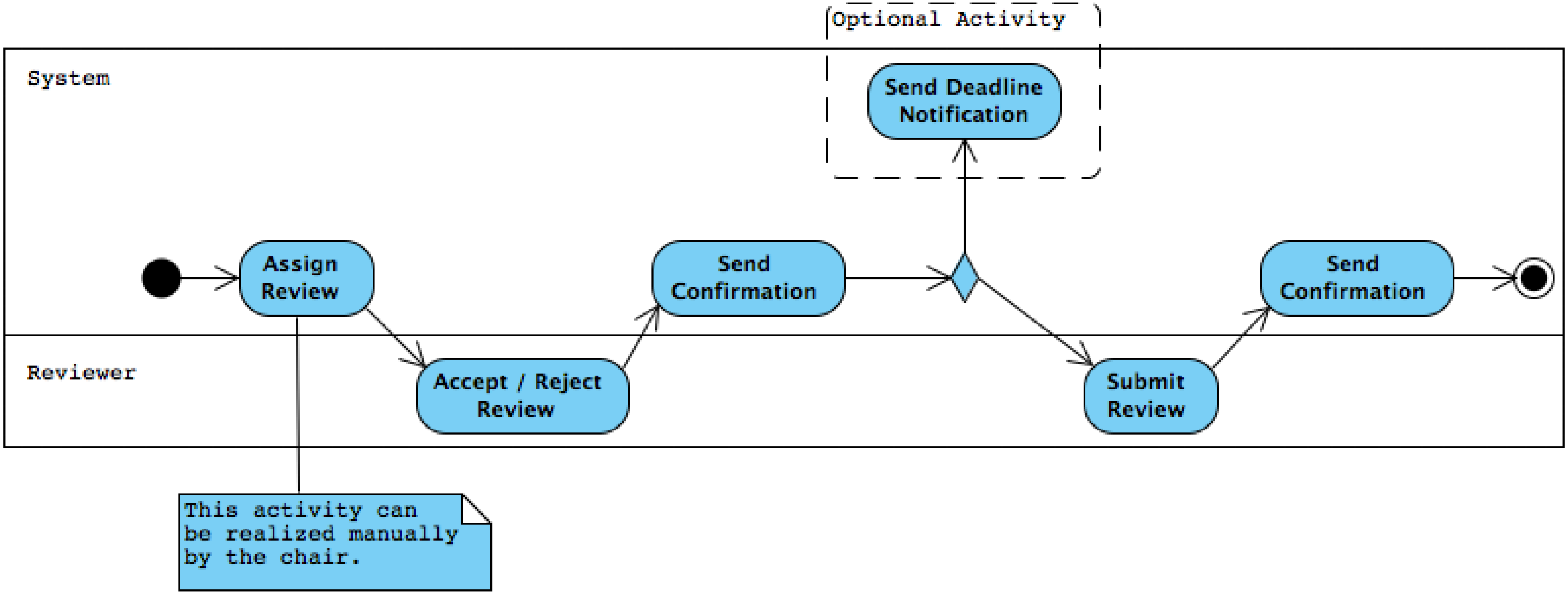

Conference
Submission
Review
Notification
[1..1]
Complete
Partial
Indication
Accept / Reject
Assignment
Result
Confirmation
News
[1..2]
Automatic
Manual
Legend:
Alternative
Optional
[min..max] Number of Variants

Submission Business Process



- Partial or complete submission
    - » Alternative feature

Review Business Process



This activity can be realized manually by the chair.

- » Event news notification (optional)
- » Manual or automatic assignments (OR)

Where we’re going

- Generate scenarios for users (submitter, reviewer, admin)
- Identify possible services
- Identify variations
- Enhance feature model
- Layer by services and workflows

Services

- Registration (author, reviewer, email confirmation/reregister)
- Scheduling (submission cut off, review cut off, reminders)
- Bidding/Selection/Assignment
- Submissions (abstracts, papers, reviews)
  - Uploads (browse)
  - Withdraw submission
  - Conflicts
- Notification (paper, review summary, conflict, Reviews complete)
- Format (administrator)
  - Paper (keywords, file, size, cycles (review, revise, final)
  - Review form
  - Review (criteria, levels, numbers of reviewers)
  - Knowledge areas

Services - 2

- Review summary
  - Accept/reject criteria
  - Conflict resolution criteria
- Storage/retrieval (papers, reviews, names)
  - tracking

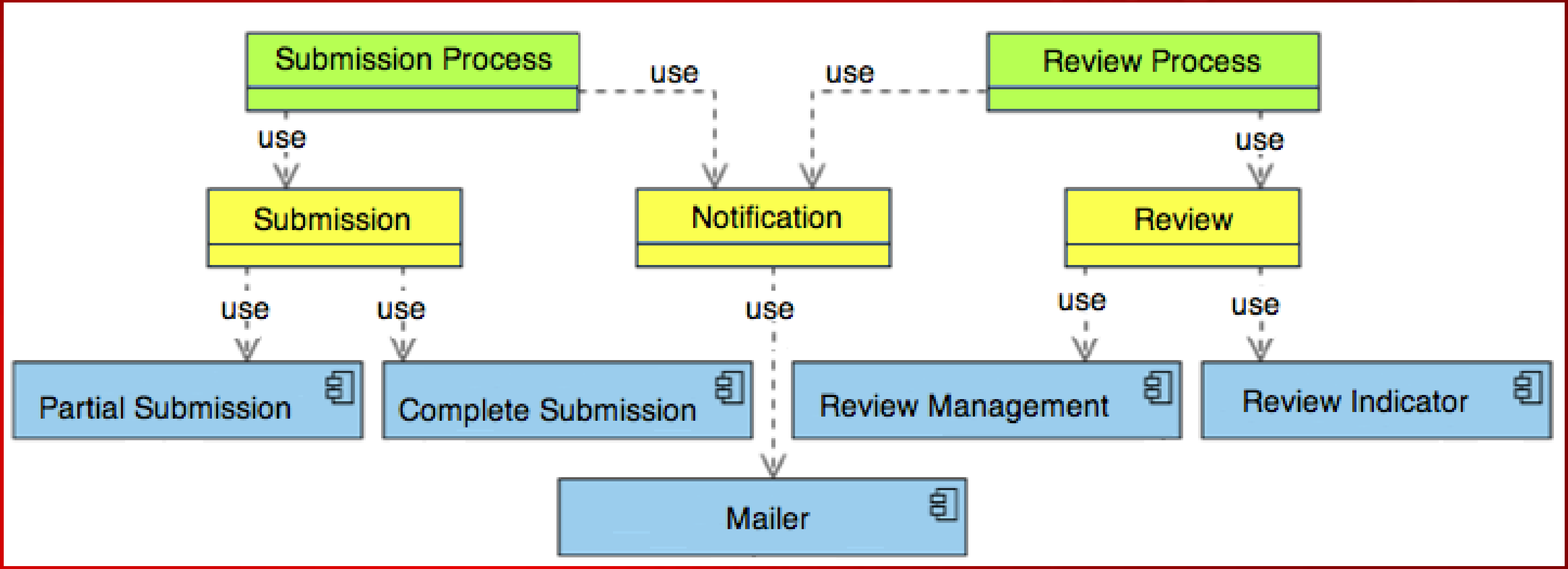
Submission

- Submit (alt: abstracts, papers, reviews, response by author to comments)
    - *– Medical tests*
- Meta data
    - » Conference identification
    - » Author anonymity (option)
    - » Conflicts
        - *– Patient, insurance,*
- Uploads (opt: browse)
    - » Word → .pdf
        - *– .pdf, images, audio, video*
    - » Submission window (cut and paste into window)
    - » Protocols (ftp, web,
        - *– Secure protocols, auditing*
    - » Format verifier, pages, words in abstract, file size, combine words/graphics in word count
    - » Format preview
    - » Via email
- Withdraw submission
- For review, as revised or final submission

Workflow -

- So far
  - » Author identifies paper *(Clinic identifies)*
  - » Submits paper *(Clinic submits)*
  - » System stores
- Next
  - » Provide summaries to reviewers → bidding service
  - » Place in queue for reviewer → start clock on review submissions
    - – *Place medical record in queue for reviewer*
  - » Allow for withdraws/updates

Architectural Elements



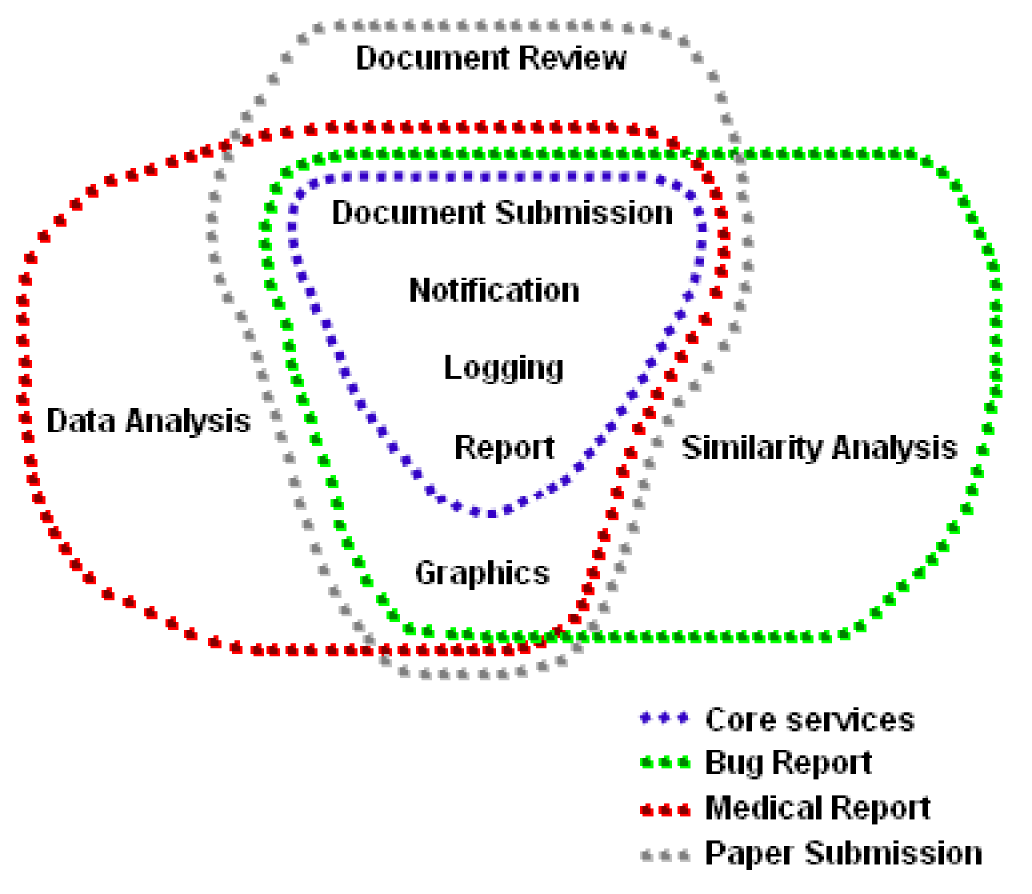
🟩 Orchestration Layer
🟨 Service Layer
🟦 Component Layer

Outline

- Introduction
- Variability Levels and Mechanisms
- Case Study
- Reuse Across Product Lines
- Conclusions and Future Work

Services for Different SPLs

- Domains
  - Paper submission
  - Medical record review
  - Trouble or Bug report
  - Item order report



Outline

- Introduction
- Variability Levels and Mechanisms
- Case Study
- Reuse Across Product Lines
- Conclusions and Future Work

Conclusions

- SPL principles can be used in service environments
  - Systematic reuse
  - High customization (variability)
- Service internal variability
  - Useful in some cases
  - Variability granularity is low
- Variability mechanism work in service environment
  - Design patterns, aspects and configuration files
  - Dependency injection and parameters

Future Work

- Evolve the case study
  - Analyze more variability mechanisms
  - Analyze different binding times

- Extend the SPL to Different domains
  - Bug report

Thank you!

Any questions or suggestions?

References I

- [1] Thomas Erl. “Service-Oriented Architecture: Concepts, Technology, and Design”, 2005.
- [2] Ali Arsanjani. “Service-Oriented Modeling and Architecture”, 2004.
- [3] Clements and Northrop. “Software Product Lines: Practices and Patterns”, 2001.
- [4] Mikko Raatikainen, Varvana Myllärniemi and Tomi Männistö. “Comparison of Service and Software Product Family Modeling”, 2007.

References II

- [5] Klaus Pohl et al. “Software Product Line Engineering: Foundations, Principles and Techniques”, 2005.

- [6] Eunsuk Ye, Mikyeong Moon, Youngbong Kim and Keunhyuk Yeom. “An Approach to Designing Service-Oriented Product-Line Architecture for Business Process Families”, 2007.

- [7] Nicola Boffoli, Danilo Caivano, Daniella Castelluccia, Fabrizio Maria Maggi and Giuseppe Visaggio. “Business Process Lines to Develop Service-Oriented Architectures through Software Product Lines Paradigm”, 2008.

References III

- [8] Mikael Svahnberg, Jilles van Gurp and Jan Bosch. “A taxonomy of Variability Realization Techniques”, 2005.
- [9] Sergio Segura, David Benavides, Antonio Ruiz-Cortés and Pablo Trinidad. “A Taxonomy of Variability in Web Service Flows”, 2007.
- [10] Cristina Gacek and Michalis Anastasopoulos. “Implementing Product Line Variabilities”, 2001.

