Course Introduction, Software Projects

Marek Rychlý
rychly@fit.vutbr.cz

Brno University of Technology
Faculty of Information Technology
Department of Information Systems

Information Systems Analysis and Design (AIS)
26 September 2019

Outline

Course Organisation

- Thursday, 10:00–12:50, room D0206
- 13 lectures on Software Engineering topics

1^{st}	Course Introduction, Software Projects
2^{nd}	Software Development Processes
3^{rd} – 4^{th}	Software Modelling in UML
5^{th} – 6^{th}	Unified Process
7^{th} – 8^{th}	Software Architecture
9^{th} – 10^{th}	Object-Oriented Design Principles and Patterns
11^{th} – 12^{th}	TDD, Refactoring, Anti-Patterns, etc.
13^{th}	Practice/Examples on Software Analysis and Design

- a project on software modelling
  (teams of two to four members, initial and final models, defences, . . .)
- a midterm exam
  (7^{th} lecture, November 7, 2019)
- a final exam
  (January/February)

The Project

- teams of two to four members, each team selects the project topic
  (the teams can be formed before/during/after the lectures or via WIS forums)
- the project results submitted in two subsequent parts
  1. Project Plan
     (detailed project specification and planned iterations)
  2. First Iteration Models and Final Models
     (UML models for the first and the final iteration)
- the topics (software projects) are described briefly in WIS
  (a selected topic should be described in details in the first project part)
- the most of software model diagrams are in the second part

Software Models Required in the Project

- use case diagrams, one for each iteration
  (in the first part of the project, as well as in the second part)
- a structured description of use cases including exceptions
- a software architecture design
- a system sequence diagram
- a domain model
- a structured description of class/object responsibilities
- a design model class diagram
- packages, their coupling, and layers
- UML interaction diagrams
- screen transitions
- a structured description of a test

Evaluation

- each student can earn 100 points
  - 5 points the first part of the project (submitted)
  - 15 points the midterm exam
  - 29 points the second/final part of the project (submitted and defended)
  - 51 points the final exam (the required minimum is 20 points for the final exam)
- to go to the final exam, a student must
  - have submitted and defended his/her project
  - earn 24 points at least before the final exam

Deadlines

2019-10-06	ask for a recognition of points earned in the project last year
2019-10-23	create the project teams of two to four members
	register the project team in WIS
	select and register a project topic for the team in WIS
2019-10-24	submit the first part of the project into WIS
2019-11-07	pass the midterm exam
2019-12-08	select a date of the project defence in WIS
2019-12-09	submit the second/final part of the project into WIS
2019-12-10	(and later) defend the project on the selected date

Core Reading Materials

Jim Arlow, Ila Neustadt, and Bogdan Kiszka.

UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.
Computer Press, 2007.
ISBN 978-80-251-1503-9.
URL
http://primo.lib.vutbr.cz/420BUT:Everything:420BUT_Aleph000076198.

Craig Larman.

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition).
Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
ISBN 0131489062.
URL http:
//www.craiglarman.com/wiki/?title=Book_Applying_UML_and_Patterns.

Leszek Maciaszek and Bruce Lee Liong.

Practical Software Engineering: An Interactive Case-Study Approach to Information Systems Development.
Pearson Addison Wesley, 2004.
ISBN 0321204654.
URL http://www.comp.mq.edu.au/books/pse/.

Software Engineering

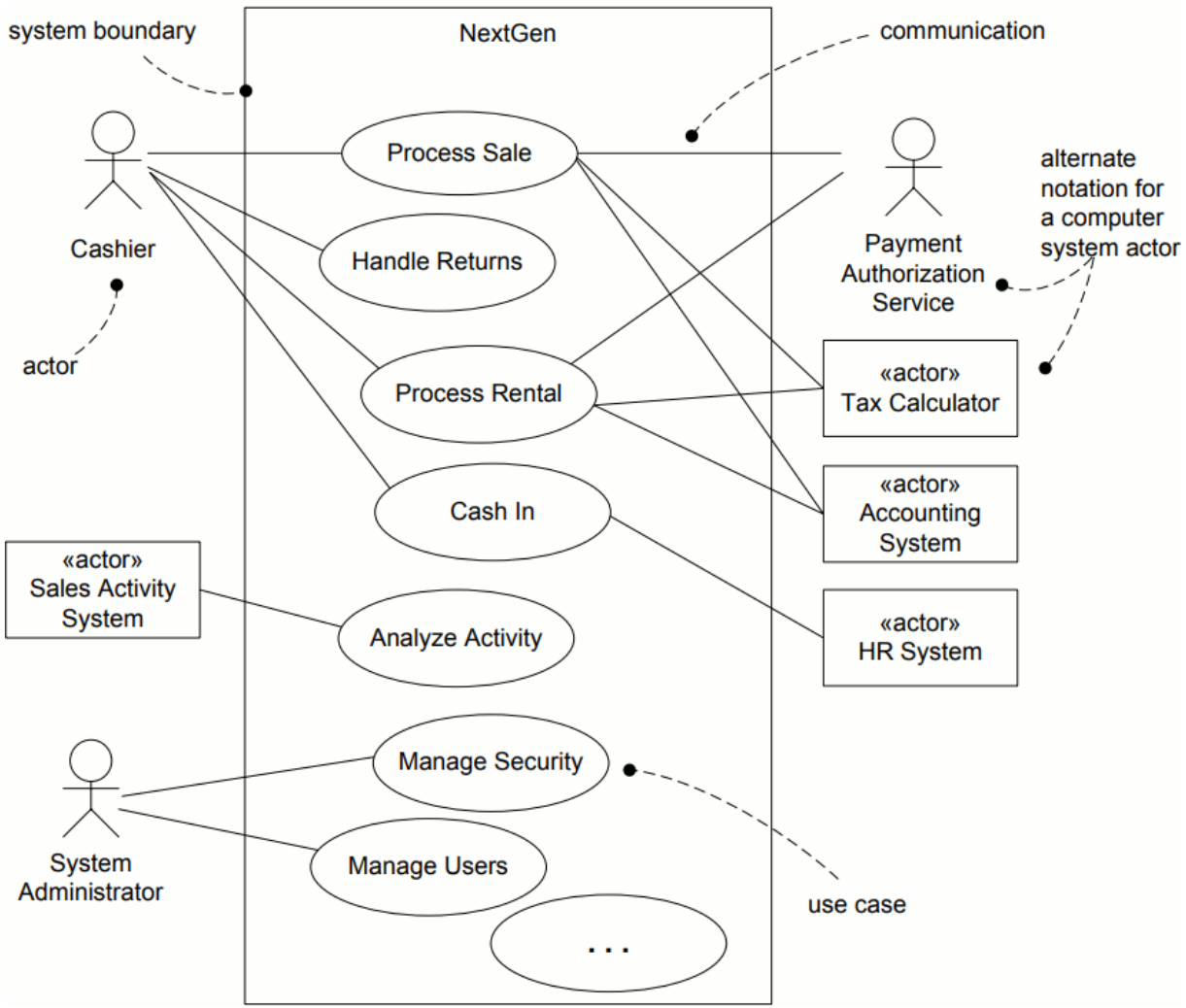
- Software Engineering is not (only) about coding.
  (it is mainly about software design for its development, maintenance, etc.)
- Software needs to be modelled before its implementation.
  (the models are products of the software design phase)
- To be able to read/understand models is not enough.
  (a software engineer must be able to create/modify such models)
- It is about big and complex software systems (information systems).
  (team projects, difficult to design and develop, exhausting/long implementation)
- One must take many difficult design decisions.

