Software Development Methodologies

Lecturer: **Raman Ramsin**

Lecture 1: Basics

Software Development Methodology (SDM)

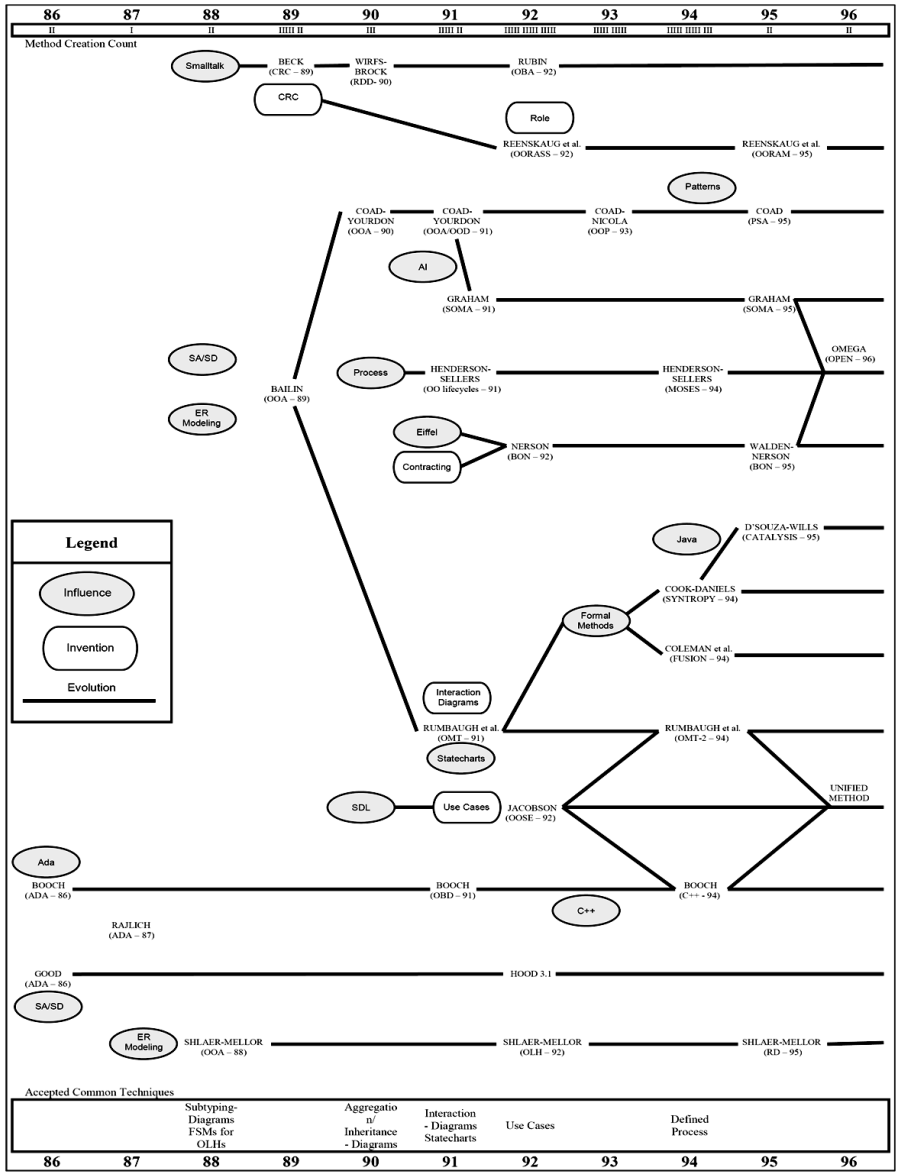
- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems

- 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)

