

About my 6 months in Lancaster

Ambra Molesini,

`ambra.molesini@unibo.it`

ALMA MATER STUDIORUM—Università di Bologna

Cesena, 27th July 2007



- 1 Aspect-Oriented Software Development
- 2 Software Architectures and Architectural Styles
- 3 What I have done



Concern

Concern

is a general term that refers to any particular piece of interest or focus in a program

are aspects of a program which affect (crosscut) other concerns. These concerns often cannot be cleanly decomposed from the rest of the system in both the design and implementation, and result in either *scattering* or *tangling* of the program, or both



Concern

Concern

is a general term that refers to any particular piece of interest or focus in a program

Crosscutting Concerns

are aspects of a program which affect (crosscut) other concerns. These concerns often cannot be cleanly decomposed from the rest of the system in both the design and implementation, and result in either **scattering** or **tangling** of the program, or both



Aspect

An Aspect

is a part of a program that *cross-cuts* its core concerns, therefore violating its *separation of concerns*

entails breaking down a program into distinct parts that overlap in functionality as little as possible



Aspect

An Aspect

is a part of a program that *cross-cuts* its core concerns, therefore violating its *separation of concerns*

Separation of concerns

entails breaking down a program into distinct parts that overlap in functionality as little as possible



Example

- Logging code can cross-cut many modules, yet the aspect of logging should be separate from the functional concerns of the module it cross-cuts
- Isolating such aspects as logging and persistence from business logic is the aim of the *aspect-oriented software development*, of which aspect-oriented programming paradigm is the most established area



AOP

- One view of aspect-oriented programming (AOP) is that every major feature of the program, core concern or cross-cutting concern, is an aspect
- and by weaving them together (also called composition), you finally produce a whole out of the separate aspects



Join Point

- A *join point* is a point in the control flow of a program. A join point is where the main program and the aspect meet.
- A *pointcut* is a set of join points
- Whenever the program execution reaches one of the join points described in the pointcut, a piece of code associated with the pointcut is executed
- This permits the addition of aspects to existing software, or the design of software with a clear separation of concerns, where the programmer weaves merge different aspects into a complete application



Join Point

- A *join point* is a point in the control flow of a program. A join point is where the main program and the aspect meet.
- A *pointcut* is a set of join points
- Whenever the program execution reaches one of the join points described in the pointcut, a piece of code associated with the pointcut is executed
- This permits the addition of aspects to existing software, or the design of software with a clear separation of concerns, where the programmer weaves merge different aspects into a complete application



Classic Definitions

Kruchten: The Rational Unified Process 1999

An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition



Classic Definitions

Garlan and Shaw 1993

...beyond the algorithms and data structures of the computation; designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives



Modern Definition

Software Architecture in Practice (2nd edition), 2003

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



So...

- The architecture defines elements: it embodies information about how the elements relate to each other
- This means that architecture *specifically omits* certain information about elements that does not pertain to their interaction
- Thus, an architecture is foremost an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements



So...

- Every software system has an architecture because every system can be shown to be composed of elements and relations among them
- The behavior of each element is part of the architecture as that behavior can be observed or discerned from the point of view of another element



Architecture Description Languages

- Architecture description languages (ADLs) are used to describe a Software Architecture
- Several different ADLs have been developed by different organizations:
 - Wright (developed by Carnegie Mellon),
 - Acme (developed by Carnegie Mellon),
 - xADL (developed by UCI),
 - Darwin (developed by Imperial College London)
- Common elements of an ADL are *component*, *connector* and *configuration*



Architectural Styles or Architectural Patterns

Architectural Style

defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style determines the *vocabulary of components and connectors* that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions



Style Examples:

- Pipe and Filter
- Client-Server
- Blackboard
- Layer
- . . .

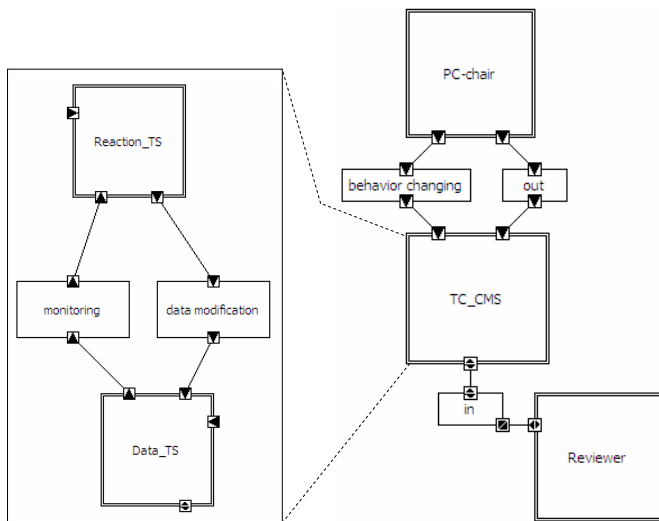


My work

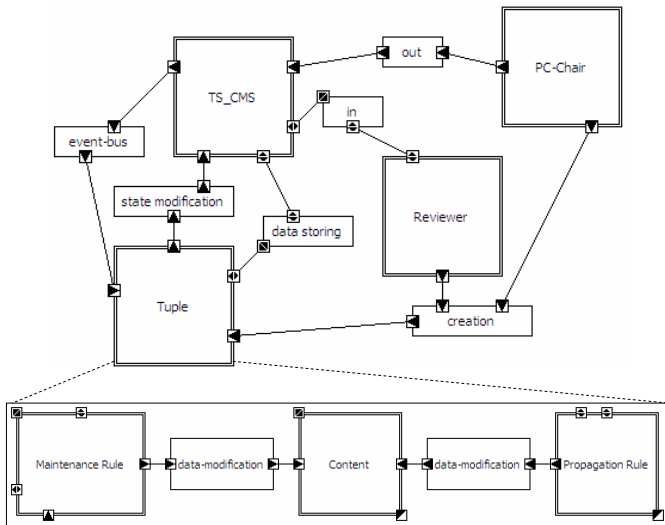
- I have studied the basic principles of AOSD and Software Architectures
- I have defined two different architectural styles for the MAS:
 - Reactive Coordination (from TuCSoN)
 - Stigmergic Coordination (from TOTA)
- I have used the CMS as a case study



Reactive Coordination



Stigmergic Coordination



My work

- I have studied how three different crosscutting concerns (coordination, security and error handling) affect the two styles
- I have studied and compared two different ADLs (Acme e xADL) and the relative tools
- I have represented my styles by xADL
- I have studied the meaning of join points and the pointcuts at the architectural level



Currently

- I'm studying the different composition techniques for the styles (Reactive Coordination = Blackboard + something)
- I'm comparing the semantical and syntactical representation of the pointcuts
- To do this I'm studying some metrics for the evaluation of the composition stability...
- I don't agree with Alessandro's methods..



Future

- I would must study Software Architectures for MAS
- I would must define other Architectural Styles for MAS
- **without** aspects



About my 6 months in Lancaster

Ambra Molesini,

`ambra.molesini@unibo.it`

ALMA MATER STUDIORUM—Università di Bologna

Cesena, 27th July 2007