Case Study: NextGen POS

- NextGen point-of-sale (POS) system from [Larman(2004)].
  (an application used to record sales and handle payments; for retail stores)
- The NextGen POS case study will be utilized in this course.
  (to demonstrate software design techniques)

1. A use case model to define actors, usage, and a system boundary.
2. A system sequence diagram to describe a particular scenario.
3. A domain model to identify software objects and their properties/relationships.
   (not only persistent objects, contrary to conceptual models)
4. A sequence diagram to describe a particular interaction of the objects.
   (in accordance with their responsibilities)
5. A design model class diagram to define implementation classes.
   (the classes of the object able to participate in the interactions)

NextGen POS: Use Case Diagram



(adopted from [Larman(2004)])

System Sequence Diagram for “Process Sale”

Simple cash-only *Process Sale* scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.

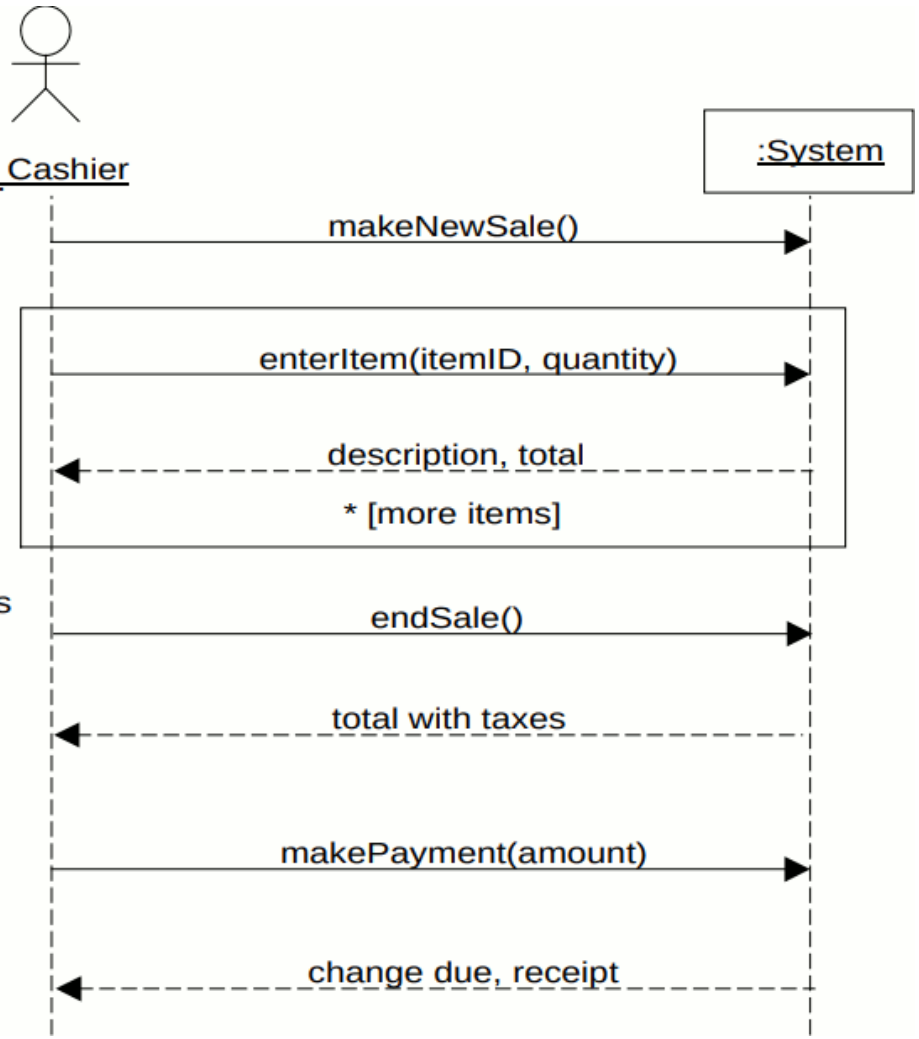
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.

