USING AN ASPECT ORIENTED LAYER IN SOA FOR ENTERPRISE APPLICATION INTEGRATION

Chinthaka D. Induruwana

Supervisor John Gurd

School of Computer Science

The University of Manchester

Service Oriented Architecture (SOA)

- SOA is a method for achieving EAI.
- SOA enables loosely coupled services to be individually deployed.
  - This means that a particular system can be constructed by joining smaller, loosely coupled components. The benefit of this is that smaller components are easier to develop, test and reuse.

Aspect Oriented Software Development (AOSD)

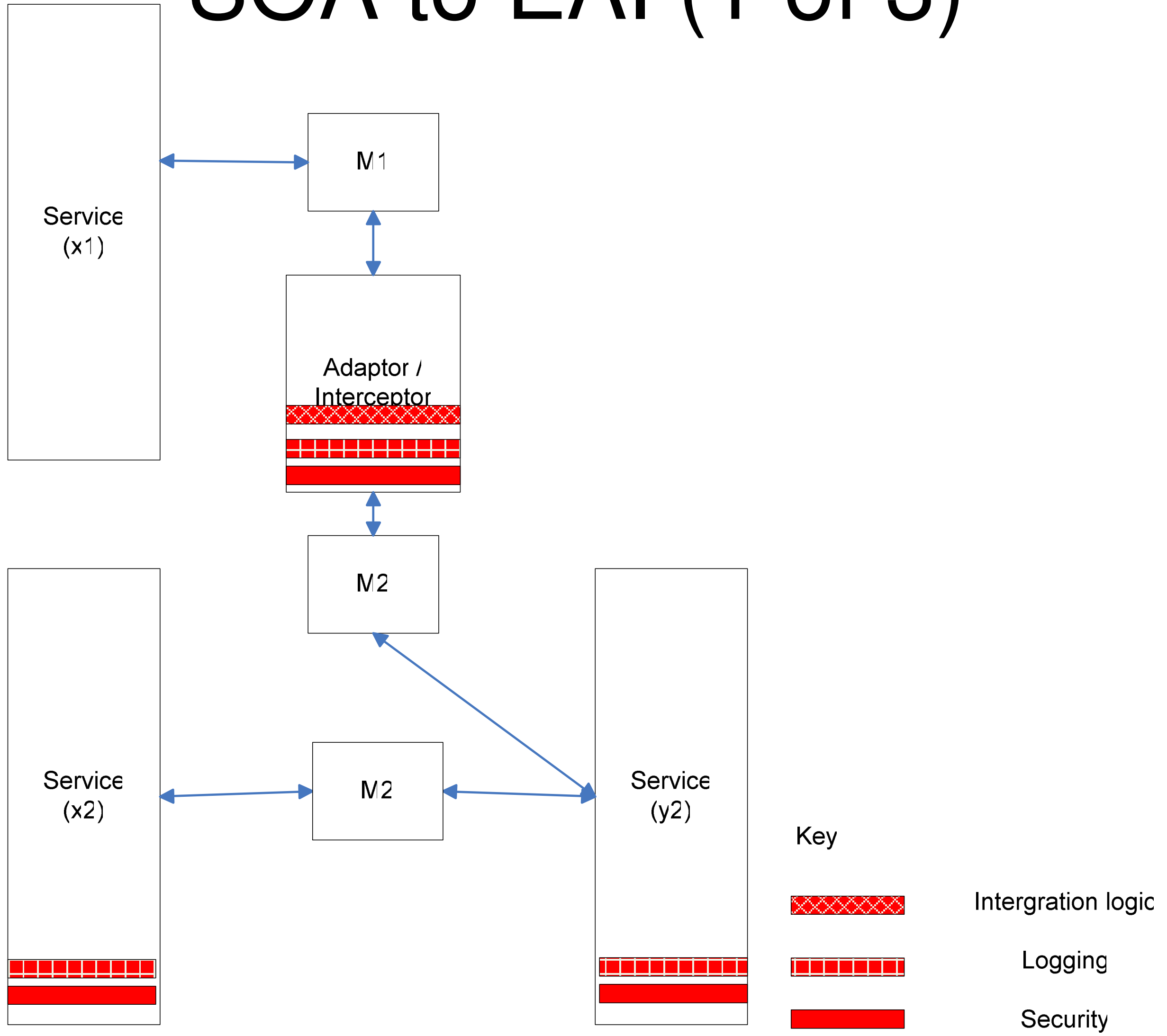
- Complements object oriented and procedural programming languages
- Aims to improve modularity
- Separates concerns
  - Functional
  - Non-functional
    - Non-functional concerns that crosscut are referred to as crosscutting concerns in AOP terminology and aspect can be used to encapsulate them