Semantic Variability Modeling for Multi-staged Service Composition

Bardia Mohabbati[1,3], Nima Kaviani[2], Dragan Gašević[3]
[1]Simon Fraser University, [2]University of British Columbia, [3]Athabasca University, Canada
mohabbati@sfu.ca, nimak@ece.ubc.ca, dgaseavic@acm.org

Abstract

Feature models as the main modeling metaphors for software product line conceptualization are not expressive enough to cover all the variability needed to support adaptive engineering of service-oriented systems in highly dynamic environments. In particular, feature models lack required semantics to incorporate non-functional requirements (NFRs) and enable reasoning over the set of possible products in order to derive the best configurations. Ontology languages, as easily expandable semantically enriched conceptualization methodologies, can be used as the underlying languages for expressing feature models. This would allow augmentation of feature models with NFRs and would add the possibility for inference and reasoning to feature models. In this paper, we show how transformation of feature models to ontologies coupled with constraints over configuring products can help with reasoning over a product family and creating adaptive service compositions in an exemplified ubicomp application.

1. Introduction

The emergence of highly dynamic environments such as mobile systems and ubiquitous computing subsuming wide range of heterogeneous computing devices (e.g. handheld computers, PDAs, and smart phones), call for more flexible and adaptive development of software-intensive systems. Those systems now need to be composed, configured and delivered based on capabilities and resource constraints of the deployment platform of the target users. Accordingly, we require support to configure final software products, which provide the utmost functionality and satisfy non-functional requirements (NFRs) derived from the target deployment platforms.

Service-oriented Architectures (SOAs) come to the scene as a promising software architecture style for addressing the on-going challenges of the development of ubiquitous computing systems. In particular, platform independence, interoperability, loose-coupling, reusability, discoverability, composablity and dynamic binding are significant traits [2] for this context. However, to be able to software systems based on the SOA principles in the provided ubiquitous computing context, we need to equip developers with appropriate software methodologies, which can allow for effective software development process. This process involves the development throughout different abstraction layers comprising such as those already identified in the literature business process, service composition, service interface, and service implementation layers [1][3][4]. While each of the abovementioned layers calls for a lot of dedicated research, in this paper we focus on one key problem: How to develop service-oriented systems by considering the specificities of different target deployment platforms. That is, more concretely, how can we consider different device capabilities in the service-oriented development process? Given a proven track record of Software Product Line Engineering (SPLE) in the domain of mobile computing (e.g., Nokia as one of the most known examples), the idea of the use of software product line principles seems to be a first natural option for the problem under study. The SPLE discipline provides methods for managing variability and commonalities of core software assets in order to facilitate the development of families of software-intensive products. Here, software families are characterized by a set of features shared by each individual product of a family. At the same time, each specific product may have certain specificities coming from particular requirements of the product at hand. While in general we may have various different sources of variability, in this paper, we exclusively focus on the problem of variability coming from the different delivery platforms of service oriented systems. For example, two different types of mobile devices might support different connectivity or security protocols. This may directly impact a decision on which payment service to use in a concrete service-oriented system under development.

As it is reflected in the previously-mentioned four layers of abstraction reflects, the prevailing approach to developing service oriented architectures is based on business processes modeling. Once defined, business process models guide the process of service composition. This is the exact place where we take in to account of applying SPLE principles. In particular, we exploit feature modeling, which allows for identifying and managing the existing services and components in terms of common and variable features of a system in a product line. Furthermore, feature models provide structural management scheme of variability derived from NFRs of domain assets and to approach semi-automatic configuration and generation of products in response to the specific requirements for different products of the same service-oriented product family.

Aiming to provide a methodology, we propose a multi-staged specialization process of feature models [5]. In this process, we model service compositions by using feature modeling diagrams, which are then annotated with the NFRs, which each deployment platform needs to satisfy. In this process, we make use of the Ontology Web Language (OWL) and Semantic Web Rule Language (SWRL)[6][7]. These languages are used to represent feature models formally, so that we can automatically detect a set of allowed feature model specialization of a given target device. In addition, through using expandability and annotation capabilities of ontologies, NFRs of service in product lines can be formally incorporated into the ontological specification of feature models, expanded over time, or augmented with additional non-functional ontologies. In other words, the variability derivation points derived from NFRs of services is catered by the specification of the relevant declarative elements (device ontologies, service descriptions). Description Logic as the underlying logic for ontologies renders non-functional requirements to be formally introduced into the feature model ontologies. Furthermore, ontology-based semantics support logical reasoning over semantic descriptions, which in turn helps with validating (non-) functional requirements, intelligent product configuration, and product consistency check.

2. Motivation Example

Dealing with diverse services offered by various service vendors requires a proper mechanism to select appropriate services whose composition can lead to a desired functional system. The architectural structure for developing a system is thus influenced by possibilities in choosing the appropriate set of services from the list of existing ones. An adaptable design hence, should lay out all different possibilities for composing services in order to enable a system detect, find, and replace its set of services when needed. Consider an application developed to be launched on a large display at an airport to enable passengers look for information about flight schedules, stores at the airport, the city, etc. In an ideal ubiquitous environment, the application should enable passengers to connect to the application seamlessly and utilize its functionalities.

A common method of interaction would be through using cell phones carried by passengers. The application should enable every passerby to use his or her cell phone to connect to the application regardless of the type of phone s/he carries. Interaction and communication with the application on the large displays can happen through different communication protocols (e.g., WiFi, Bluetooth, IrDA), and different message exchange protocols (e.g., Http over TCP, Sockets, Channels, etc.). Consequently, the large display application needs to be able to adapt its communication and interaction protocols to the type of phone requesting a connection. Furthermore, the content delivered to the phone should also be adjusted to the physical specifications of the device (e.g., the display size, the resolution, etc.). The problem gets more complicated if part of the presentation or logic for the application needs to be delivered to the mobile phone. Adaptability becomes an important issue, since the application needs to incorporate different modules depending on the infrastructure for the mobile phone, while keeping the functionality consistent from phone to phone. Being able to properly lay out a design architecture that brings all these diversities in selecting components under a unified model, would facilitate selection of components through reasoning and would enable proper adaptation of the software with respect to the diversities imposed by joining and leaving mobile devices. We believe the support for specifying variabilities and commonalities provides proper modeling requirements to deal with the dynamicity required in ubiquitous environments.

3. Proposed Approach

As indicated in the introduction, the goal of this paper is to propose a methodology for developing families of service-oriented systems based on a multi-staged specialization of feature models. Figure 1 illustrates the overview of the overall development process. The first two stages adapt the existing feature-oriented domain analysis and design approaches (i.e., *Domain Analysis and Design*). Besides their adaptation of the service-oriented context, this stage also specifies NFRs related to the capabilities of the target deployment platforms. Once defined, the next step is to provide a specialization of the feature model first based on the target deployment platforms and second based on user preferences (i.e., *Service Adaptation* in Figure 1).

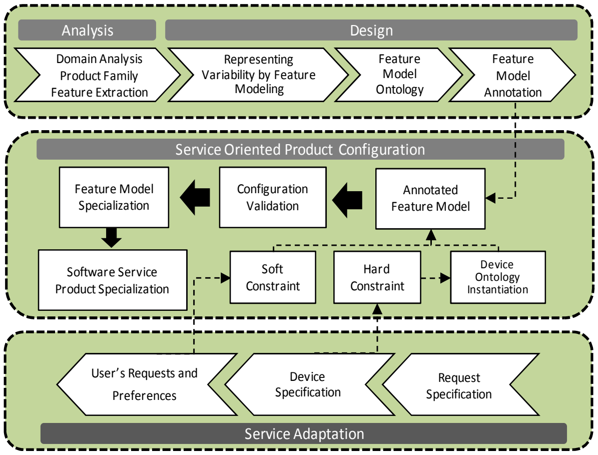

Analysis
Design
Domain Analysis Product Family Feature Extraction
Representing Variability by Feature Modeling
Feature Model Ontology
Feature Model Annotation
Service Oriented Product Configuration
Feature Model Specialization
Configuration Validation
Annotated Feature Model
Software Service Product Specialization
Soft Constraint
Hard Constraint
Device Ontology Instantiation
User's Requests and Preferences
Device Specification
Request Specification
Service Adaptation

Figure 1. The overview of the steps of the multi-staged specialization of service compositions

In this paper, we only focus on the first step, where we are considering the characteristics of the target deployment platform. In the overall process, we make use of the semantic Web languages for rules and ontologies (OWL) and SWRL to automate the process of specialization process and obtain a set of feature model specializations for a given set of the requirements (i.e., *Service-oriented Product Configuration*). The rest of the section describes the proposed approach in detail.

3.1. Analysis and Design

The purpose of analysis and domain engineering is to develop domain assets, aka features, to identify and extract a set of reusable features involved in the process of service-oriented product family. There have been many methodologies established to perform domain analysis. These methodologies could be opted in the engineering of reusable architectures and components (e.g. Feature-Oriented Domain analysis (FODA) [10], Feature-Oriented Reuse Method (FORM)[11], FeaturRSEB [12]). An appropriate comprehensive feature model should be designed to represent the system variability and commonality through the result of domain analysis. We employ an ontology-based approach to incorporate the NFRs of a system into the specification of features by transforming the feature model of a software system and constructing its feature model ontology. The annotation processes is performed to enrich feature models by associating semantic metadata though using NFR ontologies; which yields how features can contribute to the satisfaction of high-level abstract objectives of the problem domain. In our use cases, feature models are annotated using the device ontology; which contributes to finding a set of configurations of software services supported by the device capabilities.

3.1.1. Feature Models for Variability Modeling

Software Product Lines (SPL) provides an effective approach for modeling variability. SPL empowers the derivation of different product family applications by reusing the realized product family assets often known as core assets. A key principle of SPL is maintaining the variation points and dependencies in order to facilitate the exploitation of commonality and the management of variability among software features. SPL practices can be utilized to support service-oriented applications to promote the reusability and adaptation of services in product families of software services. Consequently, SOA's promise of bringing reusability and loose coupling can be further augmented using SPL engineering. The features in SPL subsume generic architecture and components which are tightly coupled, whereas services in SOA are reusable loosely coupled units [4]. Treating service units in SOA as core assets of SPL allows both perspectives on reuse to be incorporated synergistically into an integrated approach. Feature modeling in SPL is a well-defined approach to capture variability and commonality of the features and allows for expressing the permissible variants and configurations of software product family. Accordingly, the large set of existing services and their shared commonalities (particularly in the domain of ubiquitous computing) makes it possible to take advantage of SOA and feature modeling in SPL engineering for modeling variability of services.

a) Feature Model

As alluded to above, the feature modeling technique is opted to represent and describe the possible configurations of system and variants in terms of features representing system functionality units. In general, there are four types of relationships related to variability concepts in the feature model. They can be classified as: *Mandatory (Required), Optional, Alternative and Or feature group*. Mandatory features must be included in the description of their parent features and presented to function as intended. Optional features may or may not be included for the basic product to function. Alternative features indicate that only one of the features from the feature groups can be used to provide that the proper feature functionality. Figure 2 depicts the graphical representation of feature models, known as feature diagram, as well as common feature diagram notation [18][5]. The sample feature model diagram, in a three-like graph structure where primitive features are leaves and compound features are interior nodes, introduces parts of the system that require selecting and composing appropriate services for delivering particular contents based on users' requirements, functionalities of the system, and capabilities of end-point devices. The system, through conducting service adaptation, should be able to identify the capabilities of the required device, adjust the content according to device features, and select appropriate services by looking into the feature model and extracting those that have matching functionalities. Such kinds of scenarios clearly illustrate the use of SPLE in the SOA development, especially in the domain of ubiquitous computing.

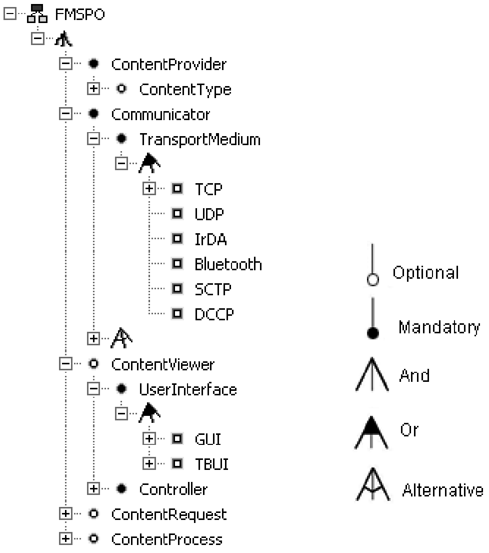


Figure 2. Feature model diagram

Feature models explicitly demonstrate different ways in which features could be composed. A valid composition of features is called a *configuration* which in turn is a valid software service product specialization. Since different applications employ features in different non-functional context, feature models help to segregate the NFRs of services from functional requirements in SOA which results in the increasing reusability of services by defining constraints among NFRs [9].Feature models enable the management of variabilities and commonalities for the set of existing services and components. Extending feature models to support non-functional or extra-functional requirements allows QoS/NFRs-driven service selection and composition. These requirements can be checked and verified against the non-functional properties interlaced with service specifications corresponding to the underlying components. In the following sections we discuss how ontologies can be exploited to extend feature models with NFRs and enable configuration of desired services based on user requests.

b) Semantic Feature Model

Current feature models mainly consider the modeling of functional features, and there is a lack of precise modeling artifacts dealing with NFRs. For instance, despite existence of several proposed approaches that bring SPL into the domain of pervasive and ubiquitous computing [13][14], most of them lack a clear specification of NFRs for service level agreement (SLA), QoS, and device capabilities [15]. Furthermore, they hardly provide possibilities for consistency check of the NFRs extended feature models, or logical reasoning over feature models which enable intelligent configuration and selection of services for product instantiation. Moreover, due to the lack of formal semantics and descriptions in a feature model, relations, dependencies and constraints of features are not specified comprehensibly through the model. Accordingly, we have employed an ontology based approach aiming at creating semantically-enabled feature models. We believe ontologies provide the appropriate means to address the aforementioned issues.