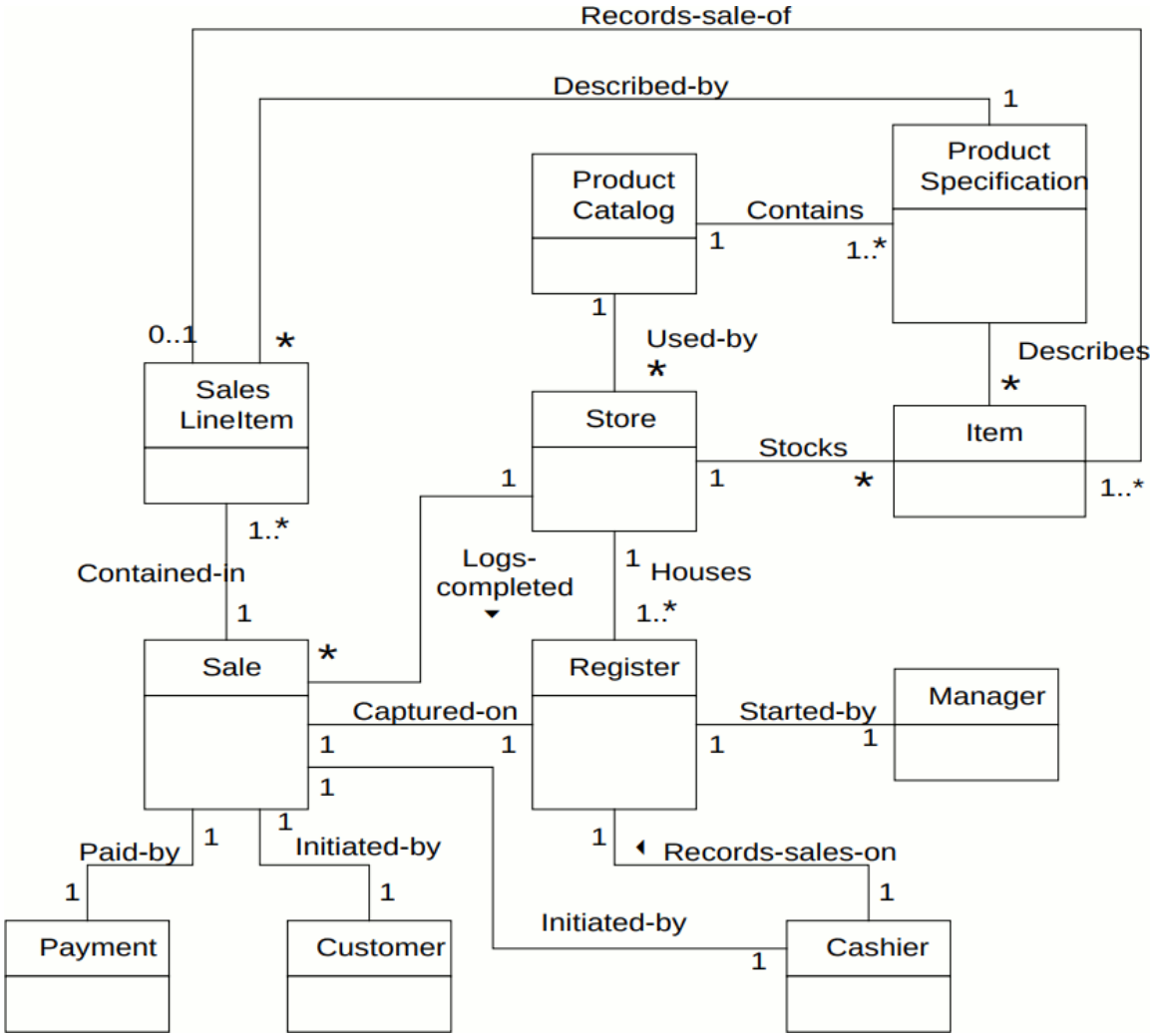
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.

...