(Gregor Kiczales et al, 1997)

Non-functional concerns / Aspects

- **Logging**
  - The logging concern deals with the encapsulation of the logging behaviour. When certain points of the program execution are reached, the system log is updated to store a record of the program execution.
- **Security**
  - The security concern deals with different security aspects of the EAI. For example, the security mechanism used to communicate between the different services.

Current flexibility issues when applying SOA to EAI (1 of 3)



Current flexibility issues when applying SOA to EAI (2 of 3)

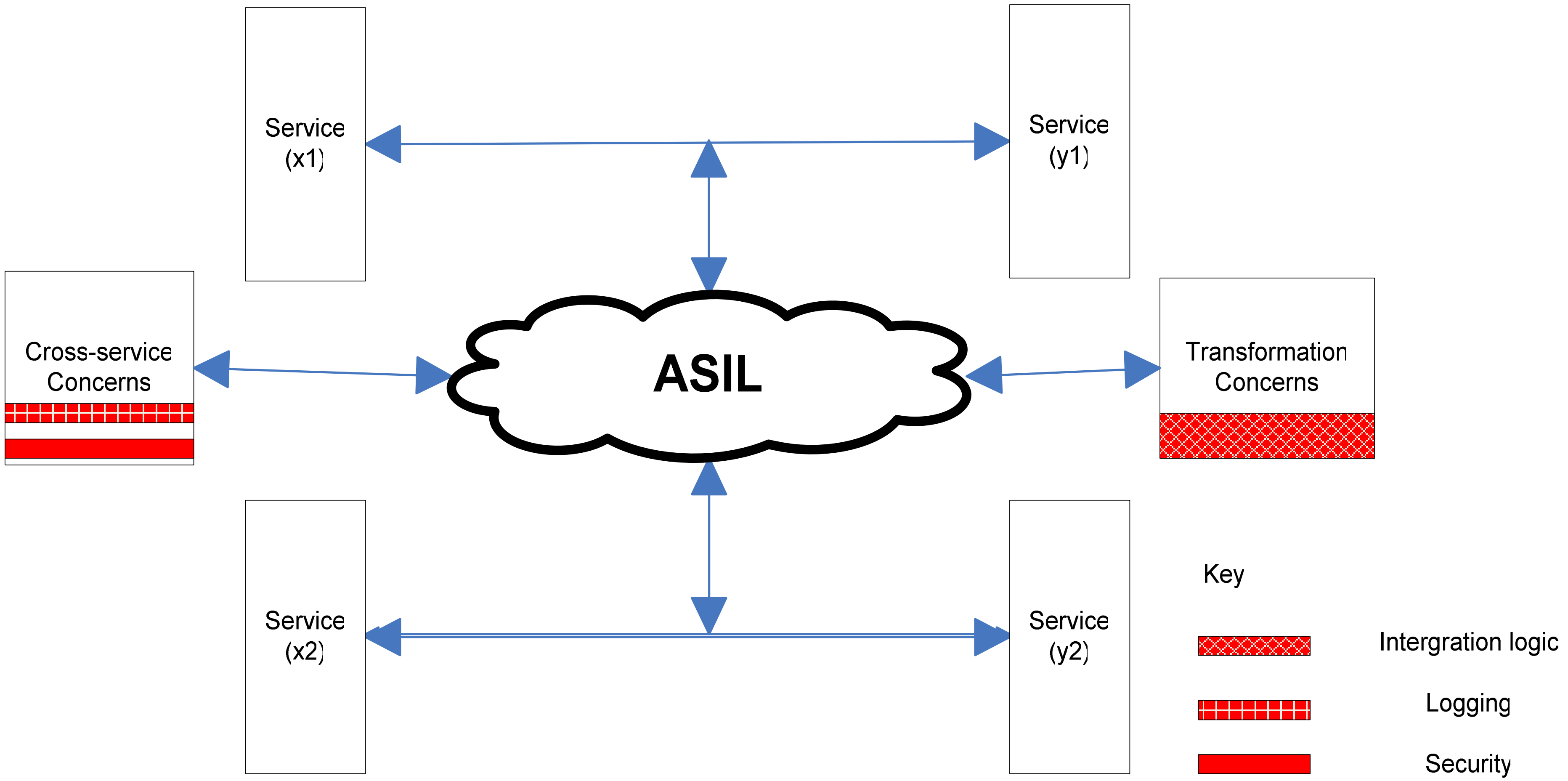
- Crosscutting
  - The implementation of a concern such as logging can result in the code for the implementation being scattered across the system.
- Code tangling
  - The implementation of a crosscutting concerns with non-AOP results in code tangling. Such that the code for a particular concern becomes intermixed with code for another concern.