3.1.2. The Feature Model Ontology

An ontology is defined as a formal specification of a conceptualization and utilizes the representation of knowledge contained in feature models. The Feature Model Ontology (FMO) provides the semantic representation of features and relations and dependencies as well as feature constrains, which express another form of relations among features, in one model entirely. We used the approach introduced by Wang et al.[7][8] for mapping feature models to ontologies using the Web Ontology Language (OWL) [6]. The transformation of a feature model is performed through constructing mutual disjoint classes corresponding to defined nodes in the feature model and assigning a class rule constructed for individual classes created in the FMO. In other words, each class rule is associated to its corresponding feature node declaring an existential restriction and is used to define the bindings of the node's child features or to define the constraints. In our OWL model of FM, an object property is created whose asserted range is the respective feature class for the feature node. Assuming that there are *i* nodes in the feature model (FM), the Description Logic (DL) presentation of the above modeling can be expressed as follows:

$F_i \sqsubseteq \top$
$FM_iRule \sqsubseteq \top$
$hasF_i \sqsubseteq ObjectProperty$
$FM_iRule \equiv \exists\, hasF_i.\, F_i$
$F_i \sqsubseteq \neg F_j$, *for* $1 \leq 1, j \leq n \wedge i \neq j$

Going back to the scenario of Section 2, let us model the need for supporting Bluetooth communication for the large display application. The DL presentation of for including Bluetooth into the feature model for our product family can be defined as follows:

$Bluetooth \sqsubseteq \top$
$BluetoothRule \sqsubseteq \top$
$BluetoothRule \equiv \exists\ has\ Bluetooth.Bluetooth$,
$hasBluetooth \sqsubseteq ObjectProperty$

3.1.3. Feature Model Annotation

Representation of a feature model as an ontology enables us to take advantages of ontology annotation capabilities in order to enrich the definition of domain assets with the constraints concerning the non-functional requirements of a domain. Through using expandability property of ontologies, NFRs can be formally incorporated into the ontological specification of feature models, expanded over time, or augmented with additional NFR ontologies. For a feature model to explicitly integrate NFRs to the possible set of configurations, at the design stage, it would be possible to annotate the feature model with an ontology representing NFRs of interest. That is, the ontology of service quality can be used to annotate features in the feature model with attributes such as precision, robustness, reliability, or any other relevant NFRs. We extend the FMO by defining *AnnotationProperties (ANs)* as part of our model so as to annotate the feature model using external ontologies. In essence, the ANs have feature classes as their domains and their ranges refer to the concepts (i.e., classes) in the NFRs or QoS ontologies, For the sake of the detailed analysis of the ubiquitous domain and the paper's research objective, we will use device capability ontology as a sample for describing the NFR ontology. An example of such an ontology is the W3C's Delivery Context Ontology [23]. This process and assertion of ANs' range is performed during designing the system and the existing services. The attributes are added to the feature model as part of the requirements for each feature and its corresponding service interface. Considering that we refer to the classes in FMO as FM_m and the classes in device capability ontology as DC_n, annotating FMO with the device capability ontology using an OWL object property, detonated as AN, is carried out as demonstrated in Figure 3 which shows annotation of the feature model using derived device ontology following the DL syntax as below:

$AN \sqsubseteq ObjectProperty$
$AN \sqsubseteq FM_m,\ domain(FM_m)$
$\top \sqsubseteq \forall\ AN.DC_n,\ range\ (DC_n)$

Considering the Bluetooth communication example, in order to relate the Bluetooth concept in the FM with the Bluetooth communication device provided by a mobile phone, the following DL associations are used:

$hasBluetooth\_AN \sqsubseteq ObjectProperty$
$hasBluetooth\_AN \sqsubseteq Bluetooth,$
$domain(BluetoothRule)$
$\top \sqsubseteq \forall\ has\ Bluetooth\_AN.BluetoothDevice,$
$range\ (BluetoothDevice)$

Due to space limits and the large collection of mappings between the device capability ontology and the feature model, we do not include a complete representation of these mappings. Every configuration instance generated from the feature model also inherits these ontology attributes with asserted values specifying the set of potential capabilities for an NFR to be satisfied.

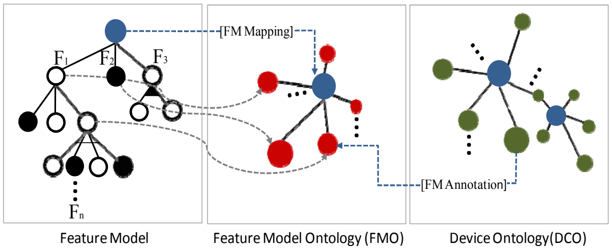


Figure 3. Feature Model Mapping and Annotation by NFRs Ontology

3.2. Service Requirement Specification

The process of selecting appropriate features for a configuration of service-oriented products is influenced by the *requests specification*, which encompasses requirements for a product configuration. In particular, we start from the definition of the *hard NFRs*. The NFRs are embedded into the requirements specification which in this case are catered by the capabilities of the target deployment platforms such as mobile devices. For example, hard NFRs describe characteristics of the device capabilities for video streaming and multimedia processing in the target deployment platform. In our approach, these hard NFP are added into an OWL knowledge base as instances of the device ontology of choice. The following ontology fragment expresses specification of a concrete delivery platform. The definition requires a particular characteristic supported by the target device display. For example, an instance of the device should be equipped with a display feature which supports specific values (i.e., x and y) for the width and height of the display along with the component of the requested aspect ratio.

$$\begin{array}{l}
Device \sqsubseteq \top \\
Hardware \sqsubseteq \top \\
hasHardware \sqsubseteq Device, \\
\top \sqsubseteq \forall\ hasHardware.Device \\
Display \sqsubseteq Hardware \\
\top \sqsubseteq \forall\ hasDisplay.Display \\
AspectRatio \sqsubseteq Display \\
\top \sqsubseteq \forall\ displayAspectRation.AspectRatio \\
aspectRatioHeightComponent \sqsubseteq AspectRatio \\
aspectRatioWidthComponent \sqsubseteq AspectRatio \\
\top \sqsubseteq \forall\ aspectRatioWidthComponent.int \equiv x \\
\top \sqsubseteq \forall\ aspectRatioWidthComponent.int \equiv y \\
NFR_H \geq x \quad , \quad NFR_W \geq y
\end{array}$$

In our overall process, which is not discussed further in this paper, we also anticipate the place for the possible preference of users defined as *soft requirements*. Examples include cases like preference towards services with textual services vs. multi-media service or precision of retrieval services vs. speed of retrieval services. Soft requirements represent user preferences unlike hard requirements, which must always hold.

3.3. Software Service Products Configuration

Ontology-based modeling of the target device reflects all the capabilities of the target device. In other words, device ontology instances reflect all the NFRs as hard constraints which exclude and include the selection of services in annotated feature models. So in the process of configuration, a subset of services is selected based on hard constraints specified by device capabilities, known as *software service product specialization*. In the process of configuration, the model of the target device requires to be compliant with the annotated feature model. The features which do not satisfy the NFRs of target device are discarded pruning the feature model into a new feature model whose set of possible configurations is a subset of the original feature model. This derived feature model is referred to as a *specialization of the feature model* and the staged refinement process which constitutes *staged configuration* [17]. In terms of the OWL-based reasoning, our goal is to determine if instances of the NFRs ontology (i.e., in our case device capabilities) are consistent with respect to the annotation properties defined in the feature model. The DL-based ontology that we have discussed in our scenario comprises two knowledge bases, Feature Model Ontology ($\mathcal{FMO}$) and Device Capability Ontology ($\mathcal{DCO}$); which is denoted as $\mathcal{O} = \mathcal{FMO} \sqcup \mathcal{DCO}$. Our rule knowledge base is a quadruple $\mathcal{K} = (\mathcal{O}, \mathcal{T}, \mathcal{A}, \mathcal{R})$, where $\mathcal{T}$ is a set of concept axioms (TBox), $\mathcal{A}$ is a set of assertional axioms (Abox), and $\mathcal{R}$ is a set of rules written as inclusion axioms. Lets us provide some definition to build the ground of the feature model specialization process.

DEFINITION 1 *Let $d \in \mathcal{DCO}$ be an instance of a device which has n capabilities. Each capability for device d is an instance (i_k) of a concept (C_k) from $\mathcal{DCO}$ such that $\mathcal{A} \vDash C_k(i_k)$ ($1 \leq k \leq n$). $S_{dc} = \{C_1,..,C_n\}$, represents the set of all concepts C_k from $\mathcal{DCO}$ that d supports.*

DEFINITION 2 *Let $S_{AF} \sqsubseteq \mathcal{FMO}$ be a set consisting of concepts CF_i such that $\forall CF_i.\top \mid \exists AN_i \sqsubseteq CF_i \sqcap \exists AN_i.C_k$ where $C_k \in \mathcal{DCO}$.*

The following Algorithm is introduced to specialize features from annotated feature set w.r.t hard constraints. The algorithm initially checks if each feature from S_{AF} has an annotation property AN_j (I) whose range is a class from S_{dc} (II). Those features whose properties do satisfy this condition are removed from the feature model specialization set (*SFA*). Each feature which satisfies the above conditions is further checked to see if there is at least one consistent instantiation of it in the $\mathcal{FMO}$ taking the value from $C_i(i) \in \mathcal{DCO}$ through using the annotation properties (III). Otherwise they are also removed from *SFA*.

Algorithm 3.1: Feature Model Specialization

```
$AN_j \sqsubseteq ObjectProperty$
$SFA = \bigcup_{i=1}^{n} CF_i$
for each $C_k \in S_{dc}$
 for each $CF_i \in S_{AF}$
   if ($\exists AN_j.C_j \sqcap AN_j \sqsubseteq CF_i$ and $C_j \in \mathcal{DCO}$)   (I)
     if ($C_j \in S_{dc}$)   (II)
       if ($\mathcal{A} \nvDash CFi(i_j)$ )   (III)
         $SFA \leftarrow SFA - \{CF_i(i_j)\}$
     else
         $SFA \leftarrow SFA - \{CF_i(i_j)\}$
 return ($SFA$)
```

Having the set of compatible features remained in the feature model; the process of composition proper of services proceeds to derive a set of suitable software service products for the requesting devices. Consistency checking and validation of a given software service configuration results from the feature model specialization and the process of the staged refinement is required to be performed in order to validate the configuration against the feature model constraints. During the process of the configuration validation, model constraints are checked against the derived configuration to provide proper assurance in terms of correct exclusion and inclusion of optional and mandatory features. To perform consistency checking and analysis over the OWL representation of an annotated feature model, we employ RacerPro[2] as one of the widely accepted OWL-DL reasoning engines, which supports automated class subsumption, consistency reasoning and detection of possible inconsistencies in the specialized feature model. Since some inconsistencies, derived from mutually exclusive properties (require and exclude), could not be represented by relying solely on OWL DL, due to expressivity limitations, we formulate and define a list of SWRL rules to represent all invalid states and detect conflicts or inconsistencies in the model.

3.4. Service Discovery and Specialization

The result of consistency checking in the previous step can be two folded. If the result is an inconsistent ontology (i.e., NFRs filtered out some mandatory features), we need to return to the feature analysis stage and design and refine our family of compositions to satisfy the discovered inconsistencies. Otherwise, if the result of this step is a consistent feature model specialization, we enter the process of further specialization and service discovery. The service specialization can be based on soft requirements (e.g., preferences towards certain features) of the stakeholder. Analyzing soft requirements can also be done through a similar specialization process as we have proposed for hard requirements. However, in this process, we are also considering the use of fuzzy logic [24]. Once the final configuration is obtained in which all the variability is resolved, we start the process for the final generation and deployment of the service-oriented system. We use the Web Service Modeling Ontology framework. More specifically, for the obtained feature configuration, we generate a complete description of the WSMO service compositions. This transformation is done by following the mapping rules between the feature models and abstract state machines defined in [25], while a complete implementation of the transformation is available in [26]. In our transformation, we generate all elements of the WSMO specification including, capabilities, pre- and post-conditions, transition, choreographies (along with state signature and transition rules) and orchestrations. In fact, during our project on transformations between WSMO services and feature models, we realized that in order to be able to generate complete WSMO service compositions, we need information about non-functional properties and information about ontological grounding of each feature. Therefore, our process of feature annotations, presented in this paper, perfectly complements the process of generation of complete WSMO service descriptions, We have deployed and tested the obtained services with the WSMX toolkit for discovery, mediation and composition of WSMO services [1].

4. Related Work

Wang et. al [8] provide a methodological approach to verify feature models using the Web Ontology Language. They transform feature models to ontologies by converting features to pairs of concept and rule classes with each pair presenting a feature in the feature model. This transformation coupled with constraints over the relations between the class nodes enables reasoning over the consistency of the ontology, and consequently helps with verifying the validity of the feature model and the instantiated products. Weis [19] considers pervasive applications as features that can be customized based on personal preferences. Users define configuration of applications using a graphical language with support for detailed customization at the price of increased complexity in using the language. Their approach provides a coarse approach to service composition in Pervasive environments.

