Computer Science
UNIVERSITY OF TORONTO

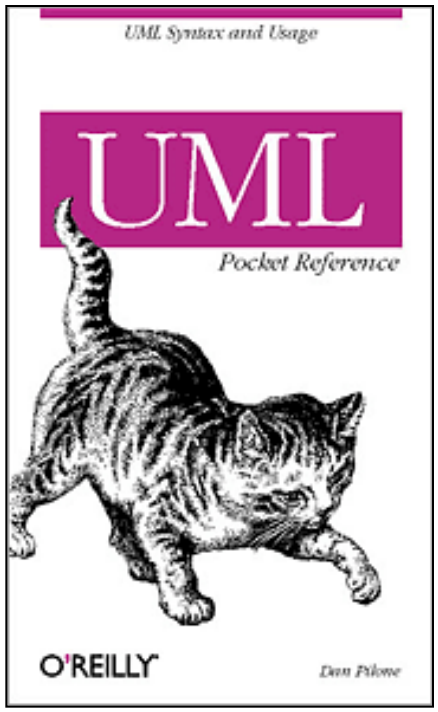
lecture 5: software development lifecycle (sdlc)

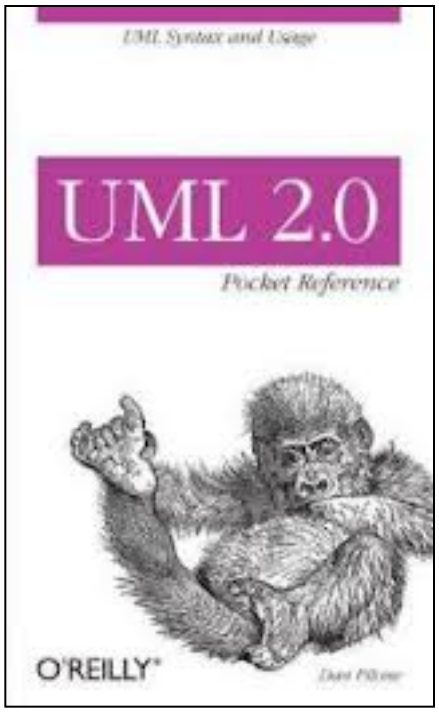
csc302h

winter 2014

recap from last time

- software design with uml sequence & use case diagrams
  - recommend one of these books:



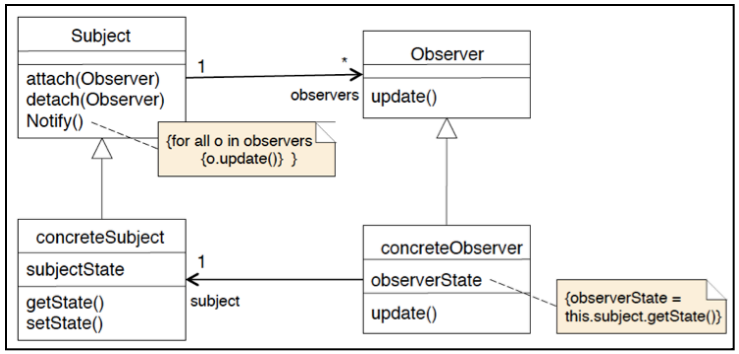


recap from last time (2)

- modeling system behavior with sequence diagrams
  - uml collaboration diagram captures control flow, sequence is a different rendering
  - emphasis is on time and ordering of “messages”
  - objects on top, arrows are messages, time is vertical
  - interaction frames (alt, opt, loop, par, …)

recap from last time (3)

- when to use sequence diagrams?
  - discussing design options
  - explaining design patterns (academic exercise)
    - ex. observer:



  - elaborating on use cases (practical exercise)

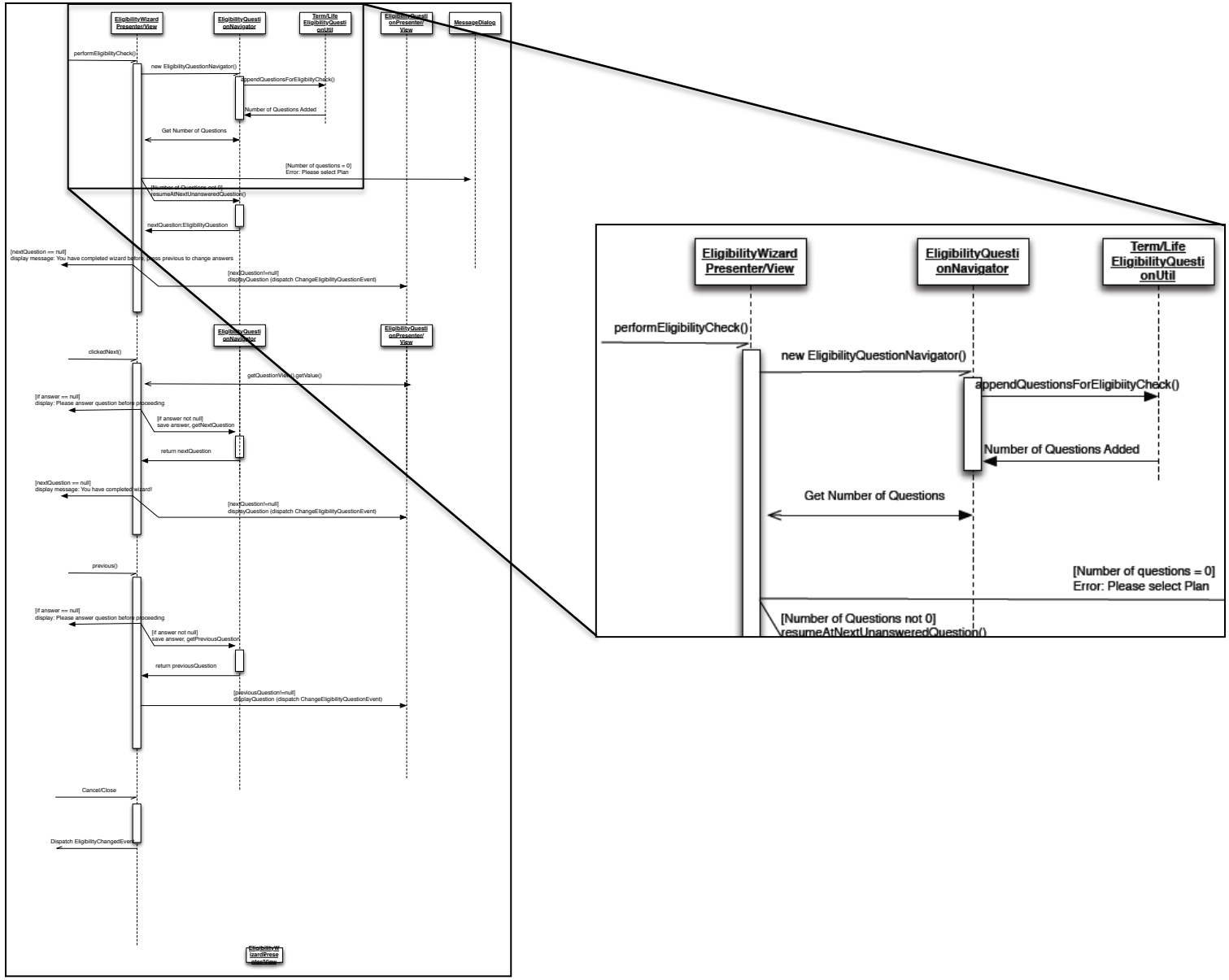
recap from last time (4)

- use case diagrams
  - capture system requirements
  - show how users interact with a system
  - short phrase to sum up a distinct piece of functionality
  - "actors" (stick ppl) show a role that a user takes on during an interaction
  - each use case has one or more actors
  - relationships between use cases like <<extends>>, <<uses>>, <<includes>>
  - reverse engineering use cases