Current flexibility issues when applying SOA to EAI (3 of 3)

- Ad-hoc solution (integration logic is hardwired to services)

- The current integration lacks flexibility, scalability and reliability

A novel layer in enterprise application integration (1 of 3)



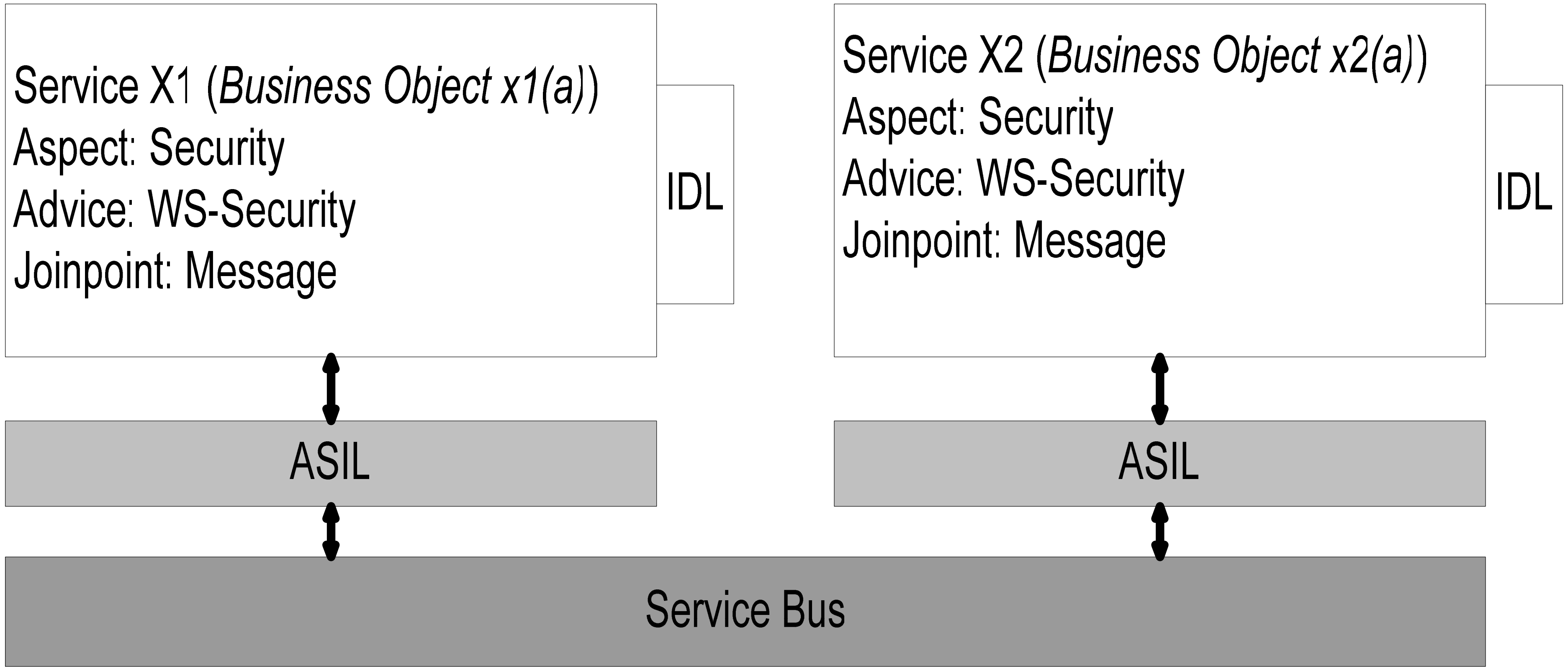
Application service integration layer (ASIL).