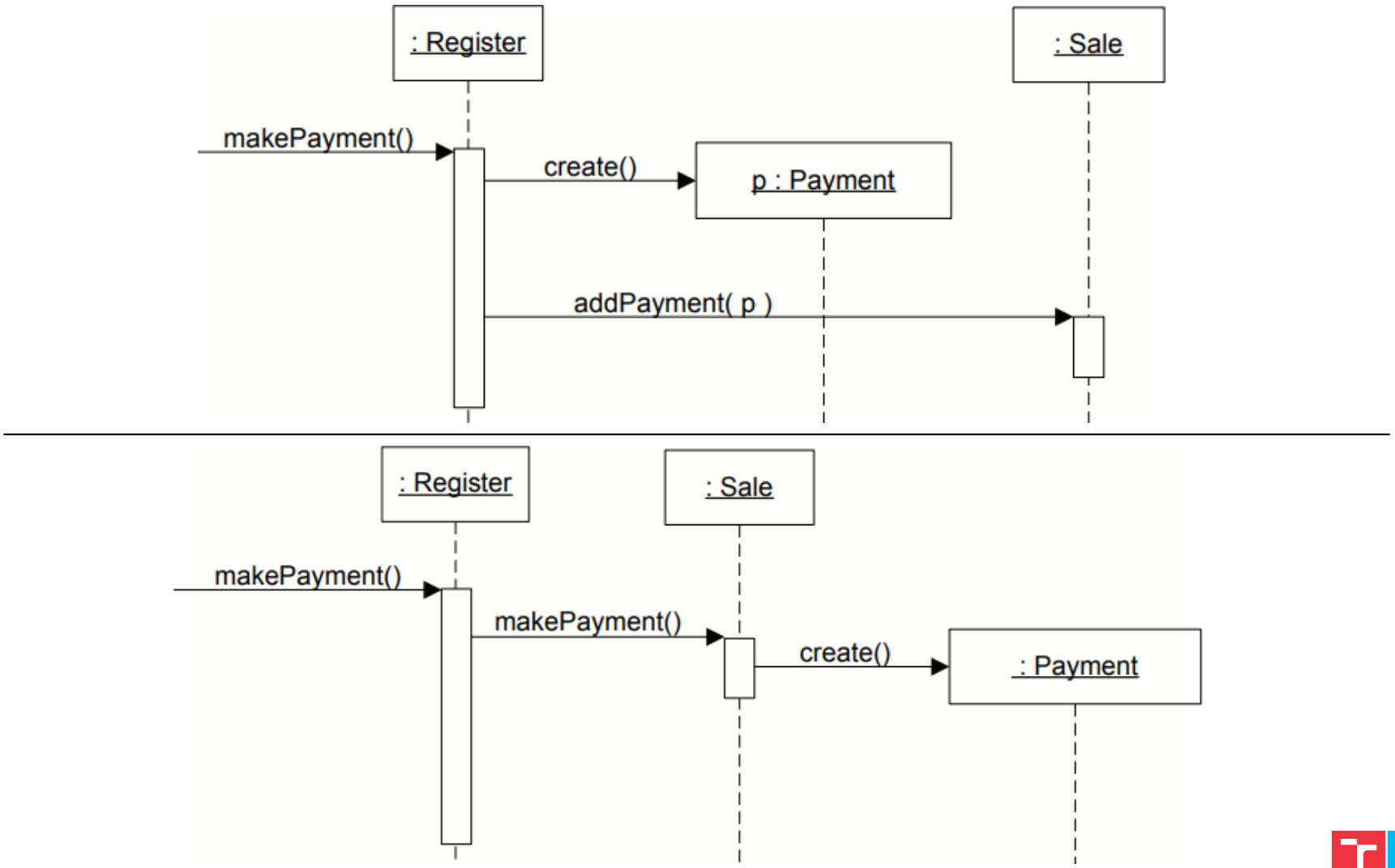
(adopted from [Larman(2004)])

Partial Domain Model



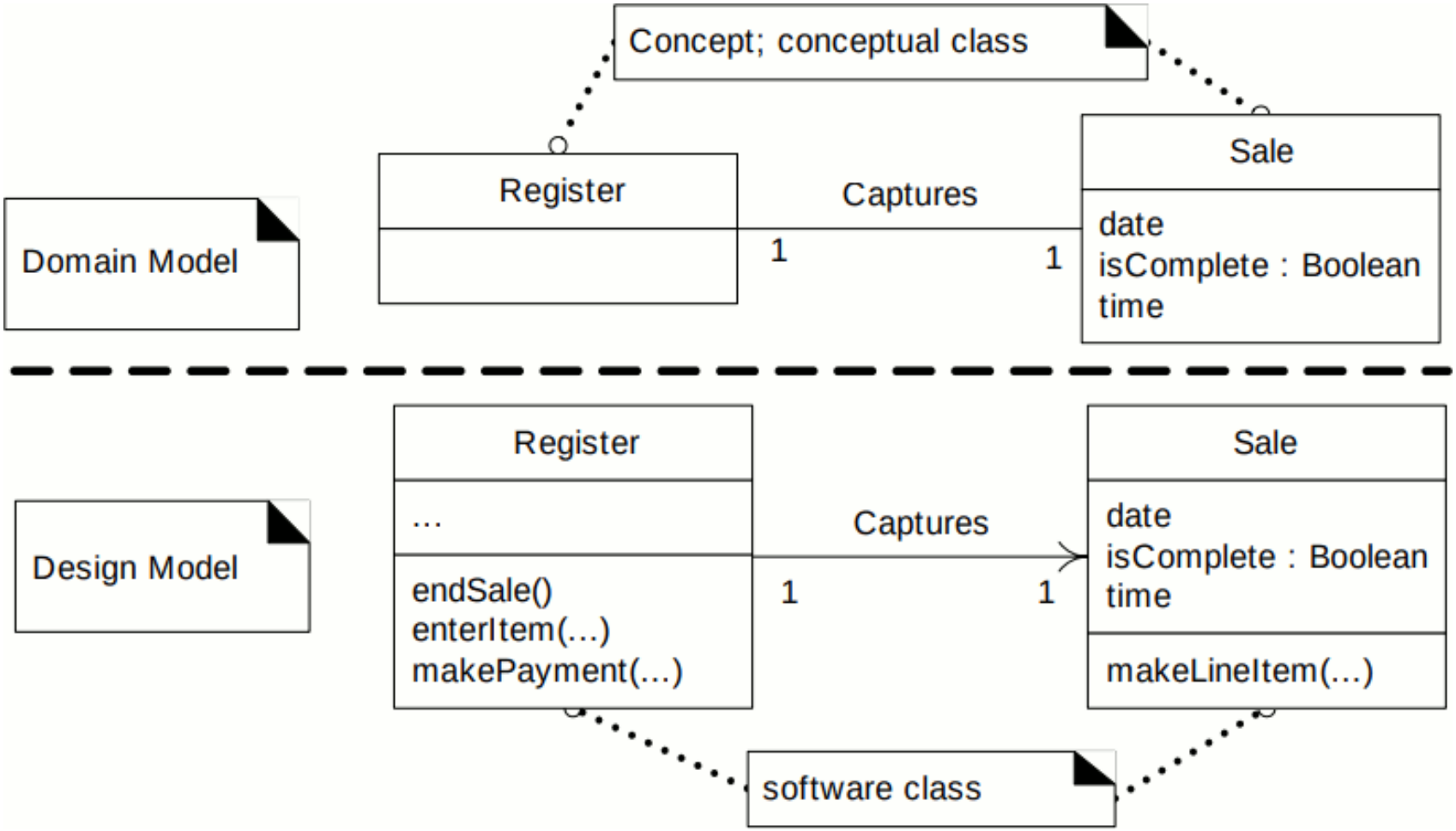
(adopted from [Larman(2004)])