Some real examples of modeling with uml

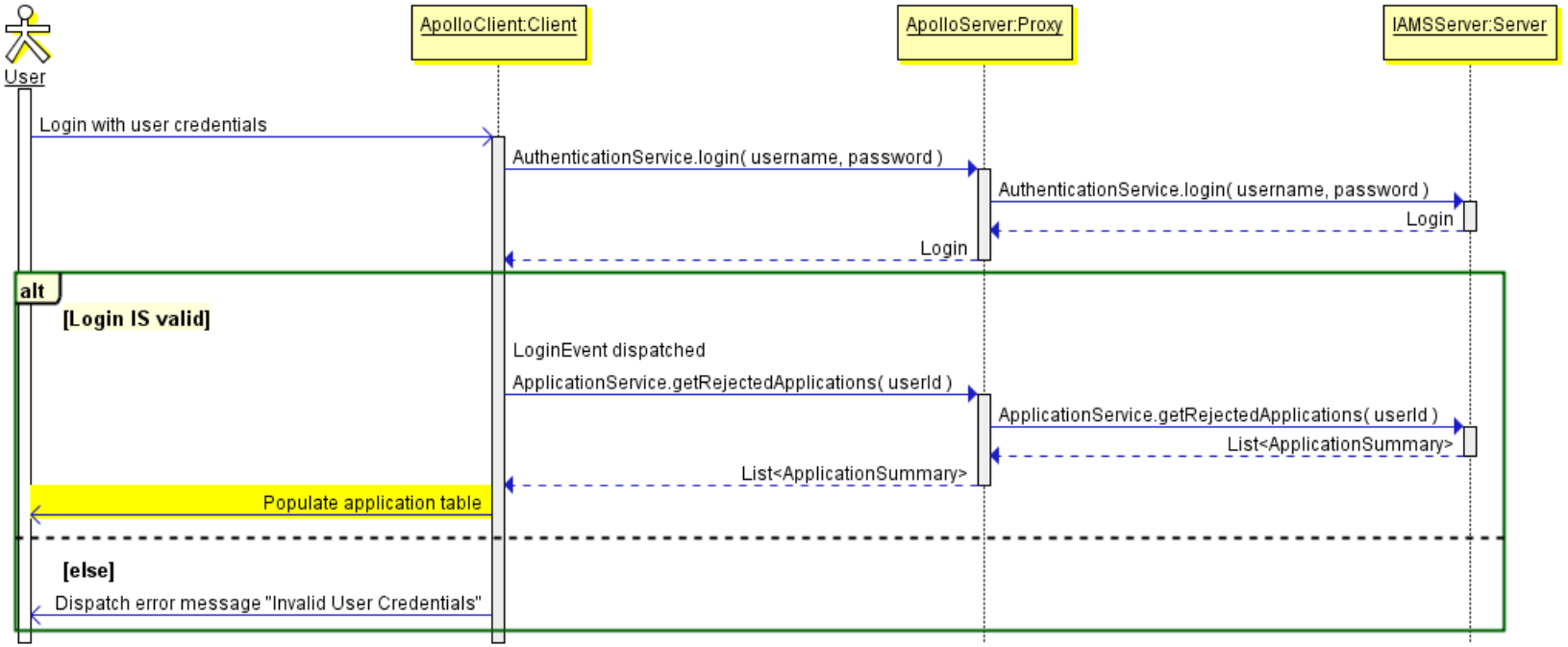
Computer Science
UNIVERSITY OF TORONTO

real-word example #1

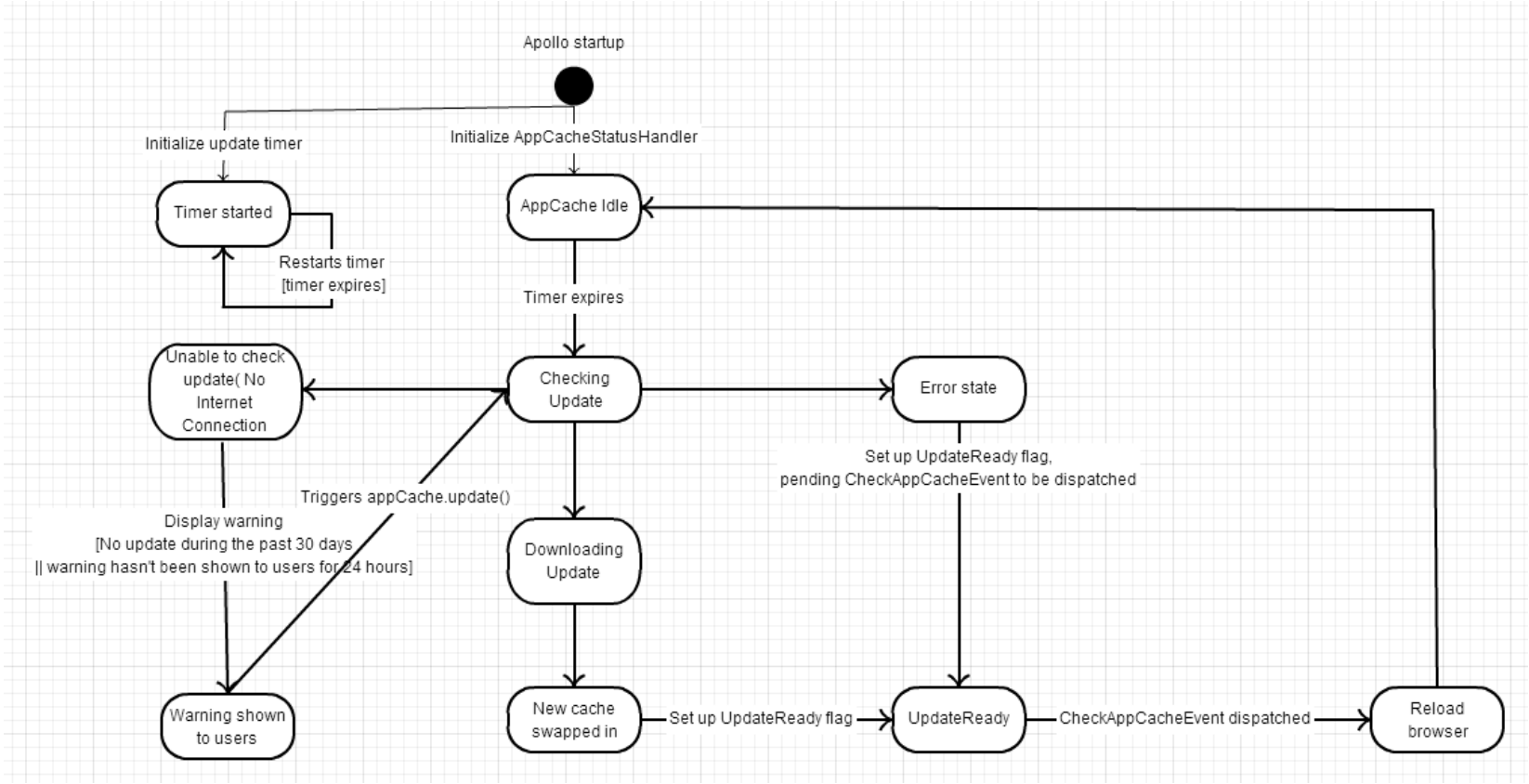


real-word example #2

real-word example #3



real-word example #4

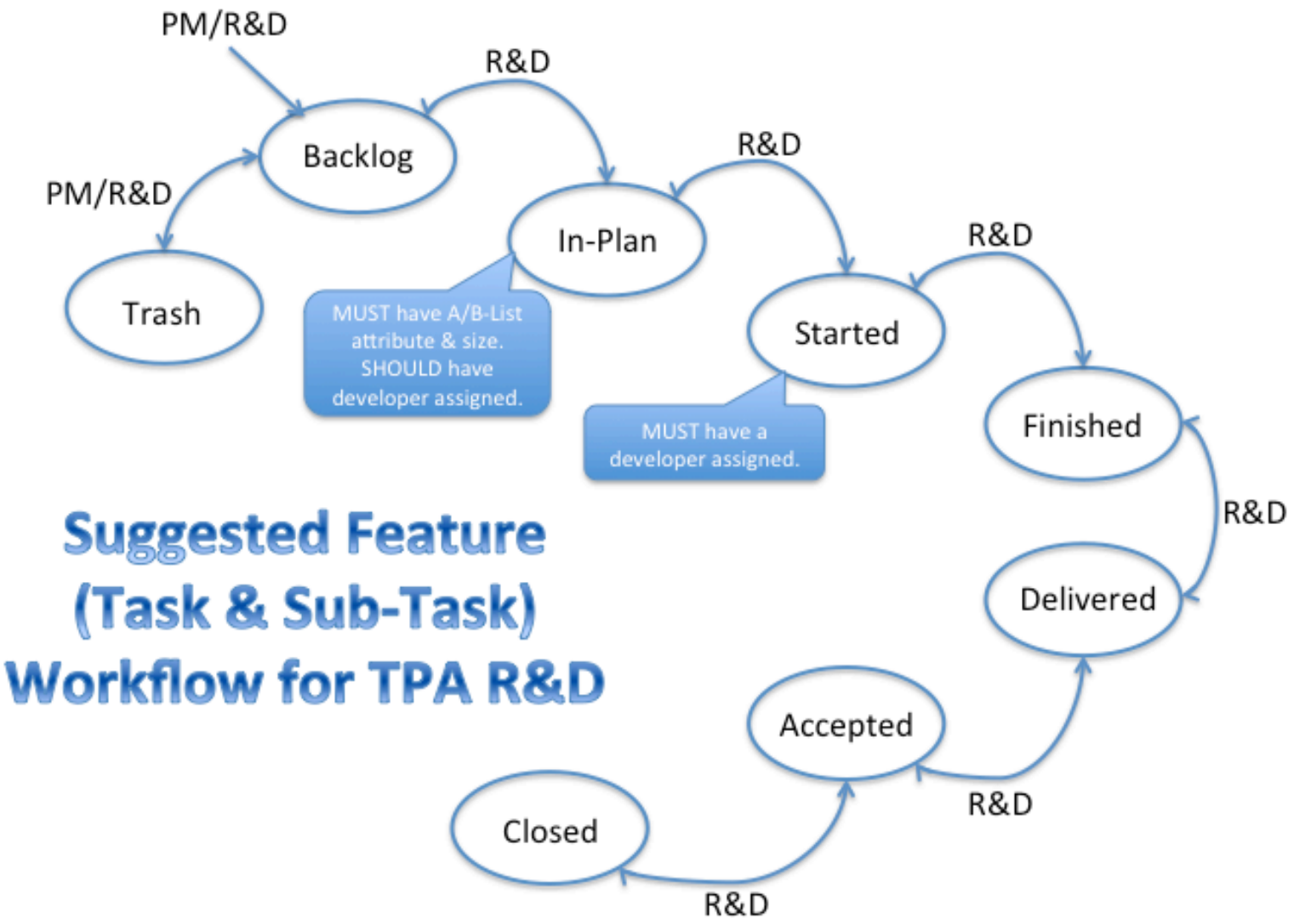


software development lifecycle (sdlc)