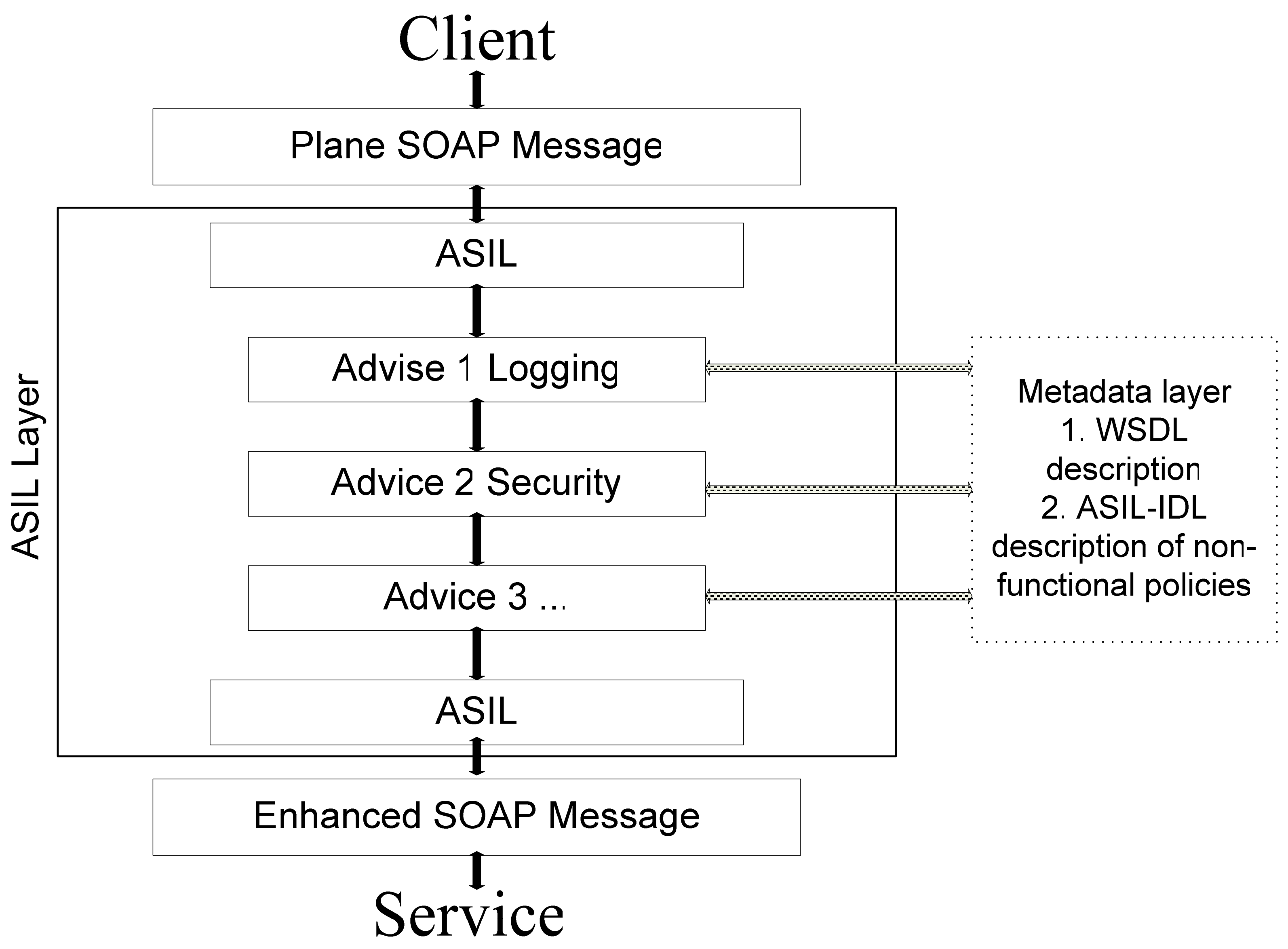
(Induruwana, 2005)