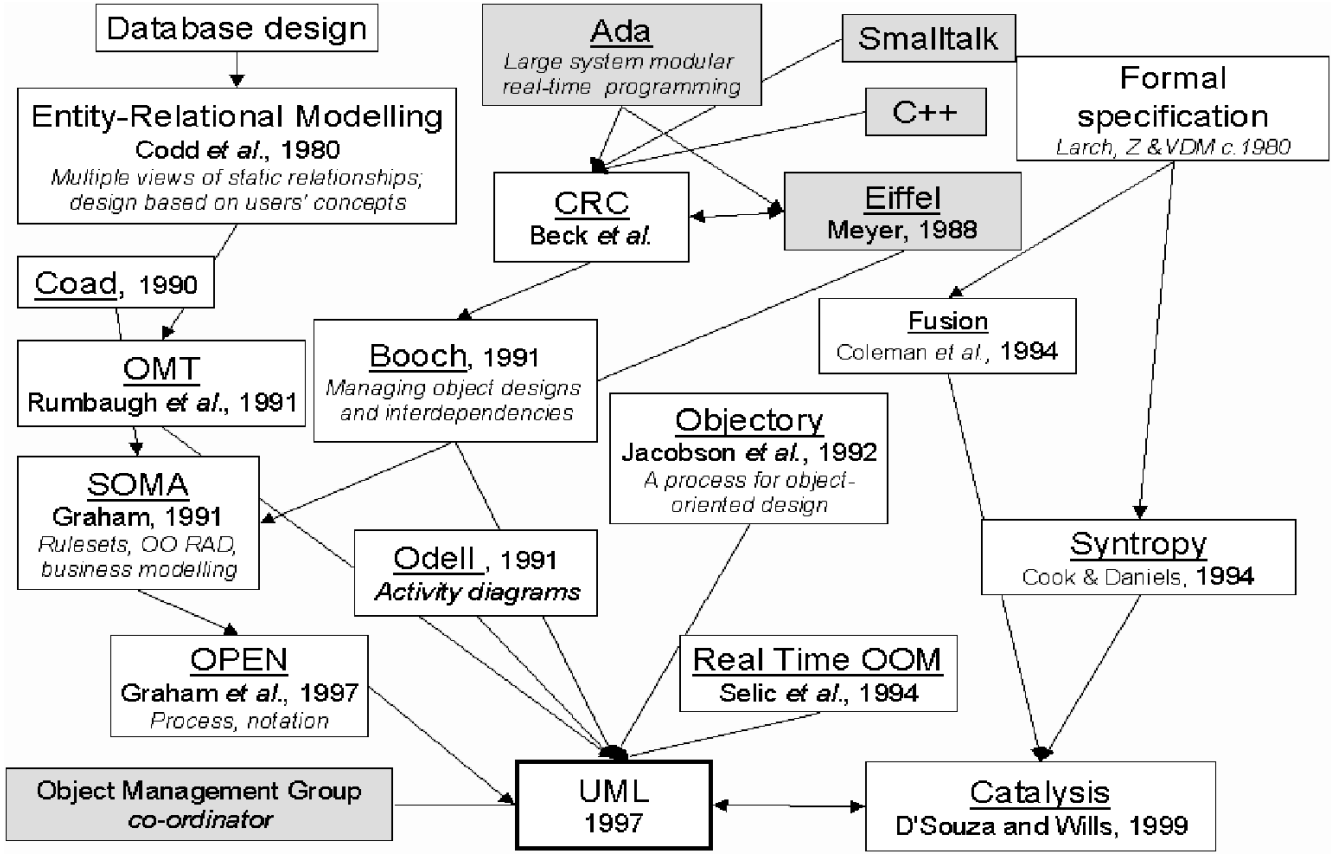
- Specifically aimed at viewing, modeling and implementing the system as a collection of interacting objects
- First appeared in late 1980s
- Categorized as
  - *Seminal (First and Second Generations)*
  - *Integrated (Third Generation)*
  - *Agile*
- UML was the result of the 'war' among seminal methodologies
- Process has now replaced modeling language as the main contentious issue

Genealogy: Seminal and Integrated Methodologies (until 1996)