sdlc

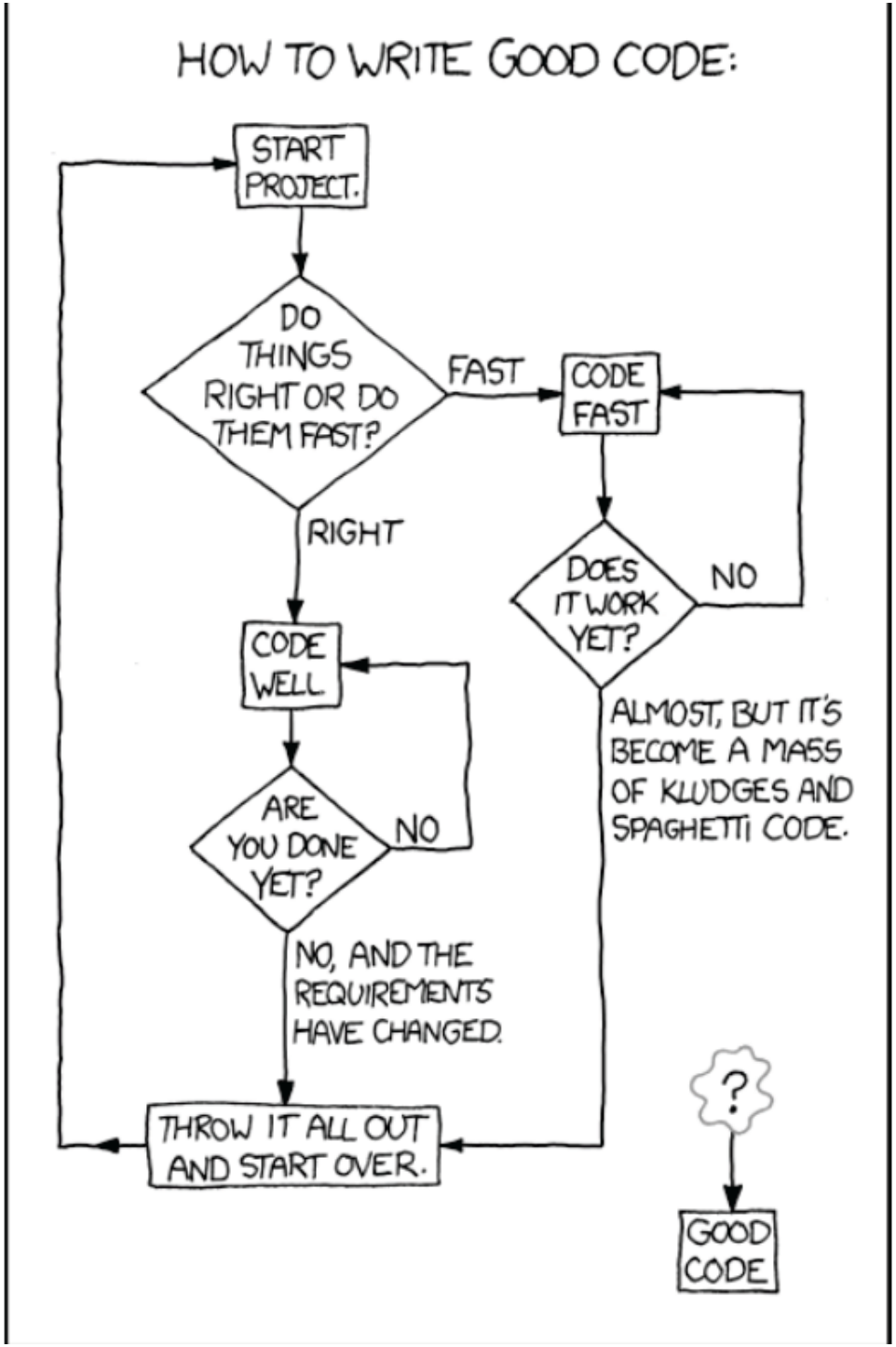
- tend to talk about sdlc in terms of a dichotomy
  - "agile" vs. well…um…"not agile"
  - or, "planned" vs. "continuous"
  - others tend to (incorrectly) think that the deployment method implies the process
    - saas $\Rightarrow$ agile
    - installed $\Rightarrow$ traditional
- think more in terms applying the process on an individual feature, or an aggregate

example feature workflow



Software Processes

- → What is a Software Development Process?
- → The Lifecycle of a Software Project
- → Agile vs. Disciplined
- → Some common approaches:
  - ↳ RUP, SCRUM, XP, ICONIX,...
- → Where UML fits in (next lecture)



Source: xkcd 844

Project Types

Reasons for initiating a software development project

- Problem-driven: competition, crisis,…
- Change-driven: new needs, growth, change in business or environment,…
- Opportunity-driven: exploit a new technology,…
- Legacy-driven: part of a previous plan, unfinished work, …

Relationship with Customer(s):

- Customer-specific - one customer with specific problem
  - May be another company, with contractual arrangement
  - May be a division within the same company
- Market-based - system to be sold to a general market
  - In some cases the product must generate customers
  - Marketing team may act as substitute customer
- Community-based - intended as a general benefit to some community
  - E.g. open source tools, tools for scientific research
  - Usually: funder ≠ customer (if funder has no stake in the outcome)
- Hybrid (a mix of the above)

Project Context

What is the current (old) system?

There is *always* an existing system!

May just be a set of ad hoc workarounds for the problem

Studying it is important:

If we want to avoid the weaknesses of the old system…

…while preserving what the stakeholders like about it

Use pre-existing components?

Benefits:

Can dramatically reduce development cost

Easier to decompose the problem if some sub-problems are already solved

Tension:

Solving the real problem vs. solving a known problem (with ready solution)

Will it be part of a product family?

Vertical families: e.g. 'basic', 'deluxe' and 'pro' versions of a system

Horizontal families: similar systems used in related domains

Typically based on a common architecture (or just shared software assets)

Lifecycle of an Engineering Project

Lifecycle models

- Useful for comparing projects in general terms
- Not enough detail for project planning
- Examples:
  - Sequential models: Waterfall, V model
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral

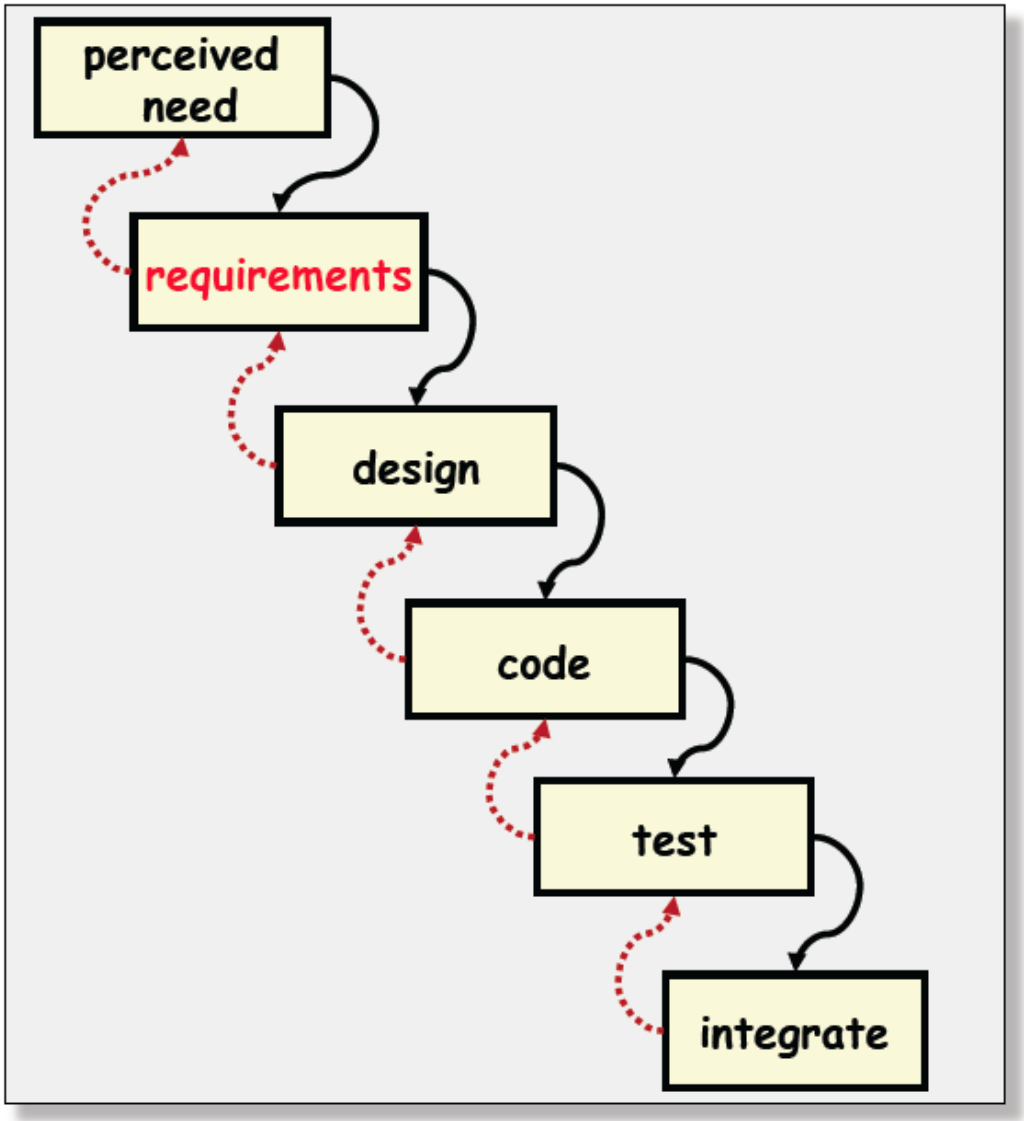
Process Models

- Used for capturing and improving the development process
- Detailed guidance on steps and products of each step

Process Frameworks

- Patterns and principles for designing a specific process for your project

Waterfall Model



View of development:

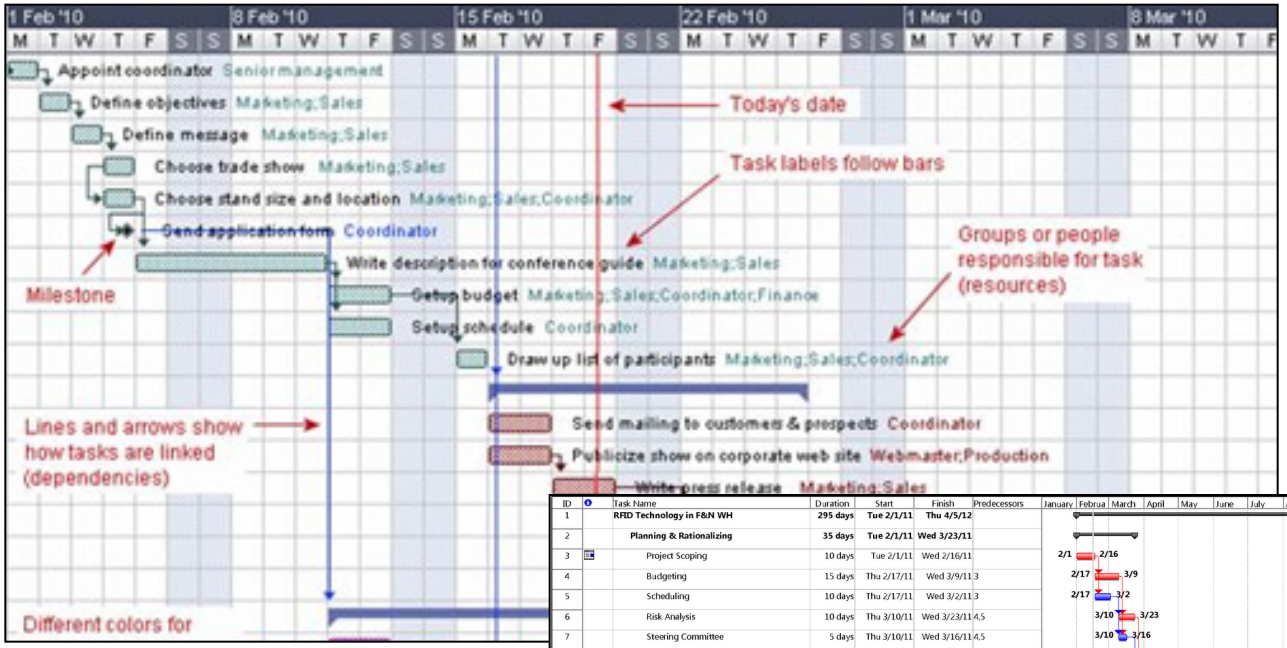
- a process of stepwise refinement
- largely a high level management view

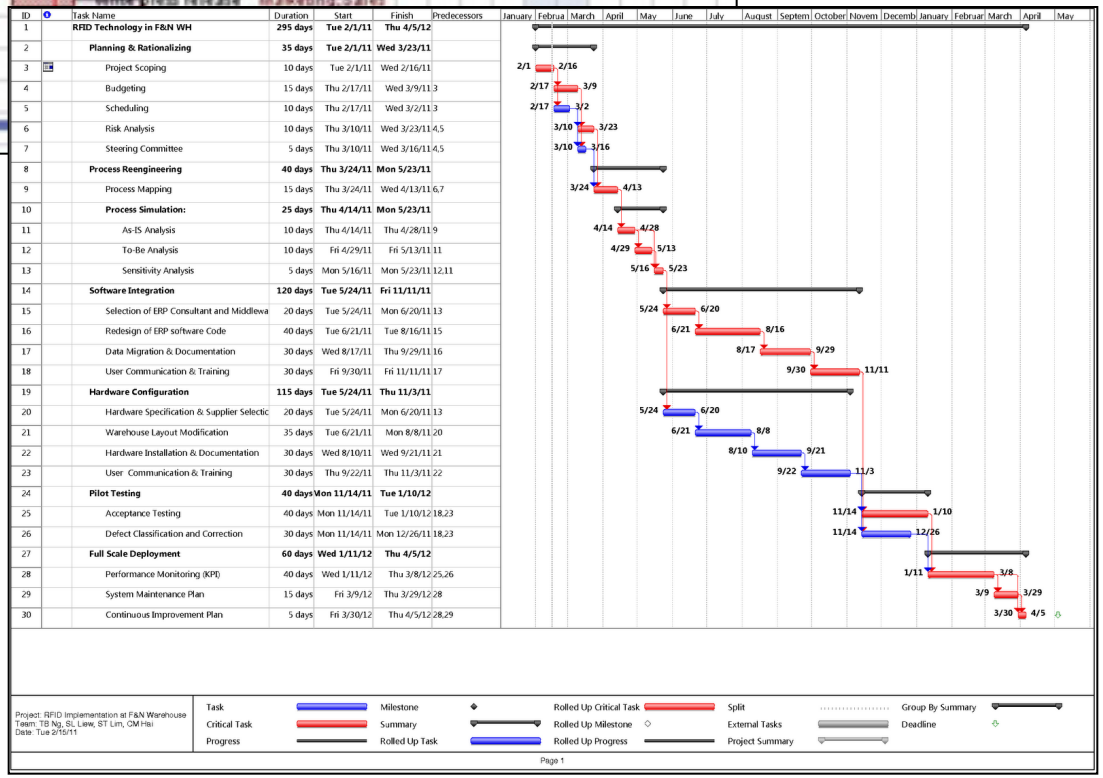
Problems:

- Static view of requirements - ignores volatility
- Lack of user involvement once specification is written
- Unrealistic separation of specification from design
- Doesn't accommodate prototyping, reuse, etc.