A novel layer in enterprise application integration (2 of 3)

- Join-points
  - Well defined places in the program execution flow where aspects can be advised.
- Advice
  - Advice is the behaviour of the aspect at the join-point.
- Weaving
  - Weaving is the process of composing a core functionality model with aspects and creating the final working system.

A novel layer in enterprise application integration (3 of 3)




Client
Plane SOAP Message
ASIL Layer
ASIL
Advise 1 Logging
Advice 2 Security
Advice 3 ...
ASIL
Enhanced SOAP Message
Service
Metadata layer
1. WSDL description
2. ASIL-IDL description of non-functional policies

IT Advantages

- Encapsulation of integration logic.
  - The advantage is that if a particular set of services is upgraded, or another one is integrated, and the XML message specification changes, only the transformation rule at one site needs to be altered.
- Encapsulation of cross-service concerns.
  - The advantage is that it enables the non-functional concerns of the system to be handled in a uniform and consistent fashion across the enterprise. Examples are logging and fault tolerance.

Business motivations for ASIL

- Enterprise wide standardisations
- Increased organisation agility
- Decreased future business automation costs
  - By decreasing the IT response time to adapt to business process change

Novel features

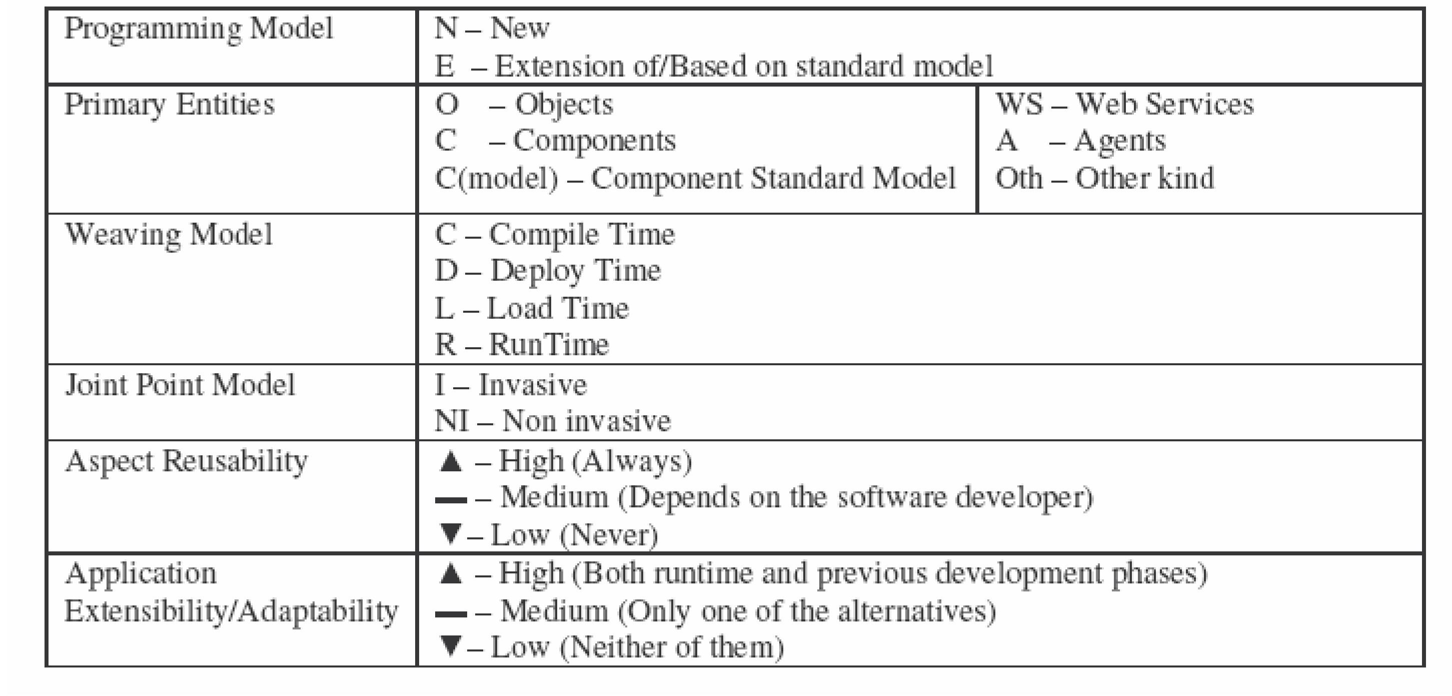
- No other AO approaches for EAI.
- ASIL enables vendor/platform independent aspect descriptions.
  - In other AO-middleware the aspects are dependent on a particular framework, and as a result have to be modified when reused within a different framework. Therefore not suitable for EAI.

References

- Gregor Kiczales, J.I., John Lamping, Jean Marc Loingtie R, Cristina Videria Lopes, Chris Maeda, Anurag Mendhekar, Aspect-Oriented Programming. ECOOP 1997:p. 220-243
- Induruwana, C.D, Using an Aspect Oriented Layer in SOA for Enterprise Application Integration. Pending Publication Workshop ICSOC 2005
- Sayavedra, A.L.a.L., EAI Business Drivers. EAI Journal, 2003. 2:p.27-29

Questions?