Different Sequence Diagrams for “Payment” Creation



(adopted from [Larman(2004)])

Domain Model vs. Design Model Class Diagram



(adopted from [Larman(2004)])

(Enterprise) Information System

Definition (Information System by businessdictionary.com)

A combination of hardware, software, infrastructure and trained personnel organized to facilitate planning, control, coordination, and decision making in an organization.

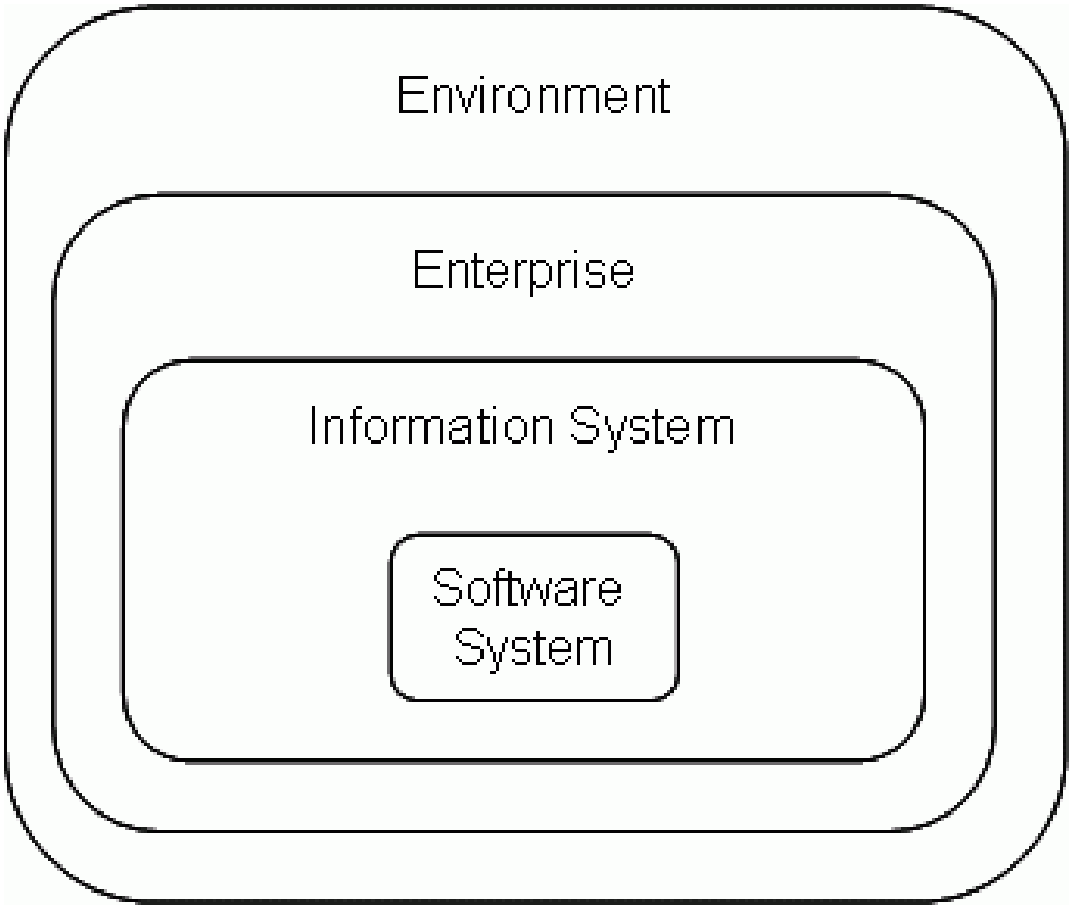
Definition (Information System by pcmag.com)

A business application in the computer. It is made up of the database, application programs and manual and machine procedures. It also encompasses the computer systems that do the processing.

Definition (Information System by Wikipedia)

An organized system for the collection, organization, storage and communication of information.

