Software Development Methodology (SDM)

- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems

- Consisting of two main parts:
  - A set of modeling conventions comprising a *Modeling Language* (syntax and semantics)
  - A *Process*, which
    - provides guidance as to the order of the activities,
    - specifies what artifacts should be developed using the *Modeling Language*,
    - directs the tasks of individual developers and the team as a whole, and
    - offers criteria for monitoring and measuring a project’s products and activities.
Object-Oriented Software Development Methodology (OOSDM)

- Specifically aimed at viewing, modeling and implementing the system as a collection of interacting objects

- First appeared in late 1980s

- Categorized as
  - Seminal (First and Second Generations)
  - Integrated (Third Generation)
  - Agile

- UML was the result of the ‘war’ among seminal methodologies

- Process has now replaced modeling language as the main contentious issue
Genealogy: Seminal and Integrated Methodologies (until 1996)

[Webster 1996]
Software Development Methodologies – Lecture 1

UML

Database design

Entity-Relational Modelling
Codd et al., 1980
Multiple views of static relationships; design based on users’ concepts

Ada
Large system modular real-time programming

Smalltalk

C++

CRC
Book et al.

Eiffel
Meyer, 1988

Formal specification
Larch, Z & VDM c. 1980

Fusion
Coleman et al., 1994

Objectory
Jacobson et al., 1992
A process for object-oriented design

Syntropy
Cook & Daniels, 1994

SOMA
Graham, 1991
Rulesets, OO RAD, business modelling

SoaM
Rumbaugh et al., 1991
Managing object designs and interdependencies

OPEN
Graham et al., 1997
Process, notation

Objective

Odell, 1991
Activity diagrams

Real Time OOM
Selic et al., 1994

Object Management Group
co-ordinator

UML
1997

Catalysis
D’Souza and Wills, 1999

[Graham 2001]
Genealogy: Agile Methodologies

[Abrahamsson et al. 2003]
Analysis: Selected Methodologies

- Seminal Methodologies
  3. RDD (1990)
  5. OMT (1991)
  6. OSA (1992)
  7. OOSE (1992)
  8. BON (1992, 1995)
  10. Syntropy (1994)

- Agile Methodologies
  5. dX (1998)

- Process Patterns

- Process Metamodels
  1. OPF – as part of the OPEN methodology (2001)
  2. SPEM (2002)

- MDA

Department of Computer Engineering
Sharif University of Technology
Problems

- Requirements engineering is still the weak link, and requirements traceability is rarely supported.
- Model inconsistency is a dire problem.
- Integrated methodologies are too complex to be effectively mastered, configured, and enacted.
- Agile methods are not mature enough:
  - Unrealistic assumptions (e.g. Scrum)
  - Lack of scalability (All, more or less)
  - Lack of a specific, unambiguous process (e.g. XP, Crystal)

- Seamless development, pioneered by seminal methodologies, is not adequately appreciated and supported in modern-day methodologies.
Methodology Development

- Methodologies can be categorized according to the circumstances leading to their development, including the approach and method applied:
  - Revolutionary: novel ideas and approaches
  - Evolutionary: based on existing methodologies
    - Extension: adding new features to an existing methodology
    - Integration: consolidating ideas from two or more methodologies
      - Merger: typically carried out through a design-by-committee procedure
      - Ad hoc: features are scavenged from prominent methodologies in order to fill the needs of the methodologist
      - Engineered: based on analysis of the problem domain and requirements thereby identified, and pre-implementation design

- ‘Software processes are software too.’
Analysis Criteria

**OOSDM Process**

- Clarity, rationality, accuracy, and consistency of definition
- Coverage of the generic development lifecycle activities (Analysis, Design, Implementation, Test, Maintenance)
- Support for umbrella activities, especially including:
  - Risk management
  - Project management
  - Quality assurance
- Seamlessness and smoothness of transition between phases, stages and activities
- Basis in the requirements (functional and non-functional)
- Testability and Tangibility of artifacts, and traceability to requirements
- Encouragement of active user involvement
- Practicability and practicality
- Manageability of complexity
- Extensibility/Configurability/Flexibility/Scalability
- Application scope
Analysis Criteria

**OOSDM Modeling Language**

- Support for consistent, accurate and unambiguous object-oriented modeling:
  - Structural – Functional – Behavioural
  - Logical to Physical (business-process/problem domain to solution domain to implementation domain)
  - Enterprise level – System level – Subsystem/Package level – Inter-object level – Intra-object level
  - Formal and Informal modeling facilities

- Provision of strategies and techniques for tackling model inconsistency and managing model complexity
References