Computer Science
UNIVERSITY OF TORONTO

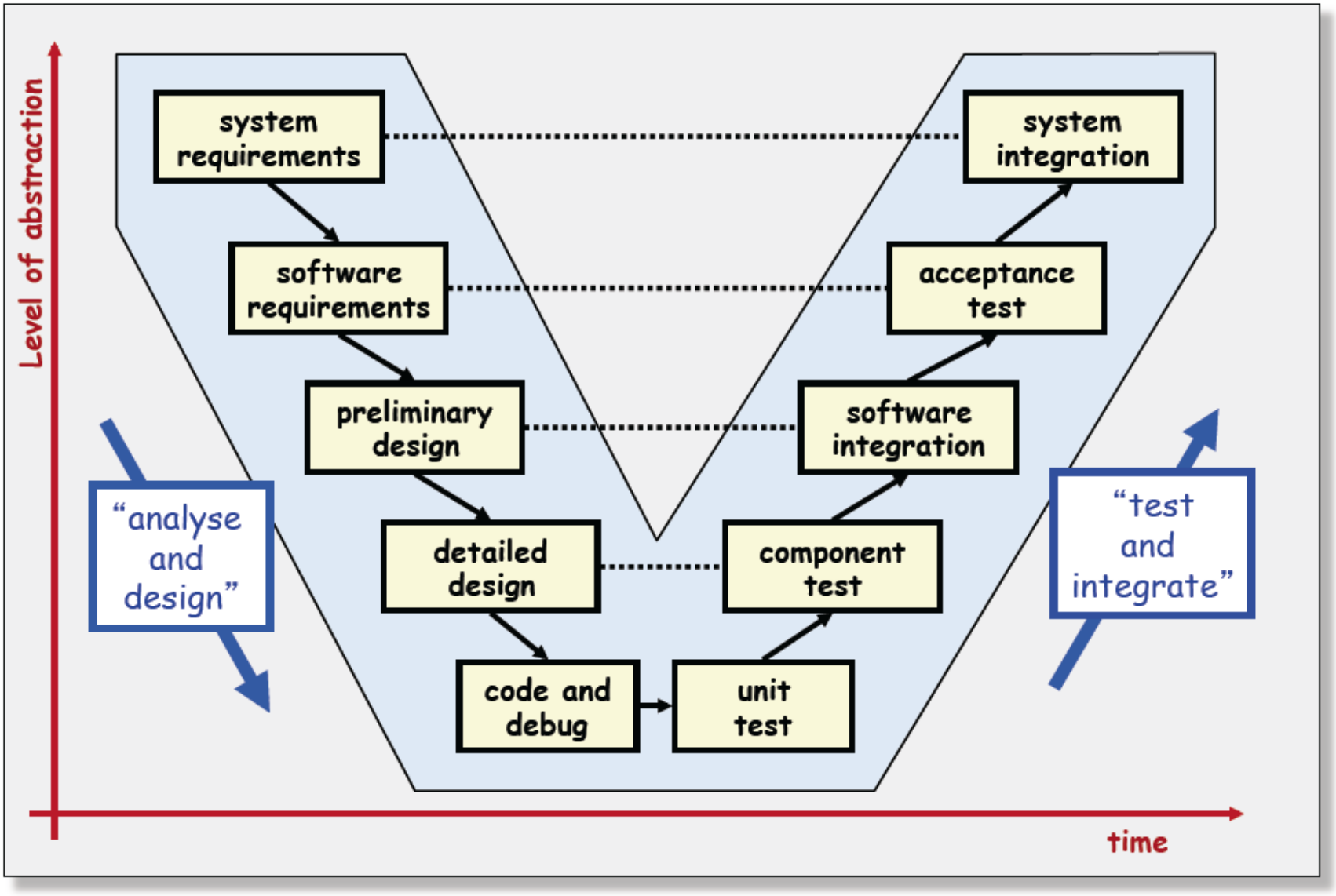
Gantt Charts & Waterfall



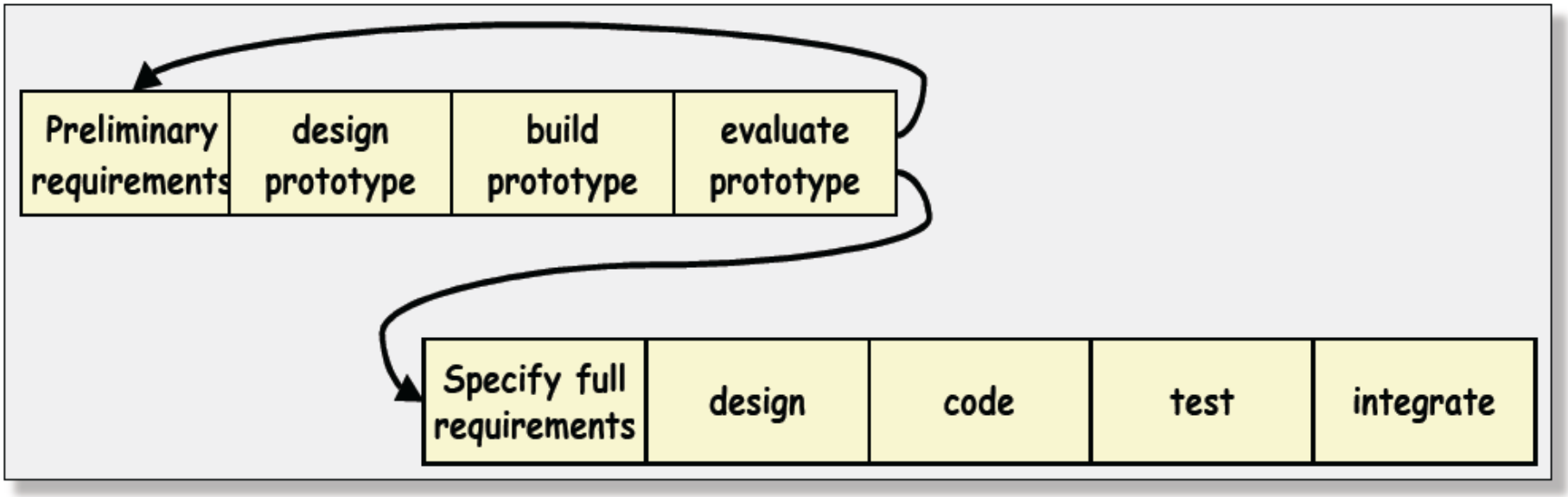


ID	Task Name	Duration	Start	Finish	Predecessors
1	**RFID Technology in F&N WH**	**295 days**	**Tue 2/1/11**	**Thu 4/5/12**	
2	**Planning & Rationalizing**	**35 days**	**Tue 2/1/11**	**Wed 3/23/11**	
3	Project Scoping	10 days	Tue 2/1/11	Wed 2/16/11	
4	Budgeting	15 days	Thu 2/17/11	Wed 3/9/11	3
5	Scheduling	10 days	Thu 2/17/11	Wed 3/2/11	3
6	Risk Analysis	10 days	Thu 3/10/11	Wed 3/23/11	4,5
7	Steering Committee	5 days	Thu 3/10/11	Wed 3/16/11	4,5
8	**Process Reengineering**	**40 days**	**Thu 3/24/11**	**Mon 5/23/11**	
9	Process Mapping	15 days	Thu 3/24/11	Wed 4/13/11	6,7
10	**Process Simulation:**	**25 days**	**Thu 4/14/11**	**Mon 5/23/11**	
11	As-IS Analysis	10 days	Thu 4/14/11	Thu 4/28/11	9
12	To-Be Analysis	10 days	Fri 4/29/11	Fri 5/13/11	11
13	Sensitivity Analysis	5 days	Mon 5/16/11	Mon 5/23/11	12,11
14	**Software Integration**	**120 days**	**Tue 5/24/11**	**Fri 11/11/11**	
15	Selection of ERP Consultant and Middleva	20 days	Tue 5/24/11	Mon 6/20/11	13
16	Redesign of ERP software Code	40 days	Tue 6/21/11	Tue 8/16/11	15
17	Data Migration & Documentation	30 days	Wed 8/17/11	Thu 9/29/11	16
18	User Communication & Training	30 days	Fri 9/30/11	Fri 11/11/11	17
19	**Hardware Configuration**	**115 days**	**Tue 5/24/11**	**Thu 11/3/11**	
20	Hardware Specification & Supplier Selectic	20 days	Tue 5/24/11	Mon 6/20/11	13
21	Warehouse Layout Modification	35 days	Tue 6/21/11	Mon 8/8/11	20
22	Hardware Installation & Documentation	30 days	Wed 8/10/11	Wed 9/21/11	21
23	User Communication & Training	30 days	Thu 9/22/11	Thu 11/3/11	22
24	**Pilot Testing**	**40 days**	**Mon 11/14/11**	**Tue 1/10/12**	
25	Acceptance Testing	40 days	Mon 11/14/11	Tue 1/10/12	18,23
26	Defect Classification and Correction	30 days	Mon 11/14/11	Mon 12/26/11	18,23
27	**Full Scale Deployment**	**60 days**	**Wed 1/11/12**	**Thu 4/5/12**	
28	Performance Monitoring (KPI)	40 days	Wed 1/11/12	Thu 3/8/12	25,26
29	System Maintenance Plan	15 days	Fri 3/9/12	Thu 3/29/12	28
30	Continuous Improvement Plan	5 days	Fri 3/30/12	Thu 4/5/12	28,29

V-Model



Prototyping lifecycle



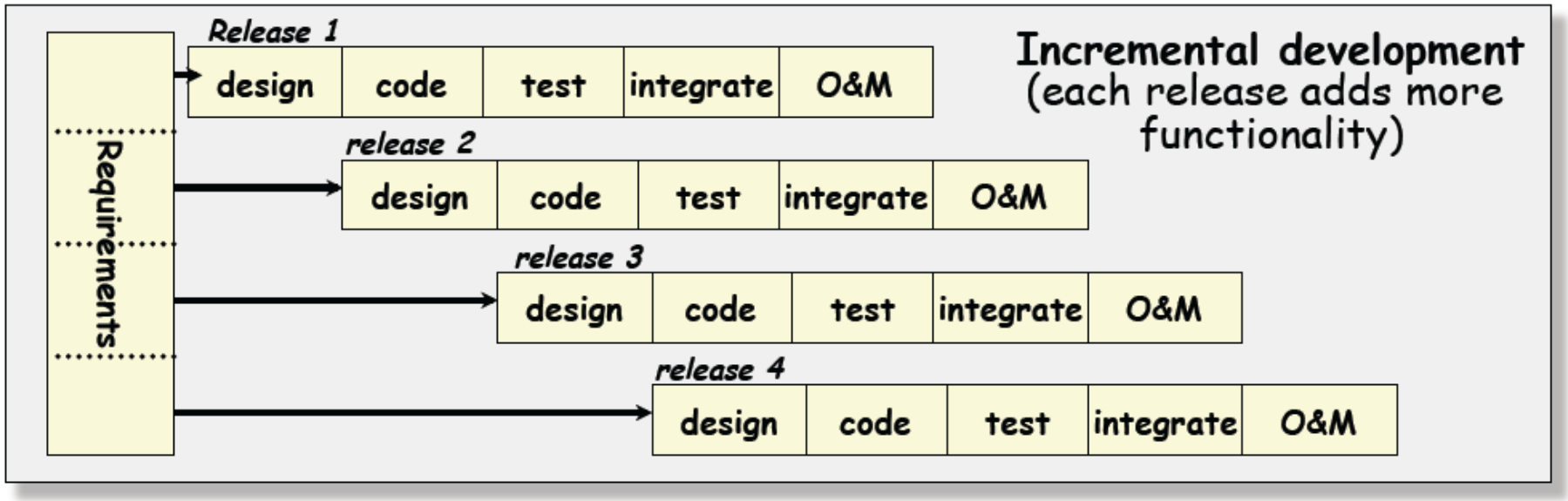
Prototyping is used for:

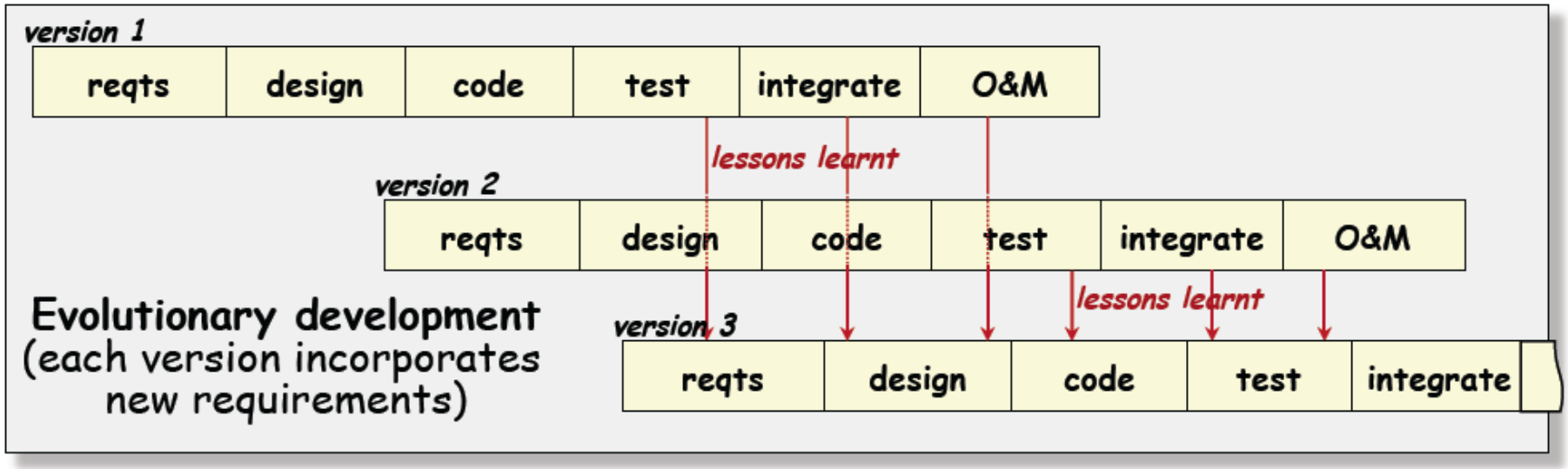
- understanding the requirements for the user interface
- examining feasibility of a proposed design approach
- exploring system performance issues

Problems:

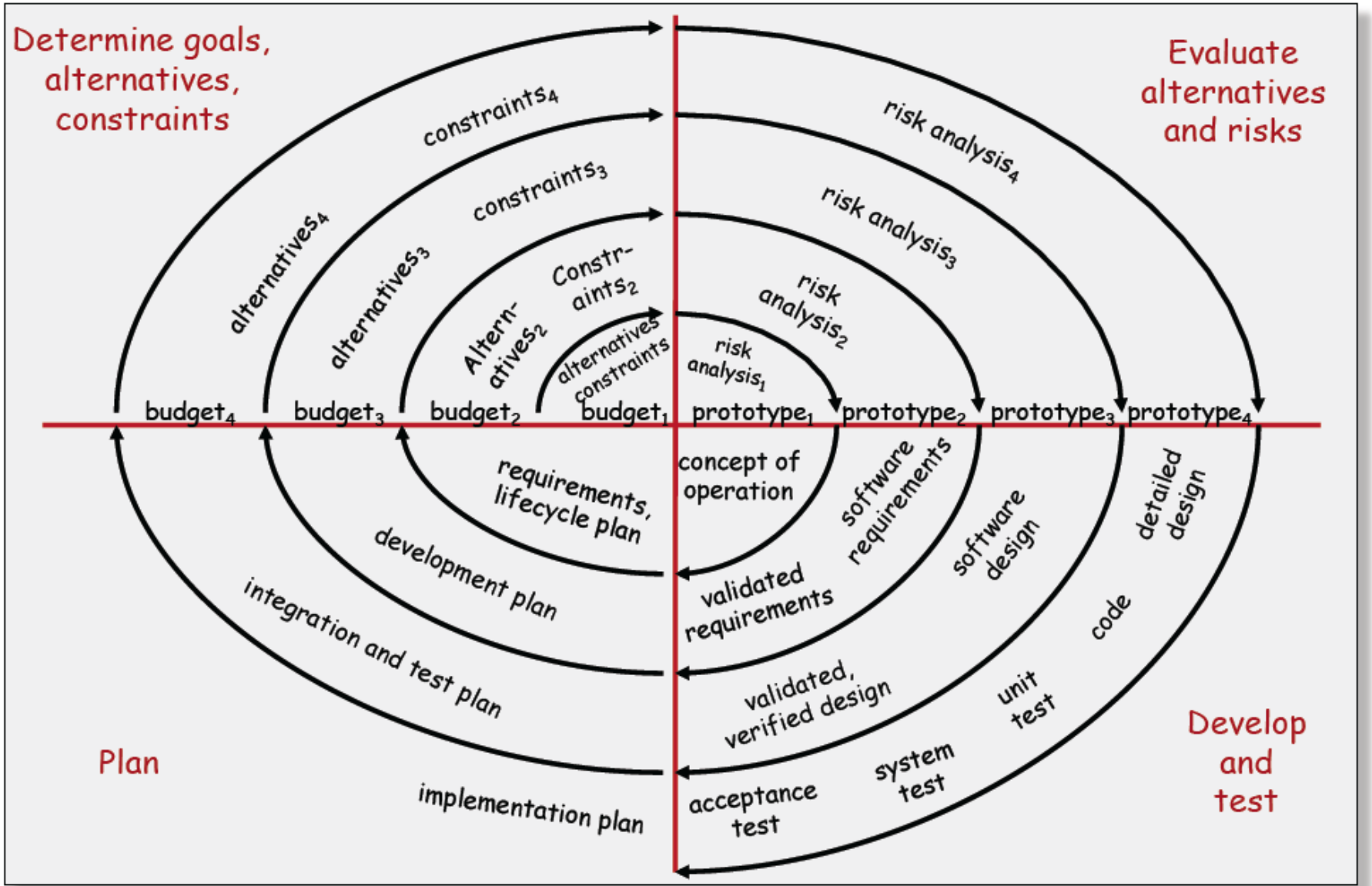
- users treat the prototype as the solution
- a prototype is only a partial specification

Phased Lifecycle Models





The Spiral Model



goal of sdlc

- what's the goal of a good sdlc?
  - passes all the tests (external quality attributes)
  - good design/architecture (internal)
  - good user experience (quality in use)
  - process quality (can process help ensure product quality)

"Agile" vs "Sturdy"

Iterative	⟷	Planned
Small increments	⟷	Analysis before design
Adaptive planning	⟷	Prescriptive planning
Embrace change	⟷	Control change
Innovation and exploration	⟷	High ceremony
Trendy	⟷	Traditional
Highly fluid	⟷	Upfront design / architecture
Feedback driven	⟷	Negotiated requirements
Individuals and Interactions	⟷	Processes and Tools
Human communication	⟷	Documentation
Small teams	⟷	Large teams

Rational Unified Process (RUP)