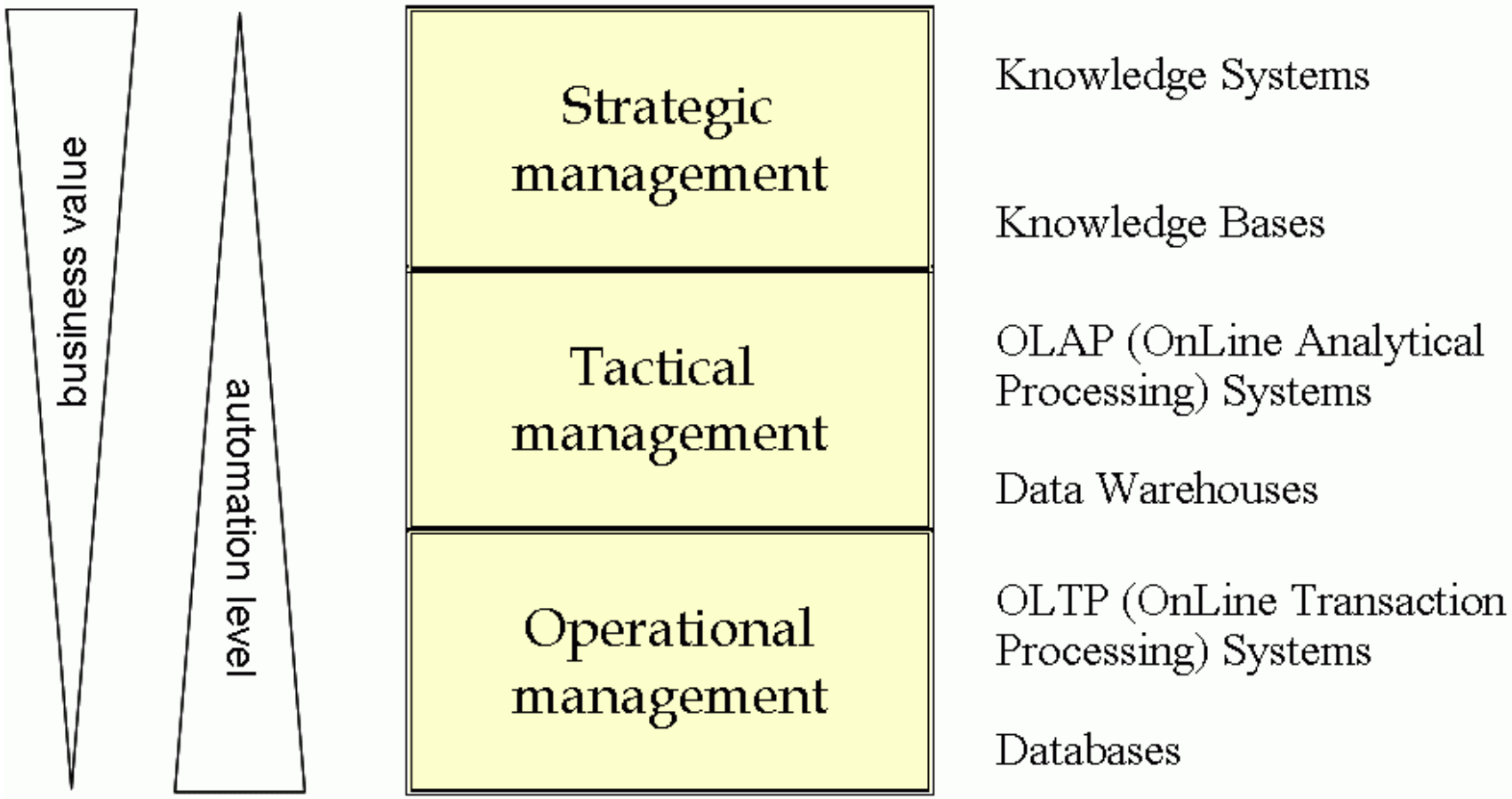
Information System and Software System



(adopted from [Maciaszek and Liong(2004)])

IT and Enterprise Governance/Management

(to support knowledge of business processes, not only to manage information)



(adopted from [Maciaszek and Liong(2004)])

Software Engineering (SE)

Definition (Software Engineering by ISO/IEC/IEEE 24765:2010)

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

- Necessary for huge and/or complex systems developed by teams.
  (however, it makes sense also for one-man software projects)
- There is a huge amount of data, a lot of constraints, both practical and given by business rules and policies.
  (the resulting software/information system must fit into an enterprise)

Software Project

Definition (Software Project by ISO/IEC/IEEE 24765:2010)

The set of (software engineering) work activities, both technical and managerial, required to satisfy the terms and conditions of a project agreement (to develop a software product or service).

- It should have specific starting and ending dates, well-defined objectives and constraints, established responsibilities, and a budget and schedule.
- A software project may be self-contained or may be part of a larger project.
- It may span only a portion of the software development cycle, as well as many years and consist of numerous subprojects, each being a well-defined and self-contained software project.

Success Rate of Software Projects

Standish Group 2015 Chaos Report

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

(adopted from Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch)

Ratios of Software Projects by their Size

Standish Group 2015 Chaos Report

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

(adopted from Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch)

Success Rates of Software Projects by Development

Standish Group 2015 Chaos Report

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Watertall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000

(adopted from Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch)