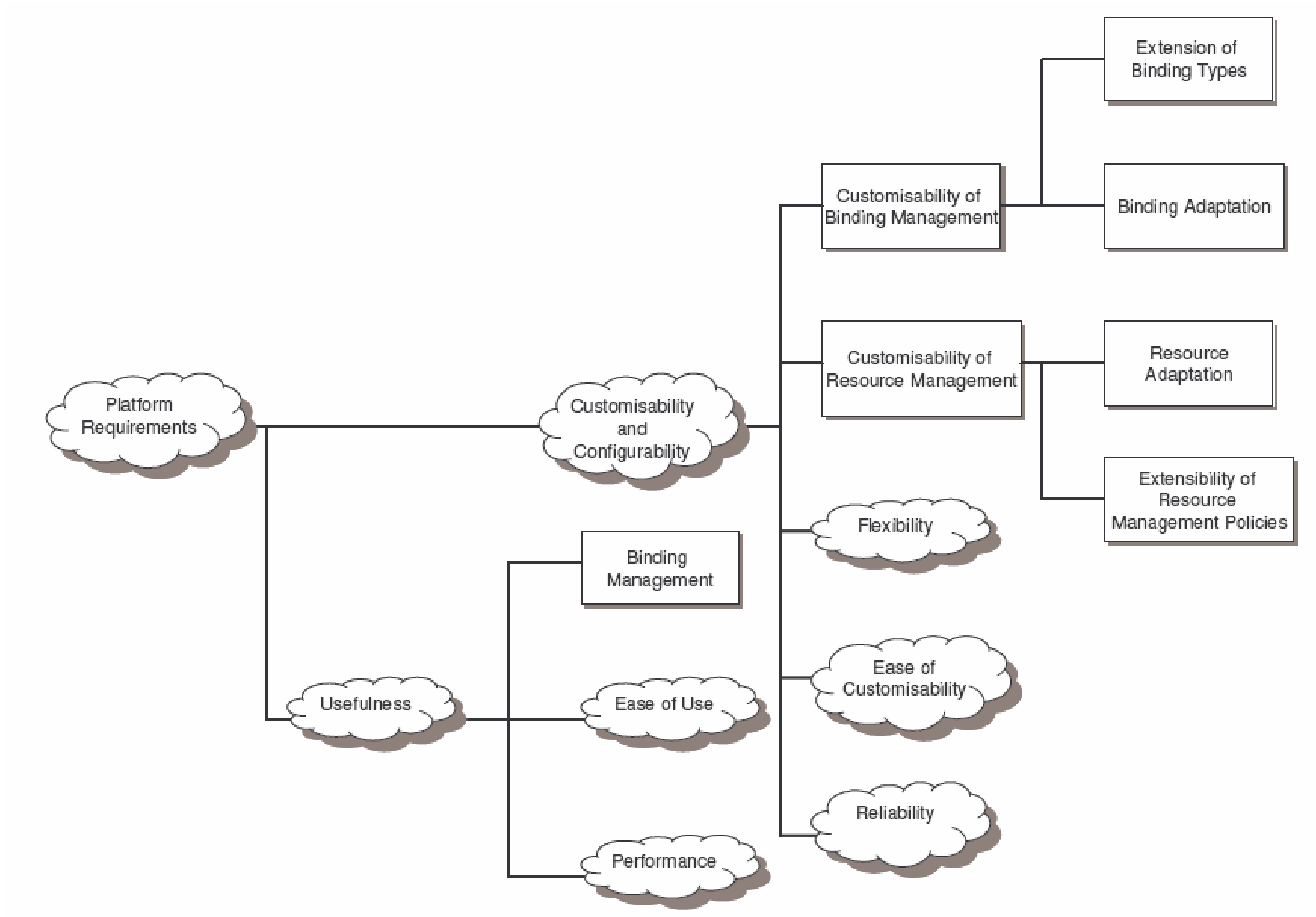
Aspect oriented evaluation criteria



Programming Model	N – New E – Extension of/Based on standard model	
Primary Entities	O – Objects C – Components C(model) – Component Standard Model	WS – Web Services A – Agents Oth – Other kind
Weaving Model	C – Compile Time D – Deploy Time L – Load Time R – RunTime	
Joint Point Model	I – Invasive NI – Non invasive	
Aspect Reusability	▲ – High (Always) ▬ – Medium (Depends on the software developer) ▼– Low (Never)	
Application Extensibility/Adaptability	▲ – High (Both runtime and previous development phases) ▬ – Medium (Only one of the alternatives) ▼– Low (Neither of them)	

Monica Pinto, L.F.F., Pablo Sánchez, Matthew Webster, Adrian Colyer, Neil Loughran, Nikos Parlavantzas, *Survey of Aspect Oriented Middleware.* Survey Version: 1.0 AOSD-Europe-ULANC-10, 2005.

Hierarchical structures for middleware evaluation



Platform Requirements
Customisability and Configurability
Customisability of Binding Management
Extension of Binding Types
Binding Adaptation
Customisability of Resource Management
Resource Adaptation
Extensibility of Resource Management Policies
Flexibility
Ease of Customisability
Reliability
Usefulness
Binding Management
Ease of Use
Performance

Monica Pinto, L.F.F., Pablo Sánchez, Matthew Webster, Adrian Colyer, Neil Loughran, Nikos Parlavantzas, *Survey of Aspect Oriented Middleware.* Survey Version: 1.0 AOSD-Europe-ULANC-10, 2005.