Inception

- Establish Scope
- Build a business case
- Get stakeholder buy-in

Elaboration

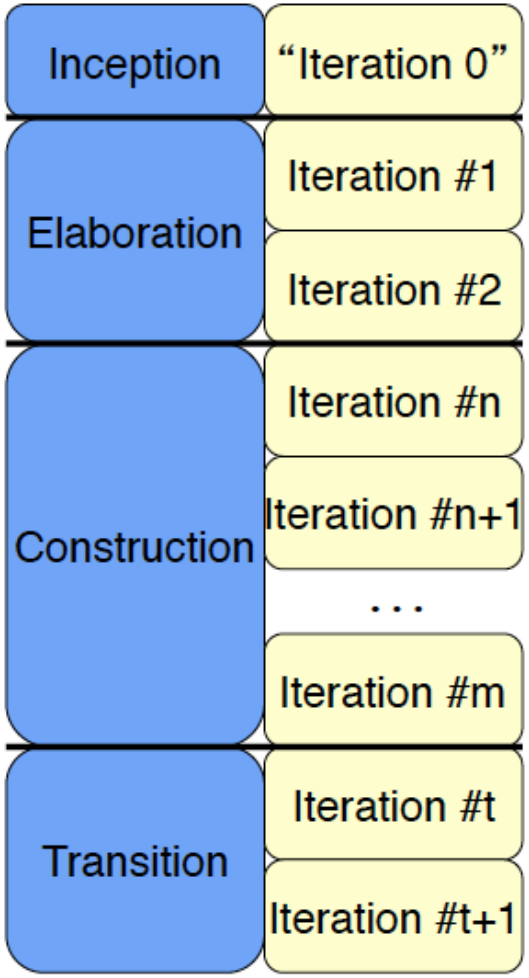
- Identify and manage risks
- Build an executable architecture
- Focus only on high risk items

Construction

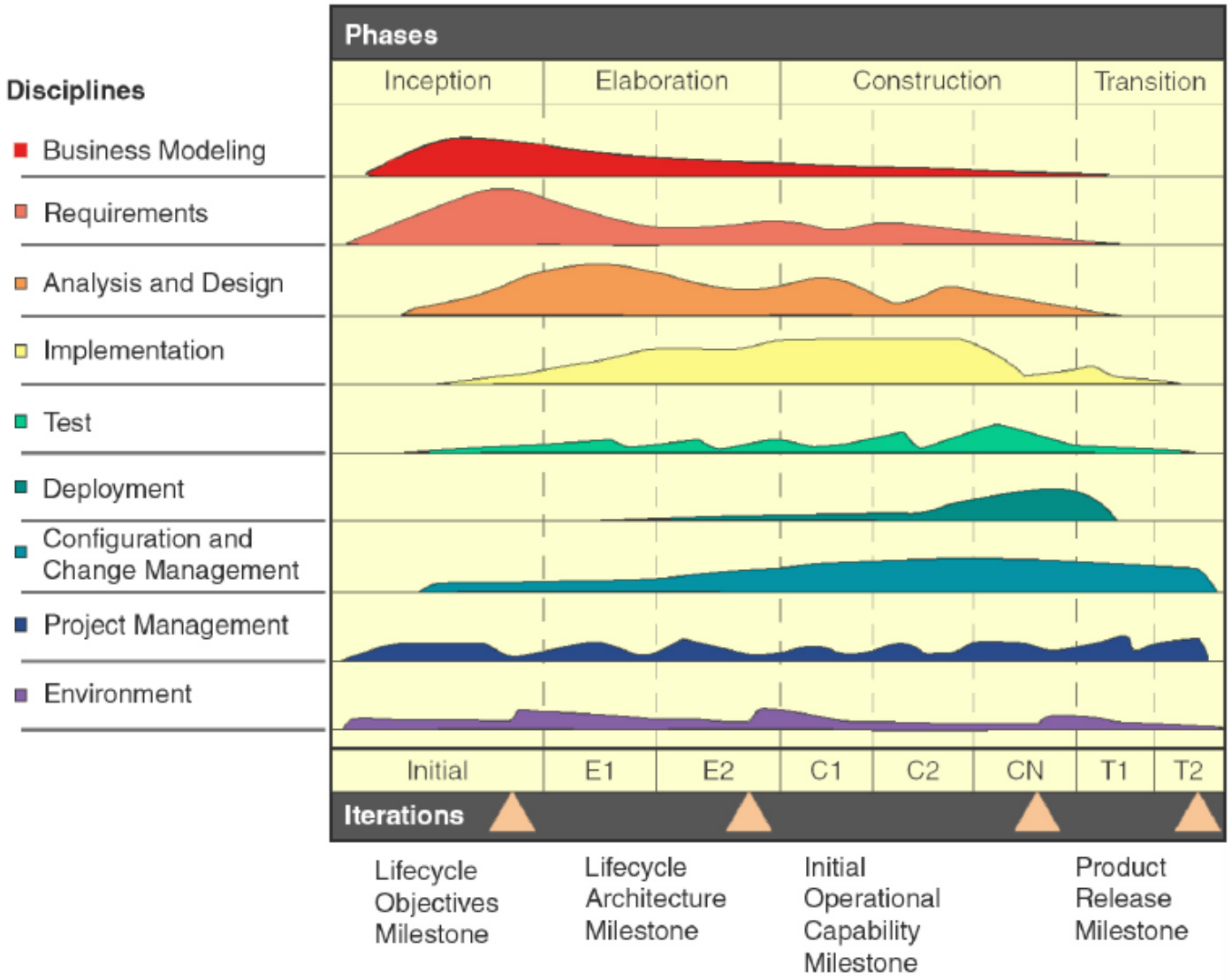
- Iteratively build operational version
- Develop support docs and training materials

Transition

- Fine-tune
- Resolve configuration, installation and usability issues



RUP Activities



SCRUM

Sprint - 30 day iteration

- Starts with 1/2 day planning meeting
- Starts with Prioritized Product Backlog (from product owner)
- Builds a Sprint Backlog - items to be done in this sprint
- 29 days of development
- 1/2 day Sprint review meeting - inspect product, capture lessons learnt

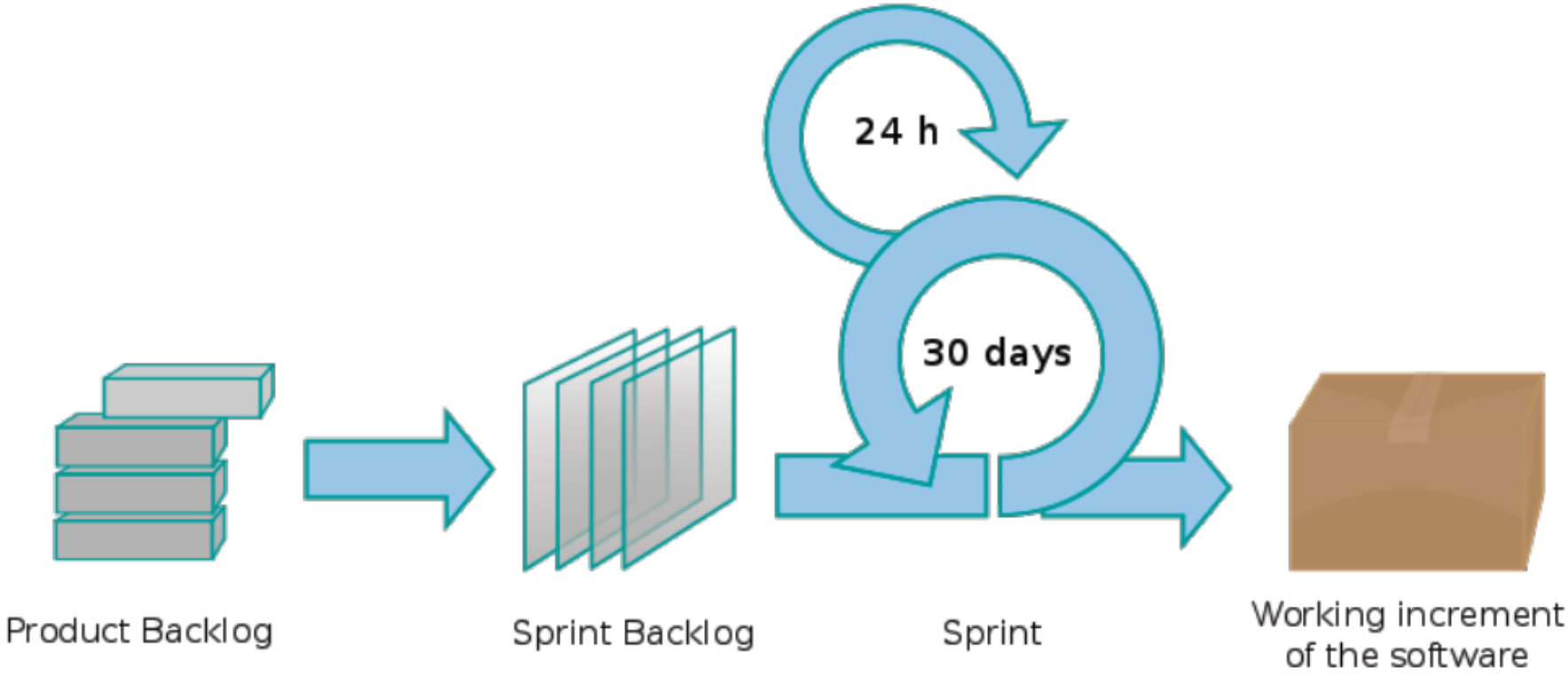
Daily Scrum

- 15 minute team meeting each day.
- Each team member answers:
  - What have you done since last meeting?
  - What will you do between now and the next meeting?
  - What obstacles stood in the way of doing work?
- Scrum master keeps meeting on track