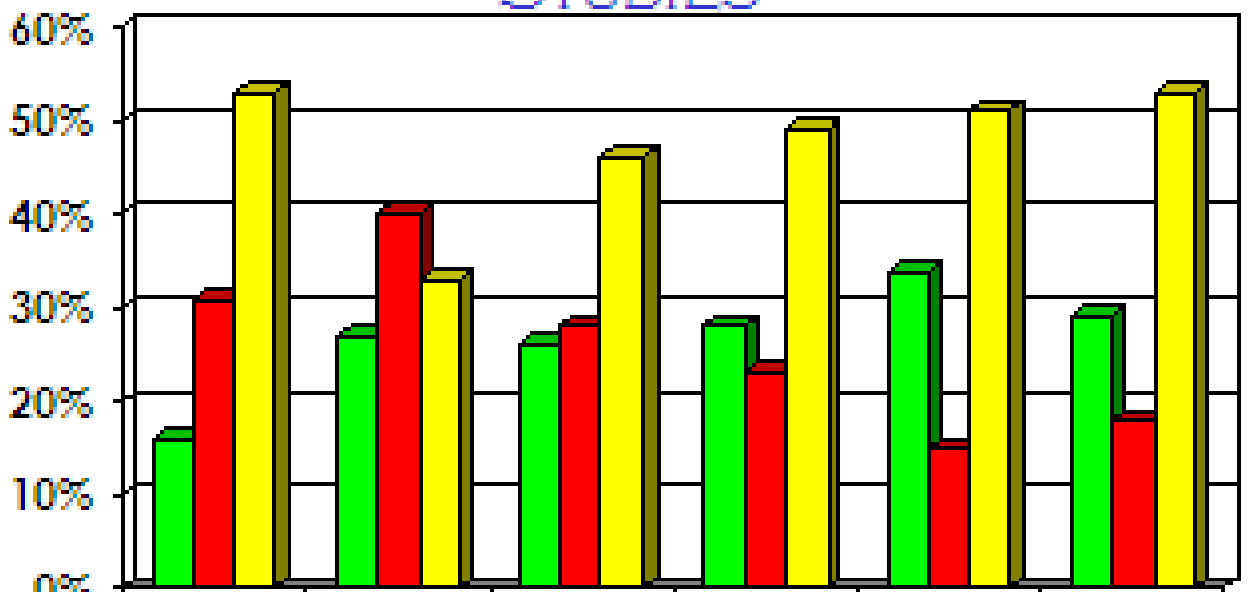
Success Rates of Historical Software Projects

Standish Group 2004 Chaos Report

THE STANDISH GROUP

CHAOS Research

STUDIES



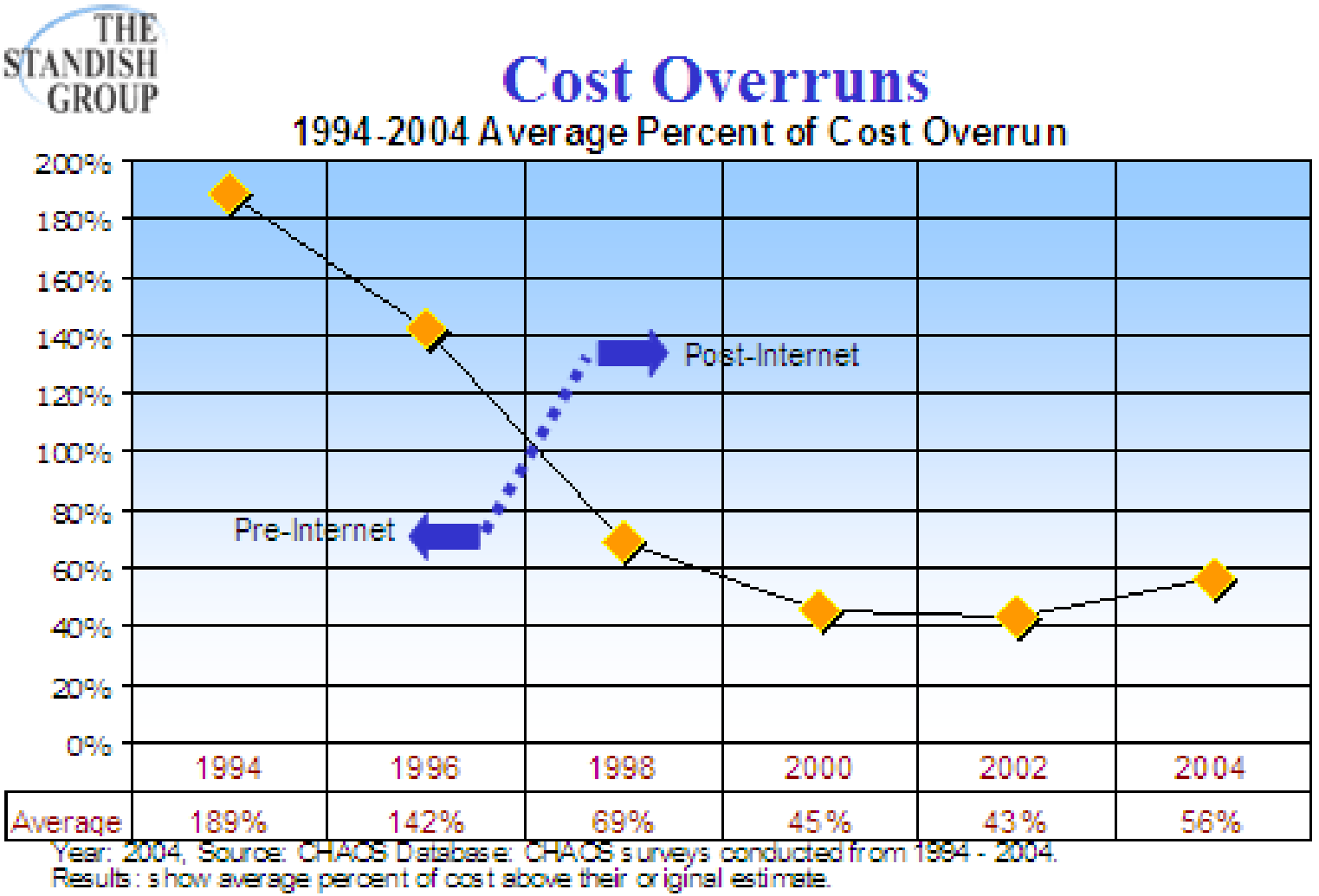
	1994	1996	1998	2000	2002	2004
Succeeded	16%	27%	26%	28%	34%	29%
Failed	31%	40%	28%	23%	15%	18%
Challenged	53%	33%	46%	49%	51%	53%

FIT

(adopted from Standish: Why were Project Failures Up and Cost Overruns Down in 1998?)