One of the main issues with respect to feature models is constraining and controlling the variability of features. This is typically done by placing constraints across the feature hierarchies. These features may be represented in the form of propositional logic [18] or richer formalisms such as first order predicate logic [20] and its variants. Object Constraint Language (OCL) [21] is also used when feature models are represented in UML diagrams. Forfamel uses Weight Constraint Rule Language (WCRL) [22] to constrain the representation of features in its ontology. As a whole, the constraints over the variability enable reasoning and resolution of features depending on the configurations required for each product member or each customer of a product line.

Lee et al. [27] propose a feature-oriented product line approach for SOA which guides developers through the composition of services in a feature model. An approach to the generation of business process models in BPMN from feature models is introduced in [25]. This work inspired our transformation between feature models and WSMO service compositions.

5. Conclusions and Future works

In this paper, we have proposed a framework for development of service-oriented architectures based on the principles of multi-staged feature model specializa-

[2] http://www.racer-systems.com/

tion. The key part of the proposed approach is the use of ontologies as an underlying formalism for representation of feature models of families of service compositions. Once created, such ontology-based feature models can be easily annotated with the non-functional properties. The critical role of ontology-based reasoning takes place in the process of specialization of the annotated feature models through a set of specific requirements, which must hold in a particular service-oriented application at hand. In our concrete case, we experimented with the non-functional requirements, which specify the capabilities of the target platform for executing service compositions. The ontology-based consistency checking approach combined with the integrity constraints defined in the Semantic Web Rule Language are used in this step to discover a feature model specialization satisfying the requirements for the final system to be built (in our case those are device capabilities). Once a requested feature specialization is obtained, we go through an interactive process where further, stakeholder soft requirements (i.e., preferences) are used to obtain a feature specialization without variability. Such a specialization is then translated to a composition represented in WSMO, where the use of ontologies again plays a critical role.

In our future work, we are going to conduct an extensive experimental study to measure the effectiveness of our proposed methodology. Also, we will further explain our approach for the use of soft requirements in the process, once the hard requirements are satisfied in the service composition process. Finally, we will fully explain the process of deployment to the WSMO SOA framework.

Acknowledgement. This research was in part supported by Alberta Ingenuity through the New Faculty Award program and by Athabasca University's Mission Critical Research Fund.

6. Reference

[1] A. Haller, et al., "WSMX - a semantic service-oriented architecture," ICWS 2005., pp. 321-328 vol.1.

[2] M. P. Papazoglou, et al., "Service-Oriented Computing: State of the Art and Research Challenges," IEEE Computer, 11, 2007.

[3] P. Krogdahl, G. Luef, and C. Steindl, "Service-oriented agility: an initial analysis for the use of agile methods for SOA development," IEEE Int'l Conf. on Services Computing, 2005, pp. 93-100 vol.2.

[4] S. Ho Chang , et al., "A Variability Modeling Method for Adaptable Services in Service-Oriented Computing," In Proc. of the 11th Int'l Software Product Line Conf., 2007, pp. 261-268.

[5] K. Czarnecki, et al., "Staged configuration using feature models," In Proc. of Software Product-Line Conf. 2004, pp. 266-283.

[6] OWL, http://www.w3.org/TR/owl-features/

[7] I.Horrocks, et al., "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C, May 21, 2004. http://www.w3.org/Submission/SWRL

[8] H.H. Wang, et al., "Verifying feature models using OWL," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, 2007, pp. 117-129.

[9] H. Wada, J. et al, "A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture," IEEE International Conference on Services Computing, 2007, pp. 187-195.

[10] K.C. Kang, et al., "Feature-oriented domain analysis (FODA) feasibility study," Carnegie Mellon University, SEI 1990.

[11] K.C Kang, et al., "FORM: A feature-;oriented reuse method with domain-specific reference architectures," Annals of Software Eng., vol. 5, Jan. 1998, pp. 143-168.

[12] M.L. Griss, J. Favaro, and M. d'Alessandro, "Integrating feature modeling with the RSEB," Proc. of the 5th Int'l Conf. on Software Reuse, 1998, pp. 76-85.

[13] W.Zhang, K. M.Hensen, "Synergy Between Software Product Line and Intelligent Mobile Middleware," In Intelligent Pervasive Computing, 2007 pp. 515-520.

[14] M. Anastasopoulos, "Software Product Lines for Pervasive Computing," IESE-Report No. 044.04/E 2005.

[15] J.White, and D. C. Schmidt, "Model-Driven Product-Line Architectures for Mobile Devices," In Proc. of the 17th Ann. Conf. Int'l Fed. of Automatic Control,2008.

[16] D. Roman: Web Service Modeling Ontology. Applied Ontology, 2005, 1(1), 77–106.

[17] K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged configuration using feature models," Lecture notes in computer science, 2004, pp. 266-283.

[18] D.Batory, "Feature models, grammars, and propositional formulas," In Software Product Lines Conference, LNCS 3714, pages 7–20, 2005.

[19] T. Weis, et.al, "Customizable pervasive applications," Pervasive Computing and Communications, 2006.

[20] K.Czarnecki, C.Kim, K.Kalleberg,"Feature Models are Views on Ontologies," SPLC 2006, 41-51.

[21] P.Simons.,I. Niemelä, and T.Soininen, "Extending and Implementing the Stable Model Semantics," Artificial Intelligence, 138, 2002, pp. 181-234.

[22] J.Warmer and A. Kleppe, 1999 The Object Constraint Language: Precise Modeling with UML. Addison-Wesley Longman Publishing Co., Inc.

[23] Delivery Context Ontology, W3C Working Draft, 2009, http://www.w3.org/TR/dcontology/

[24] E.Bagheri, D.Gasevic, "Feature Model Configuration using Fuzzy Propositional," to be submitted to IEEE Trans. on SMC, Part C (2009).

[25] I. Montero, J. Pena, A. Ruiz-Cortez, "From feature models to business processes," In Proc. of the 2008 IEEE Int'l Conf. on Services Computing. pp. 605-608.

[26] J.Rusk, "Transforming between WSMO services and Feature Models with ATL transformation language," http://io.acad.athabascau.ca/~jeffr/webdoc.html

[27] J. Lee, D. Muthig, M. Naab, "An approach for developing service oriented product lines," In Proc of the 12th Int'l Software Product Lines Conf. 275-284. 2008.

Semantic Variability Modeling for Multi-staged Service Composition

Bardia Mohabbati[1], Nima Kaviani[2], Dragan Gašević[3]
[1]Simon Fraser University, [2]University of British Columbia, [3]Athabasca University, Canada
mohabbati@sfu.ca, nimak@ece.ubc.ca, dgaseavic@acm.org

Outline

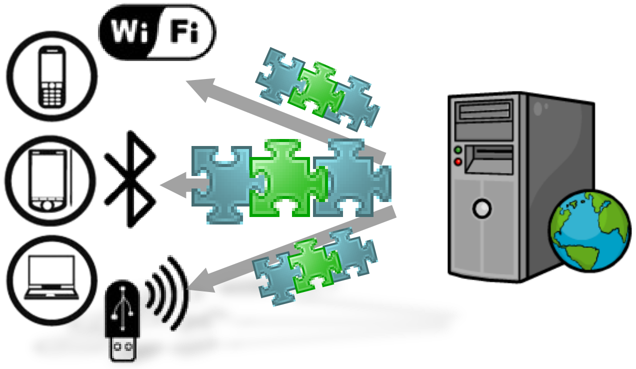
- Objectives and Introduction
- Motivating Scenario
- Overview
  - Analysis and Design
  - Feature Model and Variability Modeling
  - Service Requirements Specification
  - Software Service Products Configuration
  - Service Discovery and Specialization
- Semantic Variability Modeling
  - Feature Model Ontology
  - Feature Model Annotation
  - Feature Model Specialization
- Conclusion

Objectives

- *Adaptive development of software-intensive systems* for mobile and ubiquitous Computing

Pervasive/Ubiquitous computing (ubicomp) is about providing services and computing capabilities in heterogeneous environments

- Heterogeneous computing device
- Resource constraints of deployment platform

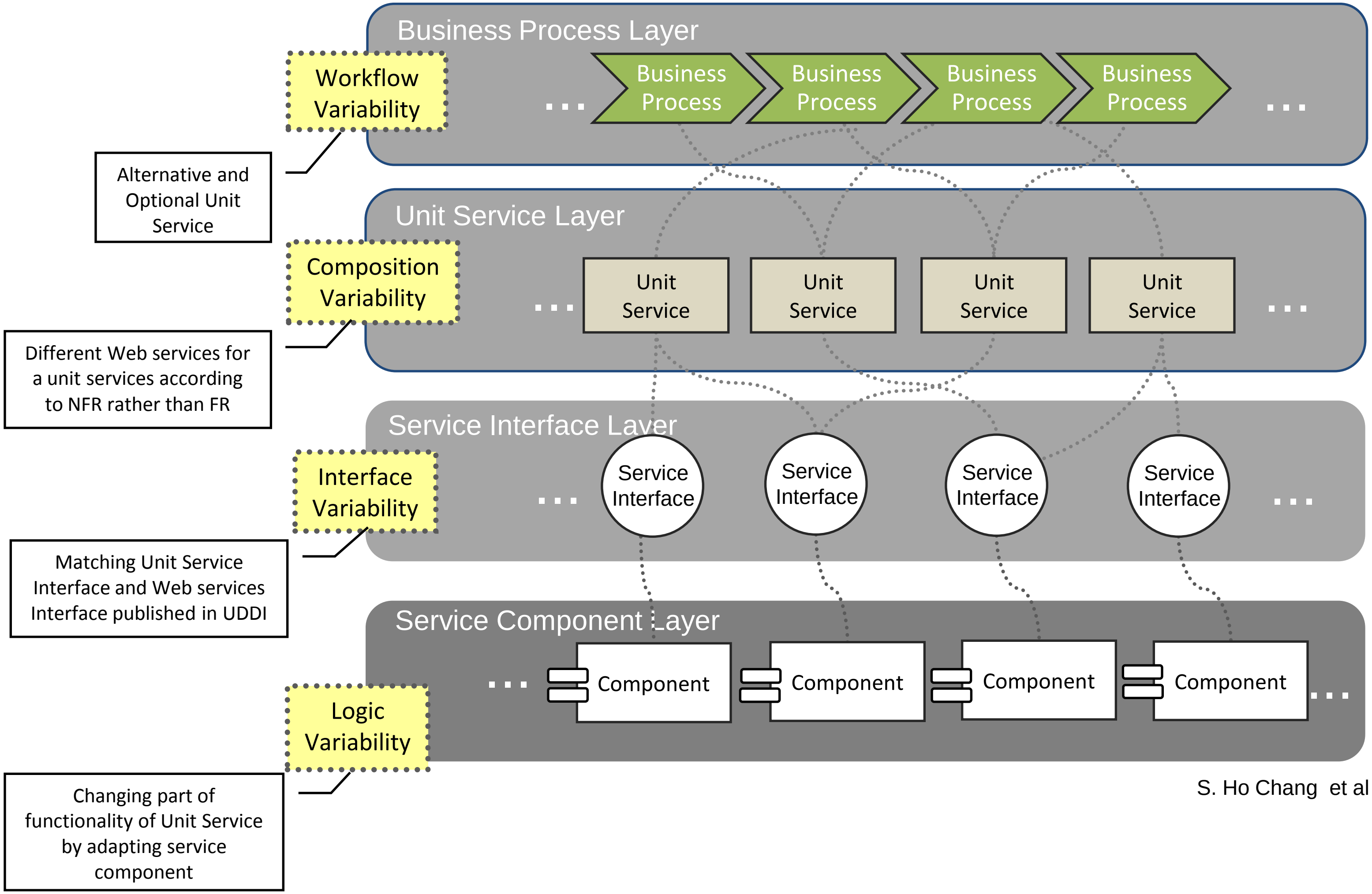


❖ Utmost functionality and satisfying non-functionality requirements (NFRs) in final product

❖ Service-Oriented Software Development and SPL

Objectives

We exclusively focus variability coming from the different delivery platform of service oriented architecture



