Aspect-oriented Software Development

Chapter 32 – Sommerville 8th Edition

Aspect-oriented software development

- An approach to software development based around a new type of abstraction - an aspect.
- Used in conjunction with other approaches - normally object-oriented software engineering.
- Aspects encapsulate functionality that cross-cuts and co-exists with other functionality.
- Aspects include a definition of where they should be included in a program as well as code implementing the cross-cutting concern.

The separation of concerns

- The principle of separation of concerns states that software should be organized so that each program element does one thing and one thing only.
- Each program element should therefore be understandable without reference to other elements.
- Program abstractions (subroutines, procedures, objects, etc.) support the separation of concerns.

Concerns

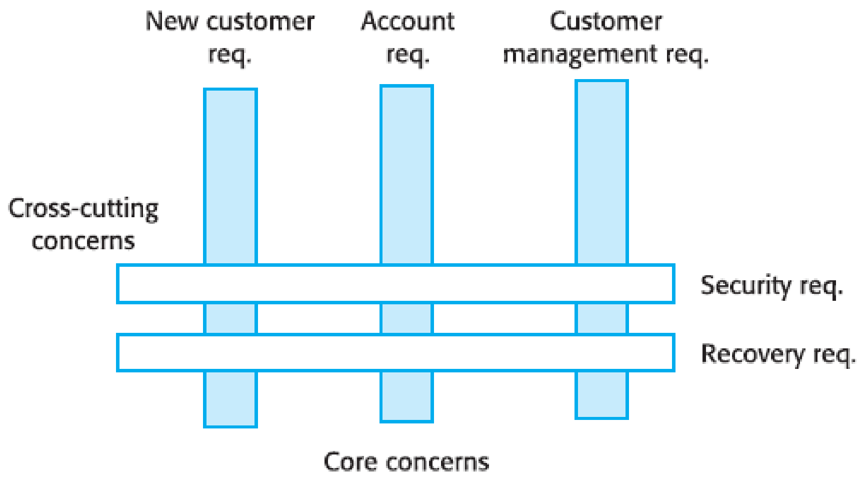
- Concerns are not program issues but reflect the system requirements and the priorities of the system stakeholders.
    - Examples of concerns are performance, security, specific functionality, etc.
- By reflecting the separation of concerns in a program, there is clear traceability from requirements to implementation.
- Core concerns are the functional concerns that relate to the primary purpose of a system; secondary concerns are functional concerns that reflect non-functional and QoS requirements.

Stakeholder concerns

- Functional concerns which are related to specific functionality to be included in a system.
- Quality of service concerns which are related to the non-functional behaviour of a system.
- Policy concerns which are related to the overall policies that govern the use of the system.
- System concerns which are related to attributes of the system as a whole such as its maintainability or its configurability.
- Organisational concerns which are related to organisational goals and priorities such as producing a system within budget, making use of existing software assets or maintaining the reputation of an organisation.

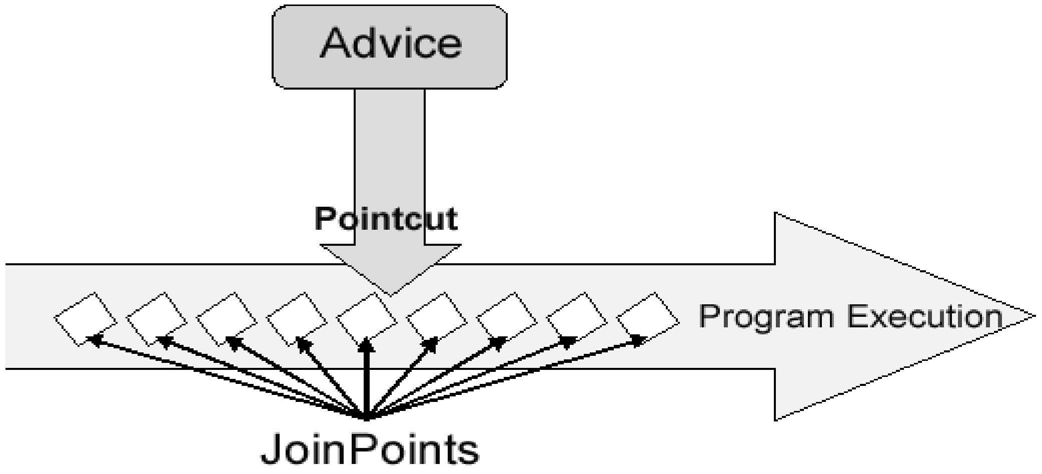
Cross-cutting concerns

- Cross-cutting concerns are concerns whose implementation cuts across a number of program components.
- Cross-cutting concerns are parts of a program that rely on or must affect many other parts of the system.
- This results in problems when changes to the concern have to be made - the code to be changed is not localized but is in different places across the system.
- Cross cutting concerns lead to tangling (significant dependencies between systems) and scattering (code duplication).