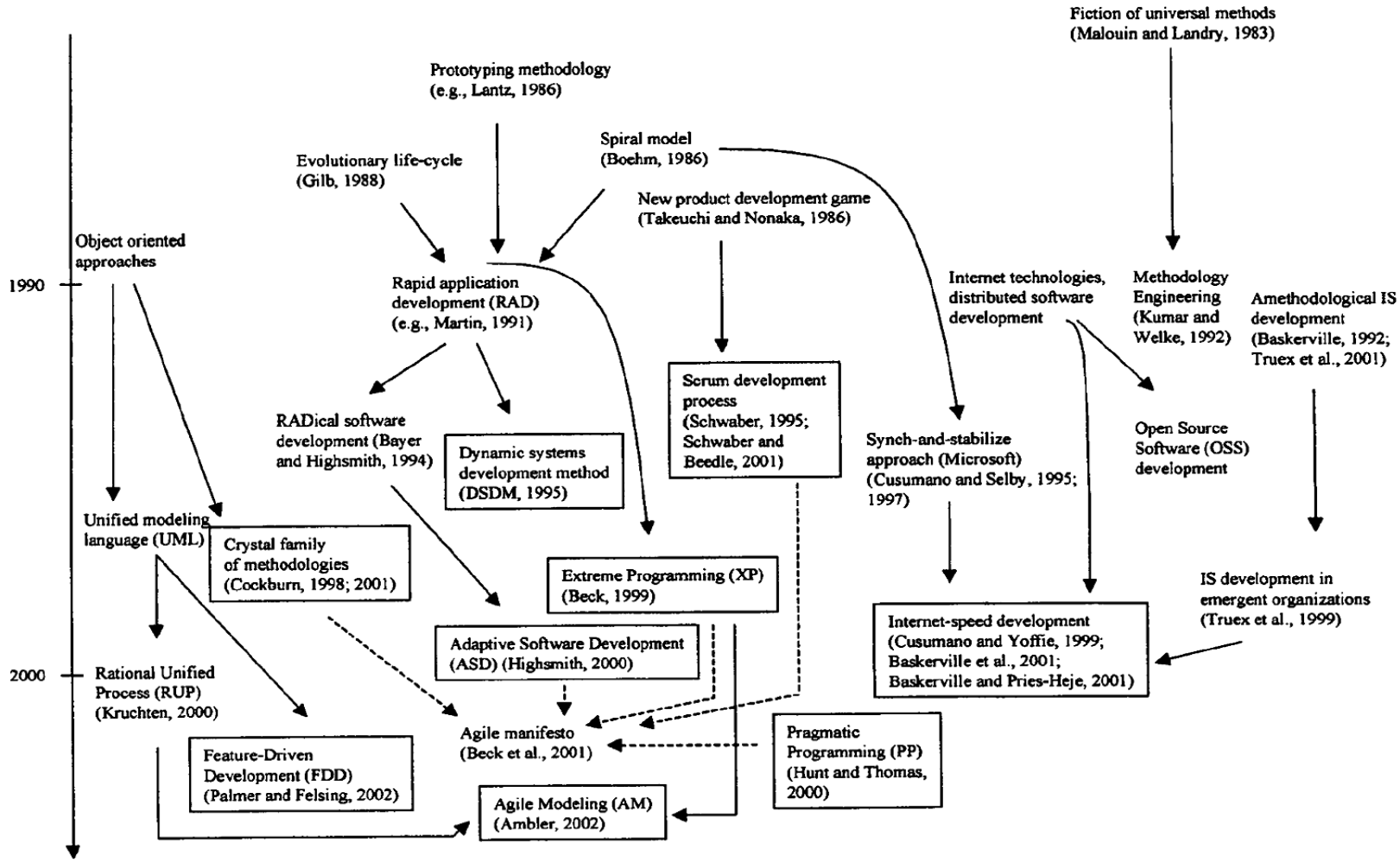
[Webster 1996]

UML



[Graham 2001]

Genealogy: Agile Methodologies



[Abrahamsson et al. 2003]