Alternative and Optional Unit Service

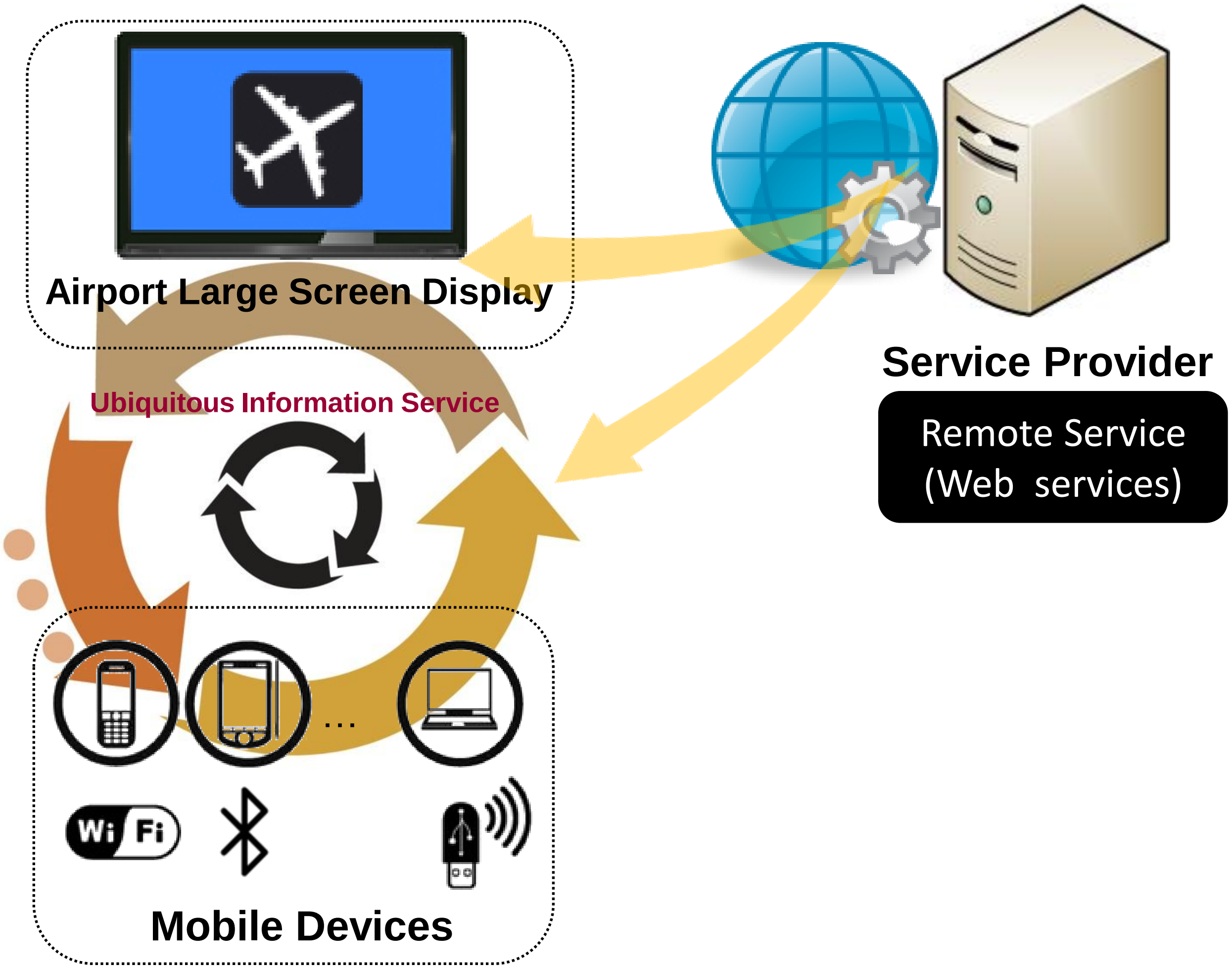
Different Web services for a unit services according to NFR rather than FR

Matching Unit Service Interface and Web services Interface published in UDDI

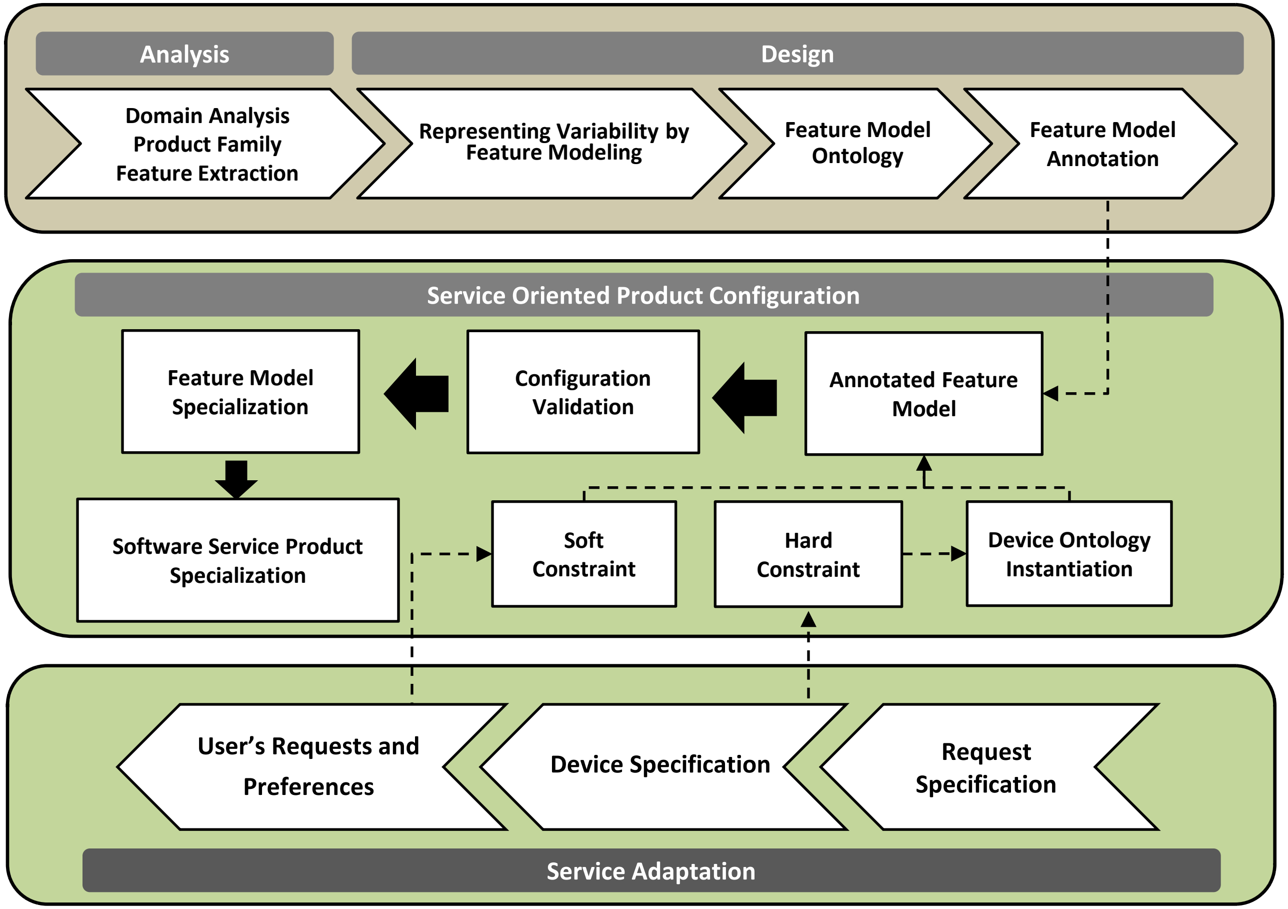
Changing part of functionality of Unit Service by adapting service component

S. Ho Chang et al

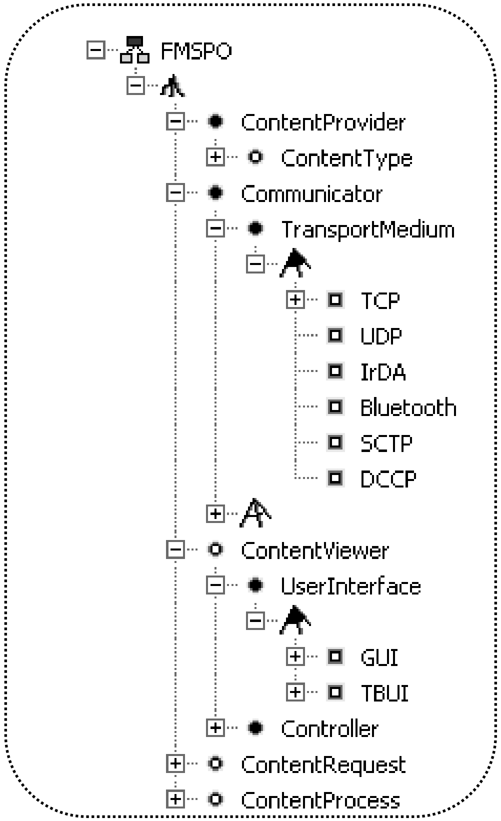
Motivating Scenario

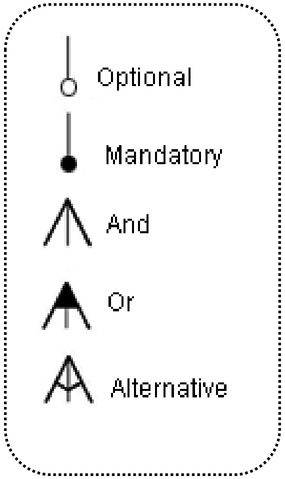


Overview


Analysis
Design
Domain Analysis Product Family Feature Extraction
Representing Variability by Feature Modeling
Feature Model Ontology
Feature Model Annotation
Service Oriented Product Configuration
Feature Model Specialization
Configuration Validation
Annotated Feature Model
Software Service Product Specialization
Soft Constraint
Hard Constraint
Device Ontology Instantiation
User's Requests and Preferences
Device Specification
Request Specification
Service Adaptation

Feature Model & Variability Modeling





Functional Requirements

- Core Services in the system
- Carry the objectives of the system

Non-functional Requirements

- Replaceable or alterable services
- Quality of Service (QoS) requirements
- Security requirements

Feature Model Ontology (FMO)
Semantic Feature Model

*Ontology is defined as a **formal specification of a conceptualization** and utilizes the representation of knowledge contained in feature models.*

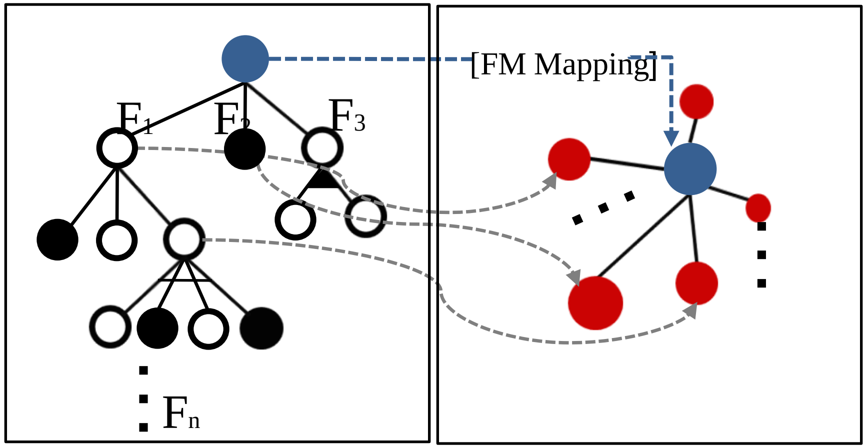
Semantic representation of features :

- Relations and dependencies
- feature constrains

Transformation :

- ○ Using the Web Ontology Language (OWL)
- ○ Constructing mutual disjoint classes

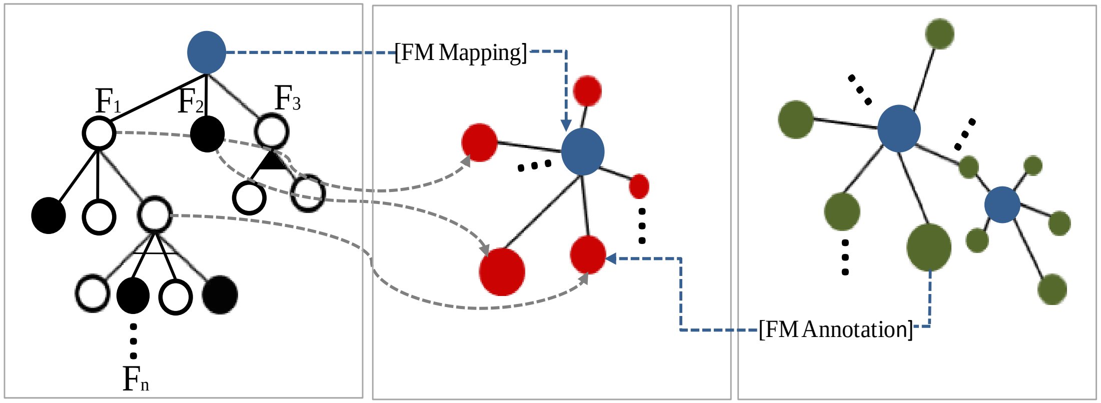
$$F_i \sqsubseteq \top$$
$$FM_iRule \sqsubseteq \top$$
$$hasF_i \sqsubseteq ObjectProperty$$
$$FM_iRule \equiv \exists\ hasF_i.\ F_i$$
$$F_i \sqsubseteq \neg F_j\ ,\ \ for\ i \leq 1, j \leq n \wedge i \neq j$$



Feature Model **Feature Model Ontology**

Feature Model Annotation

Feature Model Ontology *enables us to take advantages of ontology annotation capabilities in order to enrich the definition of domain assets with the constraints concerning the non-functional requirements of a domain*



Feature Model Feature Model Ontology (FMO) Device Ontology(DCO)

$$AN \sqsubseteq ObjectProperty$$
$$AN \sqsubseteq FM_m,\ domain(FM_m)$$
$$\top \sqsubseteq \forall\ AN.DC_n\ ,\ range\ (DC_n)$$

→

$$hasBluetooth\_AN \sqsubseteq ObjectProperty$$
$$hasBluetooth\_AN \sqsubseteq Bluetooth,$$
$$domain(BluetoothRule)$$
$$\top \sqsubseteq \forall\ has\ Bluetooth\_AN.BluetoothDevice,$$