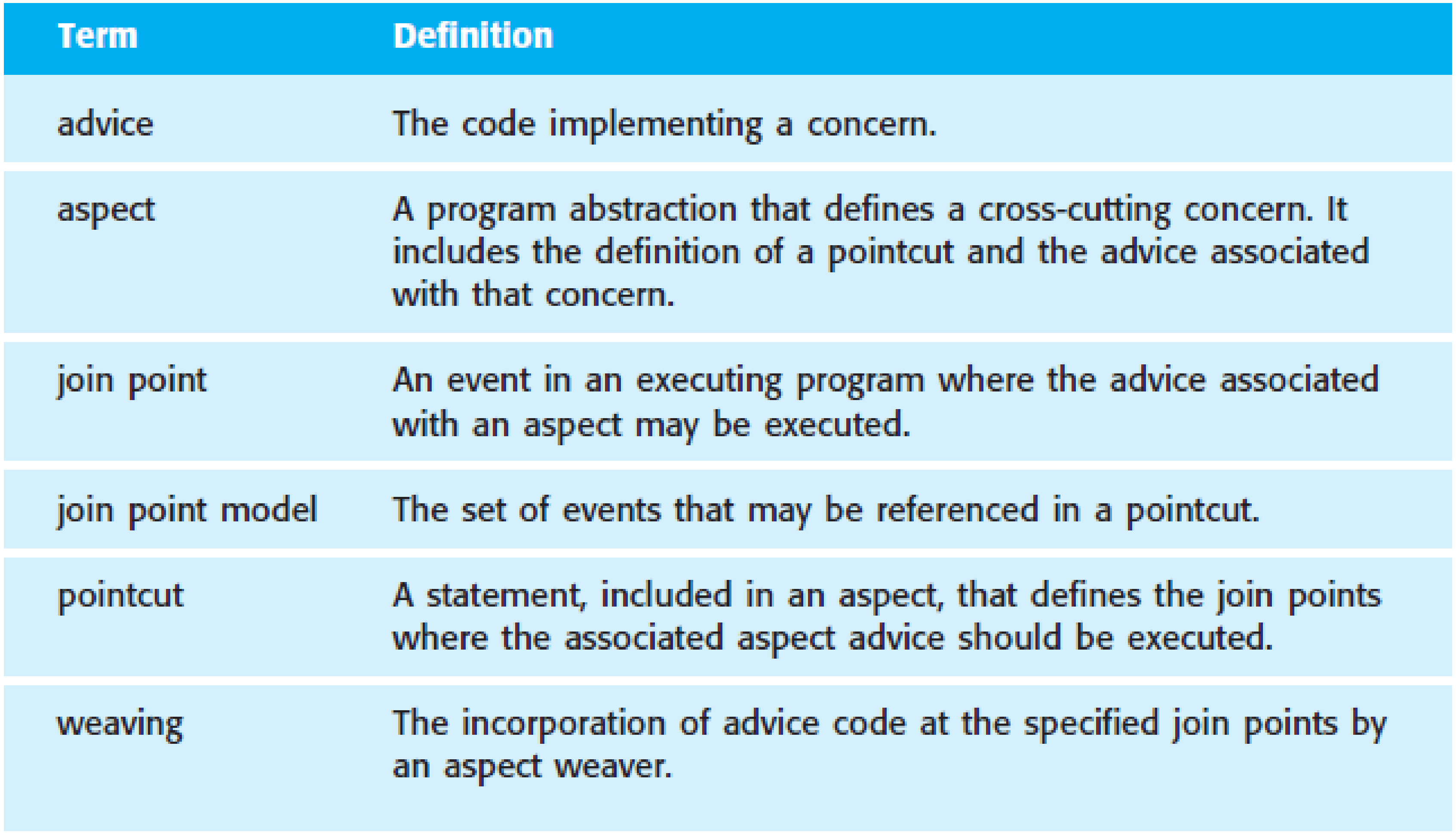


Aspects, join points and pointcuts

- An aspect/advice is an abstraction which implements a concern. It includes information where it should be included in a program.
- A join point is a place in a program where an aspect may be included (woven). This point could be a method being called, an exception being thrown, or even a field being modified. These are the points where your aspect's code can be inserted into the normal flow of your application to add new behavior.
- A pointcut defines where (at which join points) the aspect will be included in the program.



Aspect terminology



Term	Definition
advice	The code implementing a concern.
aspect	A program abstraction that defines a cross-cutting concern. It includes the definition of a pointcut and the advice associated with that concern.
join point	An event in an executing program where the advice associated with an aspect may be executed.
join point model	The set of events that may be referenced in a pointcut.
pointcut	A statement, included in an aspect, that defines the join points where the associated aspect advice should be executed.
weaving	The incorporation of advice code at the specified join points by an aspect weaver.

THANK YOU!!!