Analysis: Selected Methodologies

- Seminal Methodologies
  1. Shlaer-Mellor (1988, 1992)
  2. Coad-Yourdon (1989, 1991)
  3. RDD (1990)
  4. Booch (1991, 1994)
  5. OMT (1991)
  6. OOSE (1992)
  7. BON (1992, 1995)
  8. Hodge-Mock (1992)
  9. Syntropy (1994)
  10. Fusion (1994)
- Integrated Methodologies
  1. OPM (1995, 2002)
  2. Catalysis (1995, 1998)
  3. OPEN (1996, 2010)
  4. RUP (1998, 2000, 2003) / USDP (1999)
  5. EUP (2000, 2005)
  6. FOOM (2001, 2007)
  7. TSP/PSP (1999, 2010)
- Agile Methodologies/Frameworks
  1. DSDM (1995, 2003, 2014)
  2. Scrum (1995, 2001, 2016)
  3. XP (1996, 2004, 2013)
  4. ASD (1997, 2000)
  5. Crystal (1998, 2004, 2006)
  6. FDD (1999, 2002)
  7. AUP (2006)
  8. DAD (2012)
- Process Patterns
  1. Ambler (1998)
- Process Metamodels
  1. OPF – as part of the OPEN methodology (2001, 2009)
  2. SPEM 2.0 (2008)
- SME Approaches (2014)

Problems

- Requirements engineering is still the weak link.
- Model inconsistency is a dire problem.
- Integrated methodologies are too complex to be effectively mastered, configured, and enacted.
- Agile methods are not capable 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 – 1*

Criterion 1- Definition:

The methodology should be well-documented; a comprehensive, clear, rational, accurate, detailed and consistent description should be provided.

- What should be captured?
  - Lifecycle and work-units, producers, modeling language, products, techniques and rules, and issues pertaining to umbrella activities.
  - Metamodels suggested by SPEM and OPF provide useful information as to what should be captured in the definition.
- How?
  - Mainly process-centered: the structure of the documentation should closely resemble that of the lifecycle, and everything should be described as secondary to the lifecycle's phases, stages and activities.
  - Role-centered and product-centered views of the methodology can also be added as complements, as suggested by SPEM and OPF.

Analysis Criteria

- *OOSDM Process – 2*

Criterion 2- Coverage of the generic development lifecycle

1. Definition
   - 1.1. Problem domain exploration and modeling
   - 1.2. Requirements elicitation
   - 1.3. Feasibility analysis
2. Development
   - 2.1. Architectural Design
   - 2.2. Detailed Design
   - 2.3. Programming
   - 2.4. Test
   - 2.5. Deployment
3. Maintenance

Analysis Criteria

- *OOSDM Process – 3*

Criterion 3- Support for umbrella activities, especially including:

- Risk Management
  - Incorporating risk assessment and risk mitigation into the lifecycle.
  - Techniques proven effective: preliminary feasibility analysis, prototyping, risk-based planning, iterative-incremental process, active user involvement, early release, continuous V&V, iterative reviews, and continuous integration.
- Project Management
  - Incorporating planning, scheduling and control techniques.
  - Plans and schedules should be iteratively revisited and revised.
  - Special attention should be given to team management: enhancing intra-team and inter-team communication and collaboration.
- Quality assurance
  - Incorporating quality assessment and enhancement techniques.
  - Techniques proven effective: iterative technical reviews, design by contract, continuous V&V, and strategies/techniques enhancing requirements traceability.

Analysis Criteria

- *OOSDM Process – 4*

Criterion 4- Seamlessness, and Smoothness of Transition

- Seamlessness implies lack of paradigm shift in the modeling chain and activities; two common techniques are:
  - basing all tasks and artifacts on a common concept;
  - continuous refinement of a specific set of models, around which the development tasks are oriented.
- Smoothness of transition is needed between phases, stages and tasks.
  - Production of brand new artifacts can damage smoothness of transition, even in seamless methodologies.
- Methodologies that provide smooth transition are not always seamless:
  - agile methodologies provide smooth transition due to their iterative nature and short cycles, yet they can have a huge gap between analysis and coding.

Analysis Criteria

- *OOSDM Process – 5*

Criterion 5- Basis in requirements (functional and non-functional)

- Functional and non-functional requirements should be
  - captured early in the process,
  - modeled in their own right, and
  - used as a basis for design and implementation.
- Requirements-driven methodologies go one step further.
- It is desirable to allow the requirements to evolve.