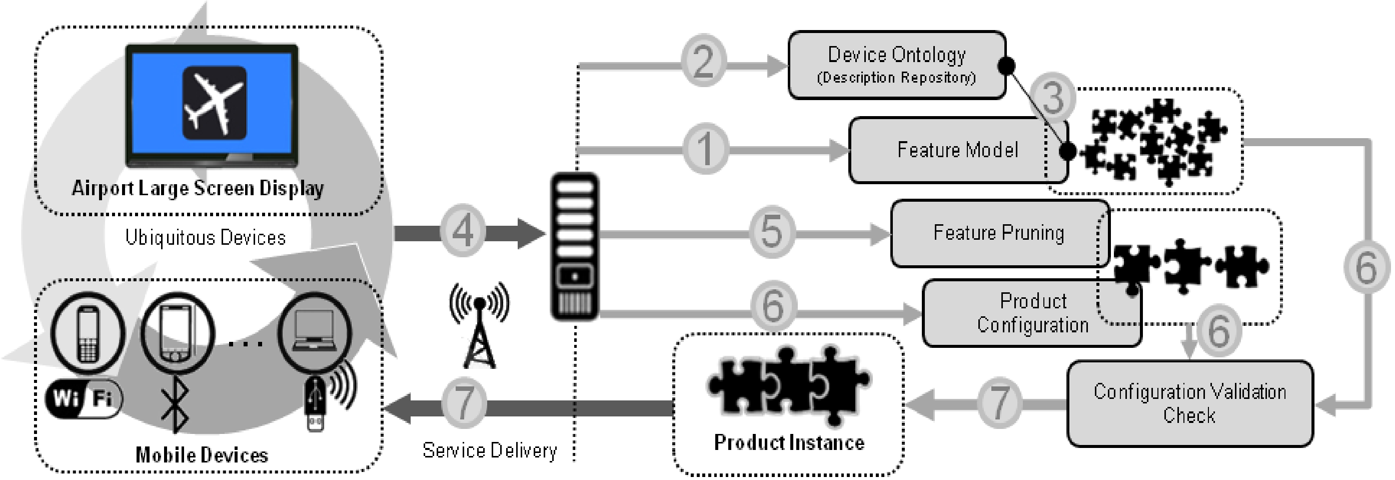
Software Service Products Configuration

- Device Ontology model reflect device capability (NFRs)
- NFRs as Integrity Constraints (IC)- which include and exclude the selection of services in annotated feature model

- ***Feature Model Specialization** : Stage refinement process which constitute staged configuration*
  - *Specialization:* The features don't satisfy the NFRs of target deployment platform are discarded and pruning feature model
  - *Ontology-based reasoning :* consistency checking and verification
    - If instances of NFRs ontology are consistent with respect to the annotation properties defined in feature model

Software Service Products Configuration

Design Time	Run Time
1. Creating the feature model	4. Platform Deployment Capability analysis
2. Incorporating NFRs into designing the feature model	5. Feature selection
3. Binding Services to the NFR ontology	6. Service composition
	7. Validation and Deployment



Software Service Products Configuration

Feature Model Specialization

$\mathcal{FMO}$: Feature Model Ontology
$\mathcal{DCO}$: Device Capability Ontology

$\mathcal{K} = (\mathcal{O}, \mathcal{T}, \mathcal{A}, \mathcal{R})$
$\mathcal{T}$: set of concept axioms (TBox)
$\mathcal{A}$: set of assertional axioms (Abox)
$\mathcal{R}$: set of rules written as inclusion axioms

DEFINITION 1 *Let $d \in \mathcal{DCO}$ be an instance of a device which has n capabilities. Each capability for device d is an instance (i_k) of a concept (C_k) from $\mathcal{DCO}$ such that $\mathcal{A} \vDash C_k(i_k)$ ($1 \leq k \leq n$). $S_{dc} = \{C_1,..,C_n\}$, represents the set of all concepts C_k from $\mathcal{DCO}$ that d supports.*

DEFINITION 2 *Let $S_{AF} \sqsubseteq \mathcal{FMO}$ be a set consisting of concepts CF_i such that $\forall CF_i.\top |$ $\exists AN_i \sqsubseteq CF_i \sqcap \exists AN_i.C_k$ where $C_k \in \mathcal{DCO}$.*

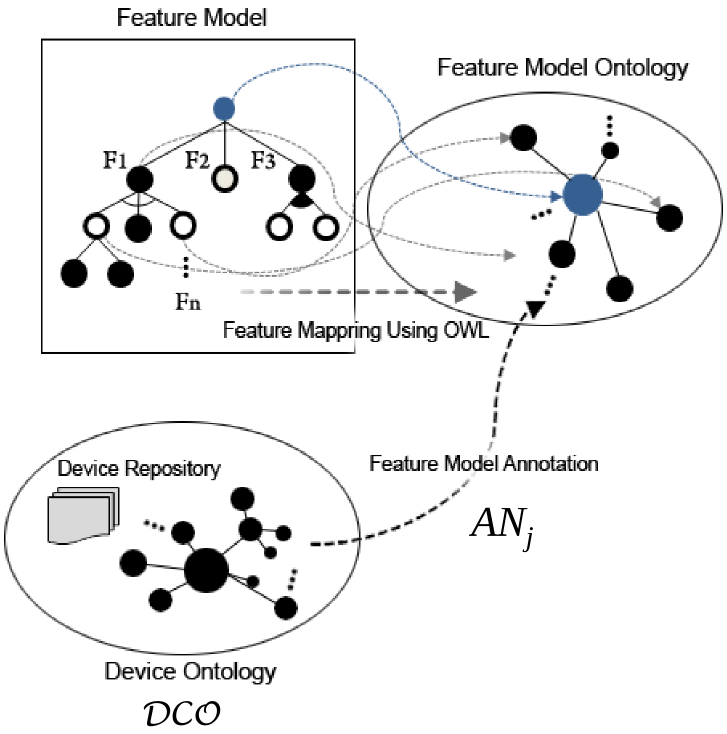
Software Service Products Configuration

Feature Model Specialization

$AN_j \sqsubseteq ObjectProperty$

$$SFA = \bigcup_{i=1}^{n} CF_i$$

```
for each $C_k \in S_{dc}$
    for each $CF_i \in S_{AF}$
        if ($\exists AN_j.C_j \sqcap AN_j \sqsubseteq CF_i$ and $C_j \in \mathcal{DCO}$)
            if ($C_j \in S_{dc}$)
                if ($\mathcal{A} \not\models CFi(i_j)$ )
                    $SFA \leftarrow SFA - \{CF_i(i_j)\}$
            else
                $SFA \leftarrow SFA - \{CF_i(i_j)\}$
    return ($SFA$)
```



Feature Model Specialization

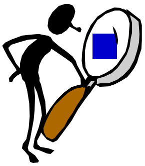
Model Verification

During the process of the configuration validation, model constraints are checked against the derived configuration to provide proper assurance in terms of correct exclusion and inclusion of optional and mandatory features.

- ***OWL-DL reasoning engines:*** *To support automated class subsumption, consistency reasoning and detection of possible inconsistencies in the specialized feature.*

- ***Consistency and conflicts detection*** *: Formulate and define a list of Semantic Web Rule Language (SWRL) rules to represent all invalid states*

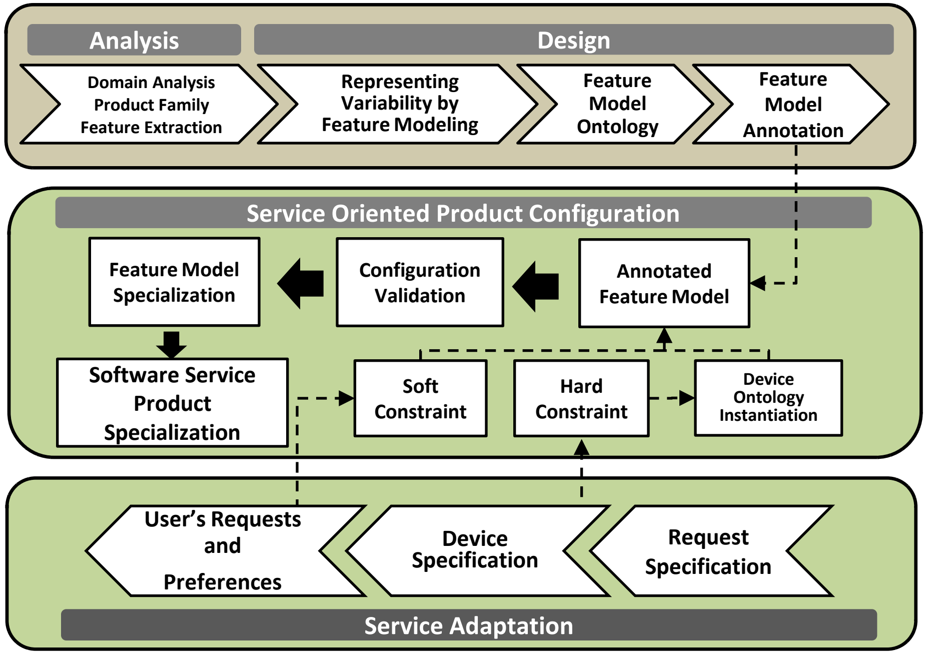
Service Discovery and Specialization



- The result of consistency checking in the previous step can be two folded :
  - Inconsistent ontology → Design Stage - family composition refinement
  - Consistent ontology → Service Specialization based on soft constraints

- Deployment of the service-oriented system:
  - Web Service Modeling Ontology framework (WSMO)
  - Transformation feature model to WSMO service (with ATL transformation language)
  - Web service Execution Environment (WSMX)

    (the reference implementation for WSMO)



Analysis

Design

Domain Analysis Product Family Feature Extraction

Representing Variability by Feature Modeling

Feature Model Ontology

Feature Model Annotation

Service Oriented Product Configuration

Feature Model Specialization

Configuration Validation

Annotated Feature Model

Software Service Product Specialization

Soft Constraint

Hard Constraint

Device Ontology Instantiation

User's Requests and Preferences

Device Specification

Request Specification

Service Adaptation

Conclusion

- *Ontologies as an underlying formalism for representation of feature models to enrich Feature Model with semantics*

- *Ontology-based approach for feature model annotation with NFRs ontologies*

- *Inference and Ontological Reasoning Over Feature Model*
  - ❑*validating (non-) functional requirements*
  - ❑ Semi-automatic Product configuration
  - ❑ Product consistency check

Thank you ☺

Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions

Dennis Smith and Grace Lewis
Software Engineering Institute, Carnegie Mellon University
Pittsburgh, PA, USA
{dbs,glewis}@sei.cmu.edu

Abstract

This paper examines the use of Service-Oriented Architecture (SOA) services as core assets in a Software product Line (SPL). After a brief introduction to the main concepts of SOA and SPL, the paper identifies a small set of decisions that are required before implementation of SOA-SPL systems. These decisions have to do with 1) the mapping of SOA concepts to the SPL framework, and 2) an initial set of potential variation mechanisms. The paper also identifies future work to more completely address SOA-SPL implementation planning.

1. Introduction

A Software Product Line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [1]. Successful products lines have enabled organizations to capitalize on systematic reuse to achieve business goals and desired software benefits such as productivity gains, decreased development costs, improved time to market, higher reliability, and competitive advantage [1, 2].

Service Oriented Architecture (SOA) is a way of designing, developing, deploying and managing systems, in which

- Services provide reusable business functionality.
- Applications or other service consumers are built using functionality from available services.
- Service interface definitions are first-class artifacts.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Protocols are predominantly, but not exclusively, message-based document exchanges [3].

The business case for both SPL and SOA emphasize efficiencies and cost savings through reuse. SPL focuses on the use of a common, managed set of core assets for rapidly producing multiple products or systems, according to a centrally managed production plan. An SOA implementation exposes standard interfaces to make services available for authorized service consumers to use in a variety of ways. These consumers of services are not necessarily anticipated by service providers, or controlled by a central authority.

A number of authors have suggested a relationship between SPL and SOA [4, 5, 6, 7, 8]. These works focus primarily on the use of services as core assets in SPL in which the services handle variability, and they address the relationship between SPL and SOA from a conceptual level but do not go into details about implementation. The goal of this paper is to identify an initial set of decisions that have to be made before going into implementation. These decisions are based on 1) the mapping of SOA concepts to the SPL framework and 2) an initial set of potential variation mechanisms.

The paper is organized as follows. Section 2 briefly outlines SPL concepts and identifies a set of specific SPL practice areas to be addressed. Section 3 outlines key SOA concepts that are relevant for using services at SPL core assets. Section 4 identifies a set of pre-implementation decisions. Section 4.1 outlines decisions that map to a selected set of SPL practice areas. Because variation points are central to a successful SPL implementation, Section 4.2 identifies an initial set of potential variation mechanisms for services. Finally, Section 5 provides a summary, conclusions and next steps.

2. Software Product Line Concepts

The SPL framework identifies three essential product line activities: core asset development, product development and management. The framework also defines a set of practice areas that are essential for carrying out these three essential product line activities. A practice area is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line. Practice areas help to make the essential activities more achievable by defining activities that are smaller and more tractable than a broad imperative such as "develop core assets" [2].

Because this paper addresses the use of services as core assets, the SPL activity of most immediate concern is that of core asset development. While a number of SPL practice areas are relevant, we identify a set of pre-implementation decisions that need to be made in the practice areas of:

- Architecture definition
- Using externally available software
- Mining existing assets
- Testing

After these decision points are identified, some potential mechanisms for using services as core assets are outlined.

3. Service-Oriented Architecture Concepts

This section identifies key SOA concepts that are relevant for using services as SPL core assets.

