An Overview of Objectory

Object-Oriented Analysis and Design

CSCI 6448 - Fall 1998

Kenneth M. Anderson

Review

- A software lifecycle governs the development of a software product
- Many lifecycles exist including those that facilitate object-oriented analysis and design
- Objectory is one such process being developed by “the three amigos”

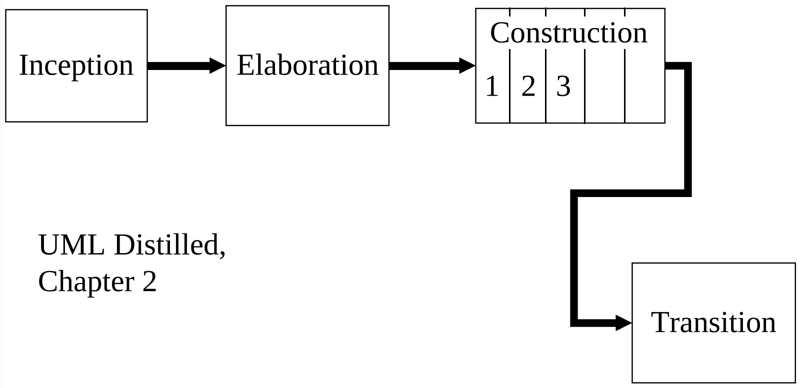
Lifecycle ≠ Notation

- This class will discuss the Unified Modeling Language (UML) notation
- A notation provides symbols to record requirements and design decisions
- Thus, the UML is not a software lifecycle
- However, it can be used by many different lifecycles to produce lifecycle artifacts

Not exactly Objectory

- The “official” book on Objectory has not yet been released. (Its late.)
  - “The Objectory Software Development Process”, due Fall, 1998
- As a result, the process described in this lecture is “Objectory-Compliant”
- For full details, get the book...

Overview



UML Distilled,
Chapter 2

Inception

- High-level planning for the project
- Determine the project’s scope
- If necessary
  - Determine business case for the project
    - Estimate cost and projected revenue
- Essentially what you are doing now with your requirements documents!

Elaboration

- Develop requirements and initial design
- Develop Plan for Construction phase
- Risk-driven approach
  - Requirements Risks
  - Technological Risks
  - Skills Risks
  - Political Risks

Requirements Risks

- Is the project technically feasible?
- Is the budget sufficient?
- Is the timeline sufficient?
- Has the user really specified the desired system?
- Do the developers understand the domain well enough?

Dealing with Requirements Risks

- Construct models to record Domain and/or Design knowledge
  - Domain model (vocabulary)
  - Use Cases (discussed next week)
  - Design model
    - Class diagrams
    - Activity diagrams
- Prototype construction

Dealing with Requirements Risks, continued.

- Begin by learning about the domain
  - Record and define jargon
  - Talk with domain experts
    - Oftentimes end-users!
- Next construct Use cases
  - What are the required external functions of the system?
  - Iterative process; Use Cases can be added as they are discovered (more info next week)

Dealing with Requirements Risks, continued.

- Finally, construct Design model
  - Class diagrams identify key domain concepts and their high-level relationships
  - Activity diagrams highlight the domain's work practices
    - A major task here is identifying parallelism that can be exploited later
- Be sure to consolidate iterations into a final consistent model

Dealing with Requirements Risks, continued.

- Build prototypes
  - Used only to help understand requirements
  - Throw them all out!
    - Do not be tied to an implementation too early
    - Make use of rapid prototyping tools
      - 4th Generation Programming Languages
      - Scripting and/or Interpreted environments
      - UI Builders
- Be prepared to educate the client as to the purpose of the prototype

Technology Risks

- Are you tied to a particular technology?
- Do you “own” that technology?
- Do you understand how different technologies interact?
- Techniques
  - Prototypes!
  - Class diagrams, package diagrams

Skill Risks

- Do the members of the project team have the necessary skills and background to tackle the project?
- If not
  - Training, Consulting, Mentoring and Hiring new people are available options!

Political Risks

- How well does the proposed project mesh with corporate culture?
  - Consider the attempt to use Lotus Notes at Arthur Anderson
    - Lotus Notes attempts to promote collaboration
    - Arthur Anderson consultants compete with each other!
  - Consider e-mail: any employee can ignore the org chart and mail the CEO!

Political Risks, continued

- Will the project directly compete with another business unit?
- Will it be at odds with some higher level manager’s business plan?
- Etc.
- Any of these can kill a project…
- Examples from students?

Reference

- Lotus Notes vs. Arthur Anderson
  - Orlikowski, W. J. (1992). "Learning from Notes: Organizational Issues in Groupware Implementation". Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work: 362-369.
- If you are interested you can get a copy from me later in the Semester...

Ending Elaboration

- Baseline architecture Constructed
  - List of Use cases (with estimates)
  - Domain Model
  - Technology Platform
- AND
  - Risks identified
  - Plan constructed
    - Use cases assigned to iterations

Construction

- Each iteration produces a software product that implements the assigned Use cases
  - Additional analysis and design may be necessary as the implementation details get addressed for the first time
- Extensive testing should be performed and the product should be released to (some subset of) the client for early feedback
- More details in the handout

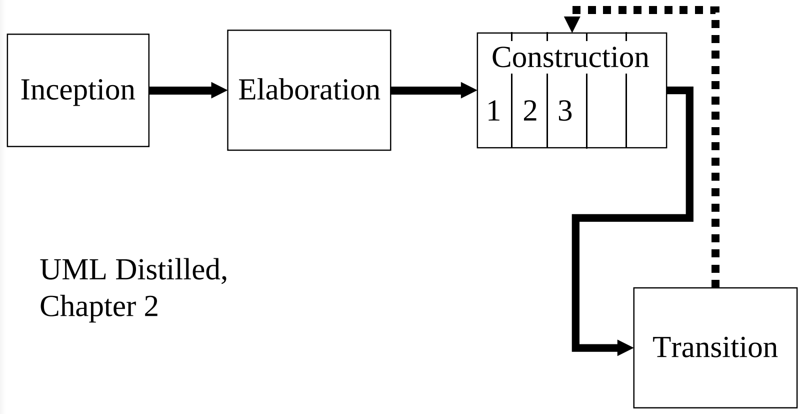
Transition

- Final phase before release 1.0
- Optimizations can now be performed
  - Optimizing too early may result in the wrong part of the system being optimized
  - Largest boosts in performance come from replacing non-scalable algorithms or mitigating bottlenecks

Missing Phase?

- What happened to Operation and Maintenance?
  - The construction phase is iterative. Each iteration produces a product that can be externally delivered. Feedback from that product can drive the next iteration
- Thus, maintenance would be an iteration occurring after transition

Maintenance



UML Distilled,
Chapter 2