Analysis Criteria

- *OOSDM Process – 6*

Criterion 6- Testability, Tangibility, Traceability to Requirements

- Testability of the artifacts depends on their complexity:
  - Artifacts should be few, simple, and understandable, with dependencies that are minimal and clearly defined.
  - Artifacts should complement each other in the context of the process, not decorate each other with clutter.
- Tangibility of the artifacts depends on the observer:
  - to Users: Executable artifacts and artifacts with syntax and semantics that are understandable to the user;
  - to Developers: Artifacts that are visibly useful in the process.
- Traceability to requirements: artifacts should be traceable
  - as direct or indirect realizations of the requirements, or
  - via the use of requirements-based evaluation scenarios.

Analysis Criteria

- *OOSDM Process – 7*

Criterion 7- Encouragement of active user involvement

- Vital for risk management and quality assurance.
- Proven techniques:
  - Ambassador users;
  - Planning and review sessions with user participants.
- Agile methodologies have a great deal to offer in this regard.

Analysis Criteria

- *OOSDM Process – 8*

Criterion 8- Practicability and Practicality

- Practicability means employability of the process; it can depend on:
  - complexity of the process;
  - nature of the target project.
- Practicality: The process should lend itself to effective and efficient use. Numerous factors can affect practicality:
  - Complexity of process units.
  - Tasks that distract the developers from mainstream activities or encumber them with unnecessary detail; focus the developers by using techniques such as team management sessions, requirements-based models, and system architecture.
  - Dependence on error-prone techniques and strategies.
  - Dependence on special tools and technologies.
  - Lack of adequate project management strategy.

Analysis Criteria

- *OOSDM Process – 9*

Criterion 9- Manageability of complexity

- The complexity of work-units should be manageable; two techniques are commonly used for this purpose:
  - Partitioning
  - Layering

Analysis Criteria

- *OOSDM Process – 10*

Criterion 10- Extensibility/Scalability/Configurability/Flexibility

- Extensibility: The process should be an extensible core, with extension points and mechanisms explicitly specified.
- Scalability: The process should be applicable to projects of different sizes and criticalities.
- Configurability: The ability to configure the process at the start of the project in order to fit it to the project situation.
- Flexibility: The process should be tunable during enactment.
  - Useful techniques include: iterative process review sessions, and feedback-based revisions.

Analysis Criteria

- *OOSDM Process – 11*

Criterion 11- Application scope

- The application scope of the process depends on the intended usage context.
- However, targeting **Information Systems** as a general usage context seems to be a logical minimum expectation:
  - This will address the minimum modeling needs of a general process.

Analysis Criteria

- *OOSDM Modeling Language – 1*

Criterion 1- Support for consistent, accurate and unambiguous object-oriented modeling

- Diverse modeling viewpoints: Structural – Functional – Behavioural
- Logical to Physical modeling: business-process/problem domain to solution domain to implementation domain
- Diverse levels of abstraction and granularity: Enterprise – System – Subsystem/Package – Inter-object – Intra-object
- Formal and Informal modeling facilities

Analysis Criteria

- *OOSDM Modeling Language – 2*

Criterion 2- Provision of strategies and techniques for tackling model inconsistency and managing model complexity

- Modeling languages can facilitate consistency-checking by providing semantics which define model dependencies and constraints.
  - A noteworthy contribution in this regard is OPM's single-model approach, which facilitates consistency-checking through eliminating model multiplicity.
- Modeling languages should include constructs for facilitating complexity management (e.g., UML packages and components).

References

- Webster, S., "On the evolution of OO methods", Bournemouth University, 1996.
- Graham, I., Object-oriented Methods: Principles and Practice (3rd Edition), Addison-Wesley, 2001.
- Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J., "New Directions on Agile Methods: a comparative analysis", Proceedings of the International Conference on Software Engineering – ACM/ICSE 2003, 2003, pp. 244-254.
- Ramsin, R., Paige, R. F., "Iterative Criteria-Based Approach to Engineering the Requirements of Software Development Methodologies", IET Software, vol. 4, no. 2 (April), 2010, pp. 91-104.