At a high level, as shown in Figure 1, there are three major components of service-oriented systems: services, service consumers and SOA infrastructure.

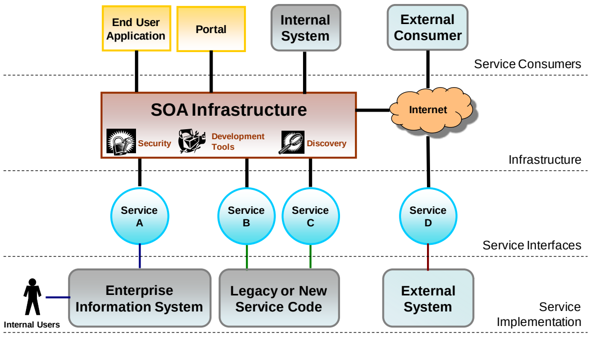


Figure 1. High-Level Representation of a Service-Oriented System

- Services are reusable components that represent business tasks, such as customer lookup, weather, account lookup, or credit card validation and that can be globally distributed across organizations and reconfigured to support new business processes. A distinguishing factor is that service interface definitions are standardized, well-defined, first-class artifacts, available in some form of service registry so that services can be discovered by service consumers.
- Service consumers are clients for the functionality provided by the services. Examples of service consumers are end-user applications, portals, internal and external systems, or other services in the context of composite services. A product in a product line could be a consumer of a service.
- SOA infrastructure connects service consumers to services. It usually implements a loosely coupled, message-based communication model. The infrastructure often contains elements to support service discovery, security, data transformation and other operations. A common SOA infrastructure is an Enterprise Service Bus (ESB) to support Web Service environments [9].

Service-oriented systems support three main types of operations: service discovery, service composition, and service invocation.

- Service discovery: Service providers place information about their services in a service registry and service consumers query this registry for services with desired characteristics.
- Service composition: Applications and other service consumers compose functionality provided by services to fulfill their goals. Languages such as the Business Process Execution Language (BPEL) support the orchestration of services in a Web Services environment [10].
- Service invocation: There are two common patterns for service invocation
  - ○ a simple invocation pattern where service consumers directly invoke services over a network, typically via synchronous, direct, request-reply connections
  - ○ a richer invocation pattern where service consumers invoke services via a middleware component that supports SOA environments, such as an Enterprise Service Bus (ESB) [9]

These basic SOA components and activities are addressed in the decision points that follow.

4. Pre-Implementation Decisions for SOA-SPL Systems

As mentioned earlier, this paper focuses on pre-implementation decisions for systems that use as SPL core assets available as services. Subsection 4.1 outlines decisions in four SPL practice areas: Architecture Definition, Using Externally Available Software, Mining Existing Assets, and Testing. Subsection 4.2 identifies decisions to make on potential variation mechanisms.

4.1 SPL Practice Area Decisions

Architecture Definition

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [11]. If core assets are to be made available as services, decisions include:

- Decisions on the specific sets of SOA standards, interfaces and technologies for implementation and how will these will interoperate with the rest of the product line architecture.
    - The first choice concerns the standards to use. The most common SOA standards are web services. Alternative SOA standards may be used, such as REST. Decisions on standards require an analysis of both Quality of Service (QoS) needs and functionality. Web service standards have greater support for QoS needs such as security, availability, and performance. REST is more flexible and easier to compose, but has less support for most of the standard QoS needs. It is more appropriate for read-only functionality, typical of mashups, where there are minimal QoS requirements and concerns.
    - Once standards are chosen, the next decisions focus on specific standards and tools required for specific SPL QoS needs. In the case of web services, a large number of choices are available—approximately 250 WS-* web service standards. Each standard in turn may have a number of different versions and tool support. Many of these decisions will have architectural implications.
- Decisions on how discovery, composition and invocation are to be accomplished. A number of options can be considered, such as building a SOA infrastructure that supports these operations, buying an ESB product or embedding these operations within the context of a broader product line architecture. Each of these basic operations requires a set of decisions.
    - Discovery. In most literature that discusses the relationship between SOA and SPL, there is an expressed need for discovery to take place at runtime. However, the current state of the practice only supports design-time discovery. Decisions need to be made on the specific mechanisms for discovery and for interacting with a service registry. If some type of runtime discovery is required, there may need to be some form of user intervention to choose an appropriate service or the non-trivial implementation of a service broker. Decisions in the former case include how to handle user intervention and in the latter case how the broker is to be implemented and used by SOA-SPL developers.
    - Composition. Policies for composing services and allowed usage of services need to be established. These policies need to be made explicit and enforcement mechanisms need to be established. Decisions include specific mechanisms for performing composition, such as WS-BPEL, and where to implement composition, such as in the infrastructure.
    - Invocation. In most cases the infrastructure will support the invocation of services. In this case decisions need to be made on the type of functionality to be handled by the middleware (e.g., routing, mediation, process orchestration, complex event processing). On the other hand, if services are to be invoked directly by the service consumer, decisions need to be made on how this will be accomplished.
- Services may be accessed via an Intranet or the Internet. Decisions need to be made on the scope of access, firewalls, permissions, and control of services. If the Internet is to be used, performance and availability metrics need to be identified, measured and tracked; sources of bottlenecks need to be identified; and satisfaction of service level agreements (SLAs) needs to be monitored.

Using Externally Available Software

There is a growing market of externally developed services that can be purchased or licensed. In addition, ERP vendors are making significant investments to add service interfaces to their existing ERP solutions to be used by custom applications. Common business services, such as check credit, customer lookup and check inventory are strong candidates for core assets if they are relevant for the product line.

Decisions required for the use of externally available services include:

- How are the services to be accessed, what standards do they support, what outputs do they return, and in what form?
- Do the external services meet the functionality and quality of service requirements of the SPL?
- What type of testing has been performed on the services, at what level, and what are the results?

- How appropriate and effective is the SLA attached to the service?
- What types of mechanisms are built into the services to handle variations?
- Do the external services have an option to establish variability points?
- Can variability be handled by the infrastructure or by consumers of services?

Mining Existing Assets

If services are to be mined from existing assets, an important pre-implementation decision concerns the viability of exposing services from existing assets. This decision requires per-system answers to a set of questions, including:

- Does it make sense to migrate the legacy system to an SOA environment?
- What services make sense to develop?
- What legacy system components can be used to implement these services?
- What changes to components are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?
- What is an ideal pilot project that can help address some of these risks?

The Service Reuse and Migration Technique (SMART) [12] provides one systematic method for answering these questions and making decisions. SMART addresses both general migration issues as well as those that are relevant to a specific situation. In the case of using services as part of a product line, specific points to be addressed include variation points, relationship to other core assets, composition strategies and SLAs.

Testing

Issues with testing SPL core assets implemented as services need to be addressed from both the service provider and service consumer perspectives.

From a service provider perspective, systems that expose functionality as services usually have "day jobs”. This means that the system operates in a "business as usual" manner and also provides service interfaces so that other systems (internal and/or external to the organization) have access to a subset of functionality that exists in the system. This requires decisions for how to address testing challenges.

- Regression tests cover both conventional interfaces as well as service interfaces to make sure that changes made for one set of users do not affect the other set of users
- Functional tests consider potentially unknown users and uses of the functionality provided by the service. Functional testing needs to cover both current as well as potential usage scenarios.
- Exposing system functionality as services creates the potential for having a greater number of consumers of system functionality. This requires additional security testing, stress testing and load testing.
- Regression testing needs to verify whether existing service-level agreements (SLAs) are affected.
- Some service providers have test instances of their services to allow service consumers to perform end-to-end testing. As a result service providers have to maintain separate instances of their service interfaces as well as their service implementation so that test data does not affect production data. In addition they require extensive logging to use failure data for internal testing and improvement.

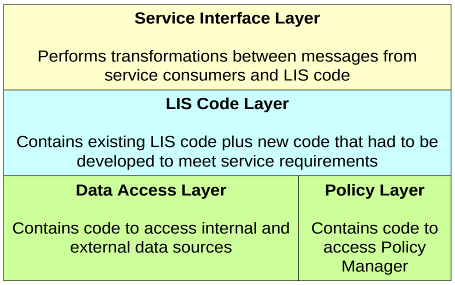
Service consumers need to make a greater set of decisions if the core assets use externally available software. For externally developed software, an SLA protects both the service consumer and provider in case of failure, but it does not prevent or eliminate failure. As a result service consumers need to develop and test their systems to consider the case when services are completely unavailable. External services also mean that there is no control over changes made to the service, release cycles, or even shutdown. Service consumers will have to be tested every time there is a new service release.

4.2 Variation Mechanism Decisions

Cohen and Tarr both present simple examples of models for product lines that are composed of services for medical records and insurance claims respectively [4, 7]. In the medical example, variations can occur depending on such factors as the medical actor who is involved (physician, nurse, technician), the type of medical practice (cardiology, radiology, endocrinology) and the type of health care organization (hospital, insurance company, physicians office). Core assets may include services for medical treatment, billing, and patient information. In the insurance claims example, variations can occur on such items as type of policy (life, home, auto), and state-specific policies.

Management of variability points is a key to product line success. However, variation mechanisms for SPL core assets implemented as SOA services have not been systematically addressed. Because decisions on specific variability mechanisms are important, we identify an initial set of decisions:

1. Parameters for invoking services. This is a simple variation mechanism in which parameters are used to invoke different variations of a service, such as different treatment responsibilities that may depend on role (physician, nurse, technician). This has been identified by in the literature as a primary variability mechanism [4,7].
2. Using infrastructure services to hide variability. A number of services can be common tasks that are delegated to the infrastructure. Examples include role-based identity management and data formatting. In the health domain, different sets of actors will require very different access and authorization privileges. Health care insurers will have the right to see financial information, physicians can see detailed treatment information, and research organizations will only be able to see information that is completely anonymous. These types of variations can be effectively handled as infrastructure services that are invoked when needed.
3. Encapsulating variability within a service. This approach isolates core service functionality from aspects that are either highly changeable, or in the case of an SPL, potential variation points. Figure 2 shows an example in which separate service layers are created for the interface, core service code, data access and in this case, access to a policy manager infrastructure service.



Service Interface Layer

Performs transformations between messages from service consumers and LIS code

LIS Code Layer

Contains existing LIS code plus new code that had to be developed to meet service requirements

Data Access Layer

Contains code to access internal and external data sources

Policy Layer

Contains code to access Policy Manager

4. Differential composition of atomic services. Services are often developed as atomic services that perform specific tasks. In situations where different configurations are required (such as SPL), or where external policies in the business environment require frequent unplanned changes (such as health care), building in a capability for composing services from a number of atomic services enables variability. The composition of services can be delegated to the infrastructure through a standard such as WS-BPEL (Web Services Business process Execution Language) or through proprietary or custom developed Business Process Management (BPM) functionality. This enables applications or products in a product line to be developed through the integration of functionality from existing services.
5. Using different protocols for interface implementations. In service-oriented applications, there may be a need for different interface implementations where the same business functionality is available through different interfaces. For example internal consumers may be able to use an internal EJB interface, and external consumers will use a web service interface to the same functionality.

5. Conclusions and Next Steps

The implementation of an SPL using core assets implemented as services has significant potential. However, to gain the full potential requires making a set of pre-implementation decision points and engineering tradeoffs. The Framework for Product Line Practice offers a good starting point. It identifies 29 key product line practice areas. We have focused on four practice areas that have strong relevance for SOA services and have identified an initial set of decisions in these areas. We have also identified a set of potential variability mechanisms that have relevance for SPL core assets implemented as services.

In addition, proof-of-concept analyses of the relevance of specific technologies, tools and methods to the context for which they were developed can also be instrumental in building up a body of knowledge in the area [13]. For example, Sidharth Surana from the Carnegie Mellon University Master of Software Engineering program is currently conducting a proof-of-concept analysis of the relevance of different variation mechanisms for a simple product line example.

Future directions will require a validation and updating of the initial mappings, more complete mapping of services to SPL product line practices, and empirical research on actual SOA-based SPL implementations. This can ultimately lead to a codification of best practice for the use of SOA in the context of SPL.

6. References

[1] Clements, P. & Northrop, L. M. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[2] Northrop, L. & Clements, P. A Framework for Software Product Line Practice, Version 5.0 http://www.sei.cmu.edu/productlines/framework.html (2009).

[3] Lewis, Grace. *Service-Oriented Architecture (SOA).* SEI Webinar Series www.sei.cmu.edu/collaborating/spins/081408webinars.html

[4] Cohen, Sholom & Krut, Robert. Proceedings of the First Workshop on Service-Oriented Architectures and Product Lines (CMU/SEI-2008-SR-006). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[5] Cohen, Sholom & Krut, Robert. Managing Variation in Services from a Software Product Line Context. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2009 - forthcoming.

[6] Ralph Mietzner, Andreas Metzger, Frank Leymann and Klaus Pohl. Variability Modeling to Support Customization and Deployment of Multi-Tenant-Aware Software as a Service Applications. In Proceedings of PESOS Workshop, ICSE, Vancouver, Canada, May 18-19, 2009.

[7] Tarr. P. *Technologies for Software Product Line Development*. https://www-950.ibm.com/events/wwe/grp/grp004.nsf/vLookupPDFs/tarr-product-lines-033009-slides/$file/tarr-product-lines-033009-slides.pdf

[8] S. G¨unther and T. Berger, "Service-oriented product lines: Towards a development process and feature management model for web services," in *SPLC '08: 12th International Software Product Line Conference*, pp. 131–136, 2008.