Scrum teams

- Cross-functional, 7 (±2) members
- Teams are self-organising

Scrum Process



Source: wikipedia

Extreme Programming

Fine Scale Feedback

- Pair Programming
- Planning Game
- Test-driven Development
- Whole team (customer part of team)

Continuous Process

- Continuous Integration
- Design Improvement (refactoring)
- Small Releases

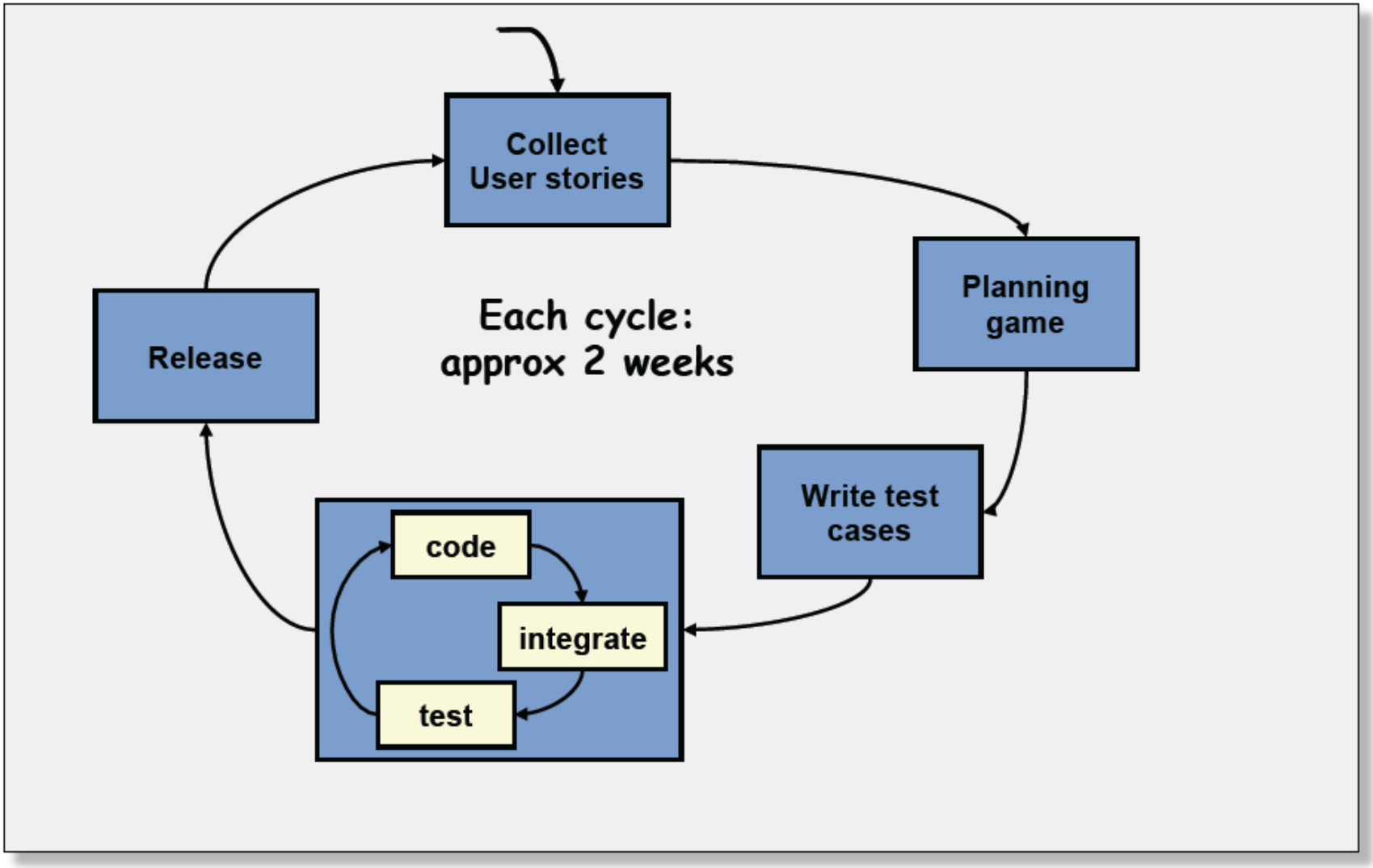
Shared Understanding

- Coding Standards
- Collective Code Ownership
- Simple Design
- System Metaphor

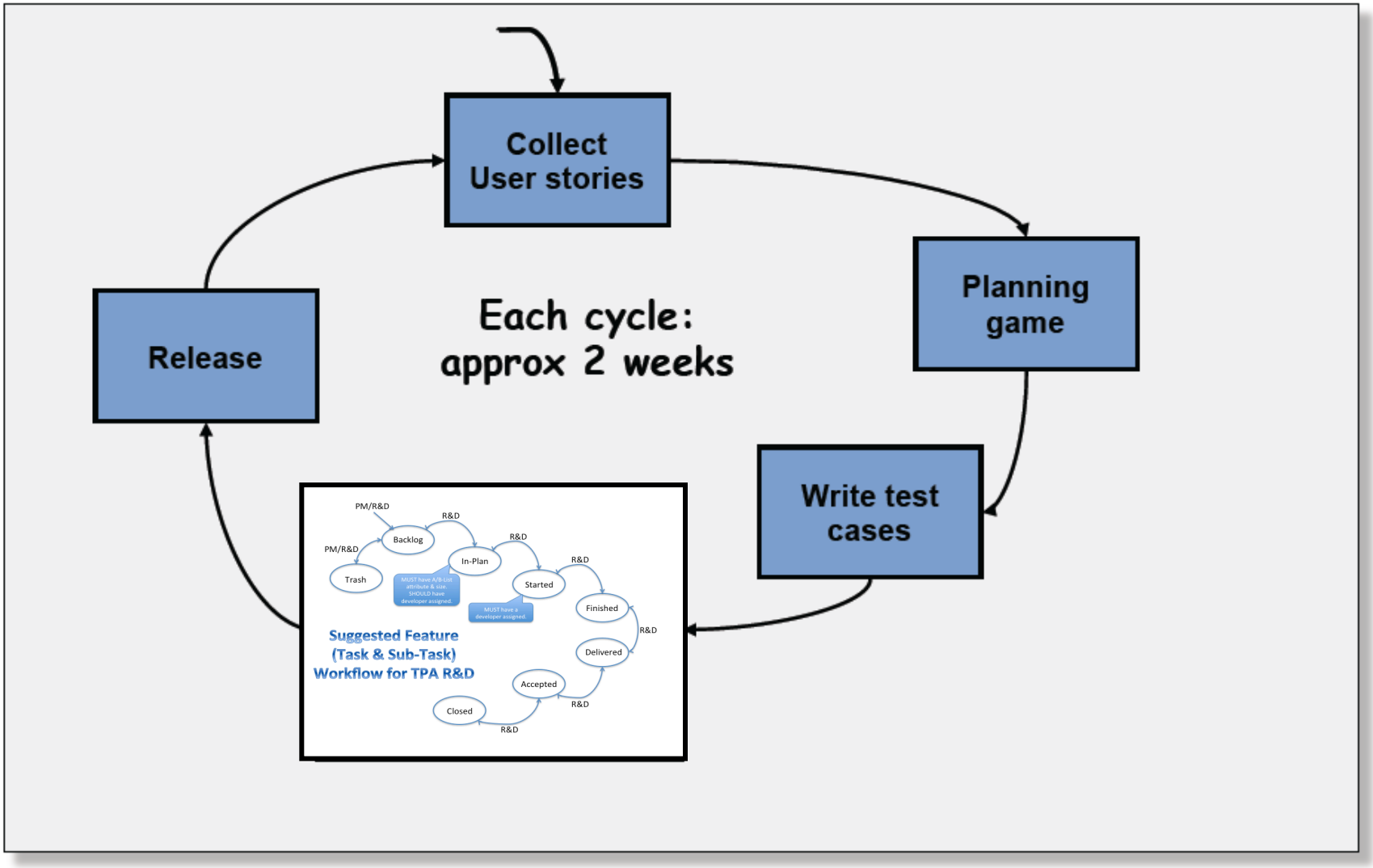
Programmer Welfare

- Sustainable pace (40 hour week)

Extreme Programming



Extreme Programming



Agile practices

- Collective Ownership
- Configuration Management
- Continuous Integration
- Feature-driven devl.
- Frequent small releases
- Onsite customer
- Organization-wide process
- Organizational training
- Pair programming
- Planning game
- Peer reviews
- Process & product quality assurance
- Project monitoring & control
- Project planning
- Refactoring
- Requirements management
- Retrospective
- Risk Management
- Simple design
- Tacit knowledge
- Test-driven development

which process is the best?

- all processes have their pros and cons, but only in the context of a given project.
  - does continuous deployment make sense for the next version of microsoft office?
  - what process is best for an x-ray machine?
  - Space Shuttle avionics – hal/s developed specifically for shuttle
    - completely independently developed primary and backup systems!
  - curiosity rover software, installed in flight! and then upgraded on mars!
- again, depends on the nature of the project

break,
then short tutorial