[9] D. Chappell, *Enterprise Service Bus*, O'Reilly, June 2004.

[10] Organization for the Advancement of Structured formation Standards, "Web Services Business Process Execution Language Version 2.0", 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html/

[11] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice*, 2nd ed. Boston, MA: Addison-Wesley, 2003.

[12] Lewis , Grace A.; Morris, Edwin J.; Smith, Dennis B.; Simanta, Soumya. SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment (CMU/SEI-2008-TN-008). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[13] F. Hueppi, L.Wrage, and G. Lewis, "T-Check in Technologies for Interoperability: Business Process Management in a Web Services Context", CMU/SEI-2008-TN-005, *Software Engineering Institute, Carnegie Mellon University,* Pittsburgh, PA, June 2008.

Service-Oriented Architecture (SOA) and Software Product Lines: Pre-Implementation Decisions

Dennis B. Smith
dbs@sei.cmu.edu

Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL)
13th International Software Product Line Conference (SPLC)
San Francisco, Ca
August 25, 2009

Software Engineering Institute | Carnegie Mellon

Agenda

Intersection of software product lines and SOA

Selected practice areas and decision points

Variation mechanisms

Conclusions and next steps

Software Product Lines

Software product line

- a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way

Successful products lines enable organizations to capitalize on systematic reuse to achieve business goals

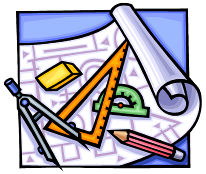
Benefits of software product lines

- productivity gains
- decreased development costs
- improved time to market
- higher reliability
- competitive advantage

Service Oriented Architecture

Service-oriented architecture (SOA) is a way of designing, developing, deploying and managing systems, in which

- Services provide reusable business functionality via well-defined interfaces.
- Service consumers are built using functionality from available services.
- Service interface definitions are first-class artifacts.
  - There is a clear separation between service interface and service implementation
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Protocols are predominantly, but not exclusively, message-based document exchanges.



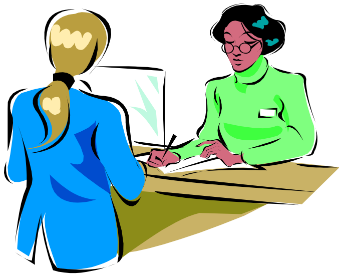
Services

Services are reusable components that represent business tasks, e.g.

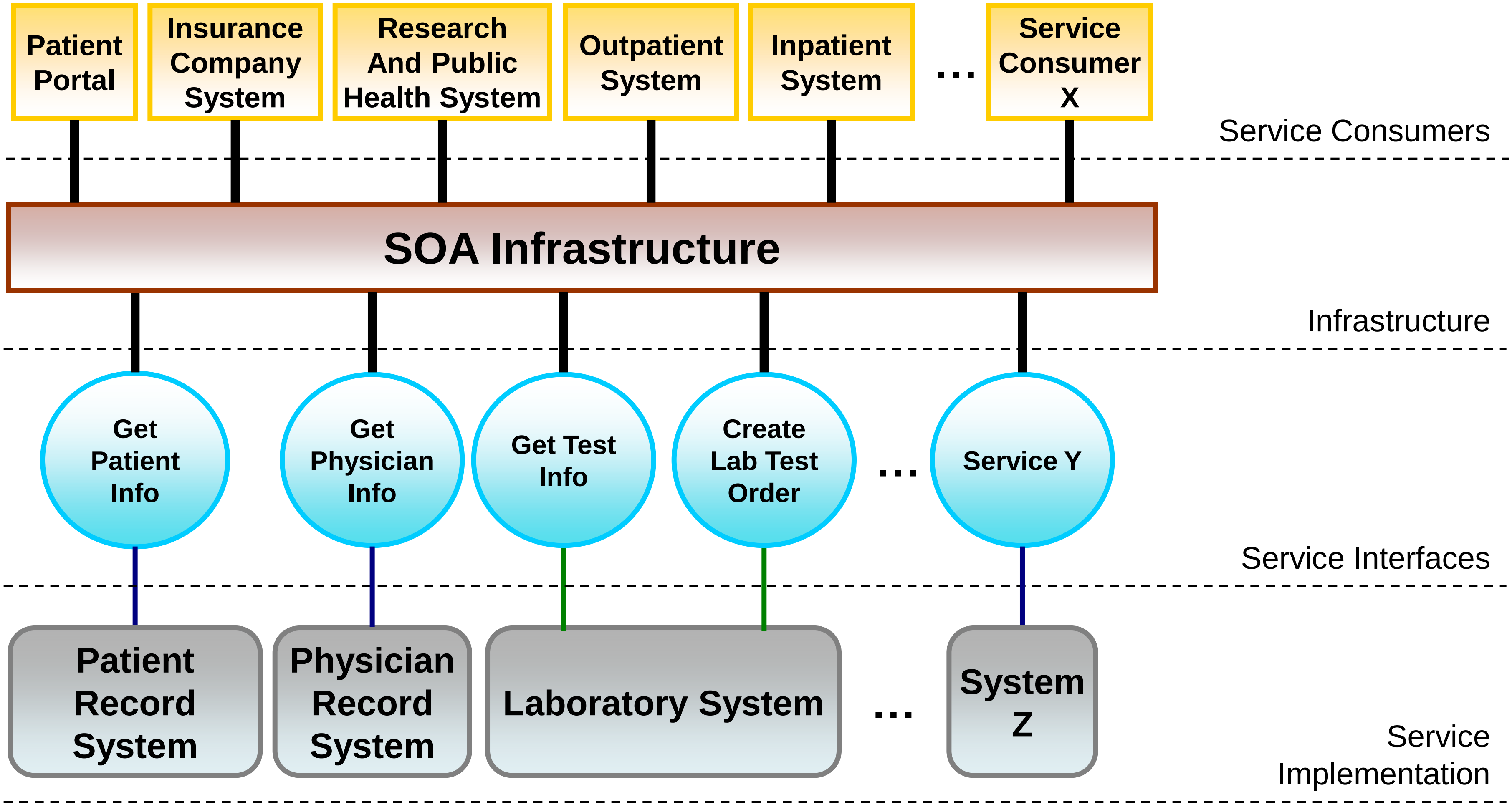
- Look up patient information
- Validate credit card
- Get test results
- Schedule appointment

Services can be

- Globally distributed across organizations
- Reconfigured into new business processes



A Notional Service-Oriented System Architecture



Intersection Between Software Product Lines and SOA

Software product lines support systematic reuse by using core assets with a production plan to enable the rapid generation of new products

SOA exposes services that can be reused by a variety of consumers, enabling:

- Agility, adaptability, cost efficiency and legacy leverage

SOA services can become core assets within a product line

Software product line framework identifies 29 required practice areas

- We initially identify decision points that need to be addressed in 4 practice areas if services are to be used as core assets

SOA services can provide significant leverage as variability mechanisms in a product line

- We initially identify 5 variability mechanisms

Agenda

Intersection of software product lines and SOA

Selected practice areas and decision points ⇦

Variation mechanisms

Conclusions and next steps

Selected Practice Areas

Architecture

Using externally available software

Mining existing assets

Testing

Practice Area: Architecture

Specific SOA standards, interfaces and technologies

- Type of standards: web services, REST, proprietary standards
- Specific implementations: eg web services has 250 different standards

Handling of basic SOA operations

- Decisions on responsibility for operations (infrastructure, services, consumer)
  - Discovery
    - Design time
    - Use of broker; how broker is to be implemented
  - Composition and enforcement mechanisms
  - Invocation: routing, mediation, process orchestration, complex event processing

Practice Area: Using Externally Available Software

Decision Points

- How are the services to be accessed, what standards do they support, what outputs do they return
- Do the external services meet the functionality and quality of service requirements of the software product line
- What type of testing has been performed on the services, at what level
- How appropriate and effective is the service level agreement
- What types of mechanisms are built into the services to handle variations
- Do the external services have an option to establish variability points
- Can variability be handled by the infrastructure or by consumers of services

Practice Area: Mining Existing Assets

Does it make sense to migrate the legacy system to an SOA environment?

What services make sense to develop?

What legacy system components can be used to implement these service?

What changes to components are needed to accomplish the migration?

What migration strategies are most appropriate?

What are the preliminary estimates of cost and risk?

What is an ideal pilot project that can help address some of these risks?

Practice Area: Testing- 1

Service provider perspective

- Regression tests cover both conventional interfaces as well as service interfaces to make sure that changes made for one set of users do not affect the other set of users
- Functional tests consider potentially unknown users and uses of the functionality of the service.
- Greater number of consumers requires additional security testing, stress testing and load testing.
- Regression testing needs to verify whether existing service-level agreements (SLAs) are affected.
- Service providers have to maintain separate instances of their service interfaces as well as their service implementation so that test data does not affect production data.

Practice Area: Testing-2

Service consumer perspective

- service consumers need to develop and test their systems to consider the case when services are completely unavailable.
- external services
- no control over
  - changes made to the service,
  - release cycles,
  - shutdown.

Service consumers will have to be tested every time there is a new service release

Agenda

Intersection of software product lines and SOA

Selected practice areas and decision points

Variation mechanisms

Conclusions and next steps

Parameters Invoke Service

Parameters invoke different variations of a service, such as treatment responsibilities that may depend on role (physician, nurse, technician)

- a single service is invoked
- the parameters sent to the service are modified for the appropriate behavior

In health care domain, the service “OrderTest” can have variability to enable carrying out different laboratory tests.

- Variability mechanisms can allow for differential input and output
  - Input: list of test names/codes along with the notification rules
  - Output: list of orderIDs corresponding to the specific test ordered

Infrastructure Services Hide Variability

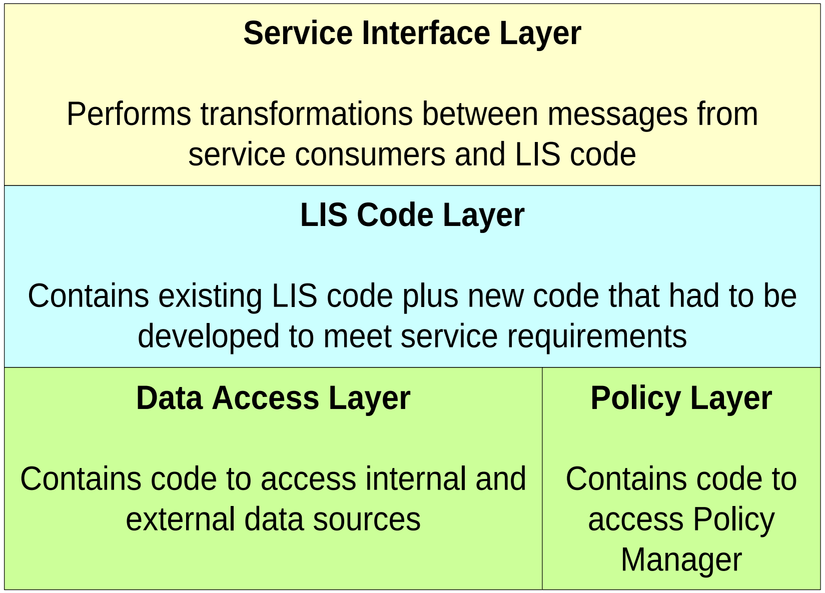
Common tasks become services that are delegated to the infrastructure.

- Examples include role based identity management and data formatting.
  - In the health domain, different sets of actors will require very different access and authorization privileges.
    - Health care insurers can access financial information
    - physicians can access detailed treatment information
    - research organizations will only be able to see information that is completely anonymous.
  - -the infrastructure can determine authorization privileges for different roles

Encapsulating Variability Within a Service

This approach isolates core service functionality from aspects that are either highly changeable, or in the case of an SPL, potential variation points.

Service layers can be created for the interface, core service code, data access and in this case, access to a policy manager infrastructure service



Service Interface Layer

Performs transformations between messages from service consumers and LIS code

LIS Code Layer

Contains existing LIS code plus new code that had to be developed to meet service requirements

Data Access Layer

Contains code to access internal and external data sources

Policy Layer

Contains code to access Policy Manager

Differential Composition of Atomic Services

Services are often developed as atomic services that perform specific tasks. In software product lines, different configurations are required.

Composing services from a number of atomic services enables variability

- composition of web services can be delegated to the infrastructure through
  - a standard such as WS-BPEL (Web Services Business process Execution Language)
  - proprietary or custom developed Business Process Management (BPM) functionality

Differential composition enables applications or products in a product line to be developed through the integration of functionality from existing services

Different Protocols for Interface Implementations

The same business functionality can be implemented through different interfaces.

Example: ordering laboratory tests

- internal consumers can use an internal EJB implementation
- external consumers requiring the internet may need to use more standard web service implementations with greater firewall protection.

Agenda

Intersection of software product lines and SOA

Selected practice areas and decision points

Variation mechanisms

Conclusions and next steps ⇦

Conclusions and Next Steps

Examine rest of software product line practice areas for decision points and engineering tradeoffs

Expand potential set of variation mechanisms

Perform proof of concept analyses of the relevance of specific technologies, tools and methods to the context for which they were developed can also be instrumental in building up a body of knowledge in the area

- Sidharth Surana of Carnegie Mellon has completed a proof of concept analysis of the relevance of different variation mechanisms for a simple product line example

Validation and updating of the initial mappings, more complete mapping of services to SPL product line practices, and empirical research on actual SOA based SPL implementations.