Software Projects in the Internet Era (C/S → Web)

Standish Group 2004 Chaos Report



(adopted from Standish: Why were Project Failures Up and Cost Overruns Down in 1998?)

Success Factors of Software Projects

Standish Group 2015 Chaos Report

30% Executive support, Emotional maturity
(financial/emotional backing; the maturity by skills and the weakest link)

30% Optimization, Skilled staff, Project management expertise

15% User involvement
(users involved in the project decision-making and information-gathering process)

13% Agile proficiency, Modest execution
(agile team and the product owner; a few simple tools, automated and streamlined)

8% Standard architectural management environment
(practices/services/products for developing, implementing, and operating software)

4% Clear business objectives
(the understanding of all stakeholders and participants in the business purpose)

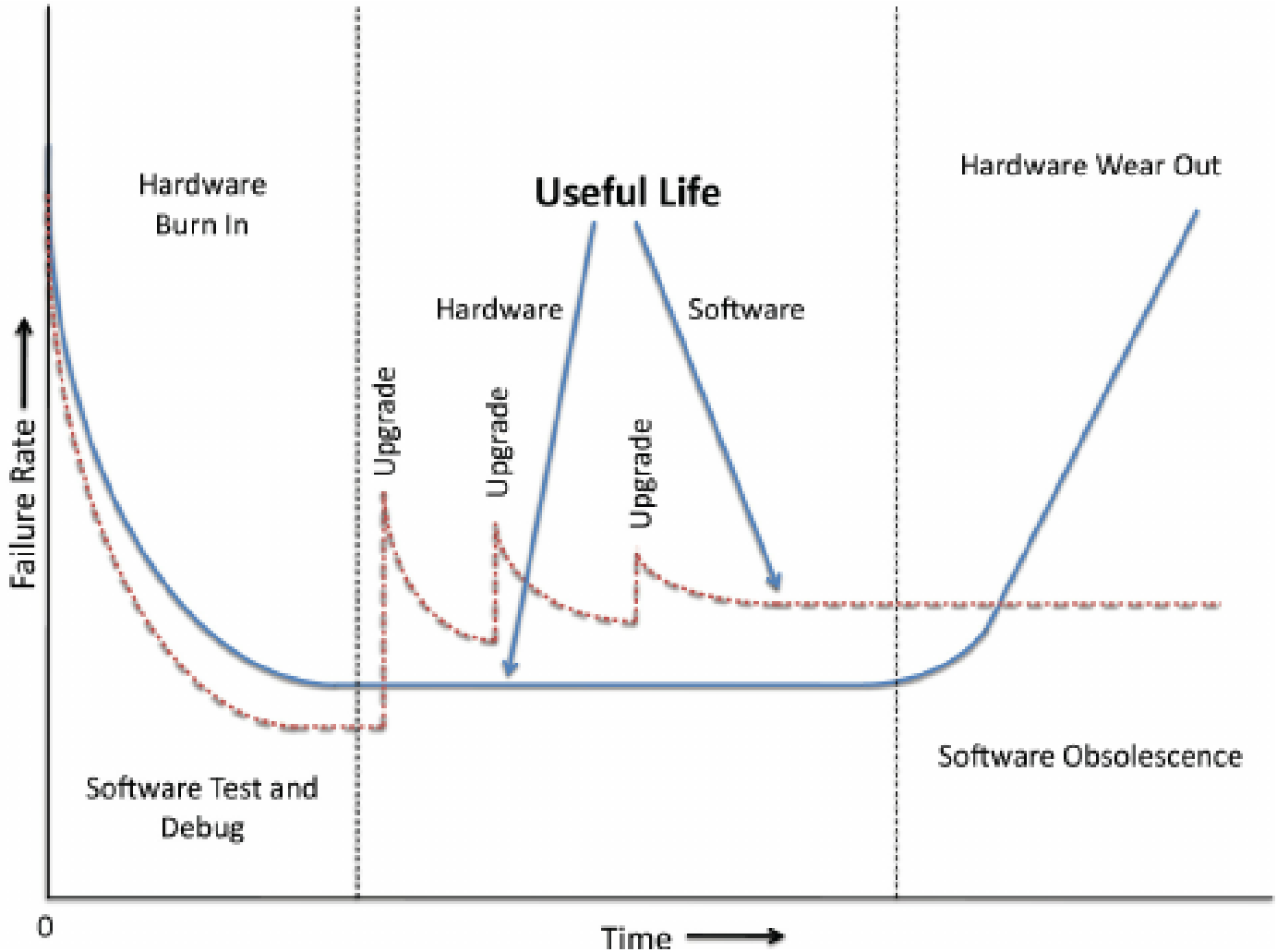
(according to Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch)

Software is a Special Product/Service

- It is abstract.
  (vague and unstable specification, limited applicability of classical engineering methods, problematic validation of the result, etc.)
- It is not manufactured but implemented.
  (there is no well-established prototype that would be copied by the manufacturing)
- It must be flexible/agile.
  (i.e., supportable = well understood/maintainable, manageable, and scalable)

Software/Hardware Failure Rate in Time

(a “bathtub curve”)



(adopted from Application monitoring and checkpointing in HPC: looking towards exascale systems)

Software Engineering is Beyond Programming/Coding

- Software requirements have to be collected and comprehended.
  (their correctness and understanding are critical for the success of a SW project)
- A complex SW must be well-designed before its implementation.
  (software failures are mostly caused by wrong design, not the implementation)
- Software must be validated and verified.
  (valid requirements and design, verified implementation)
- Software is mostly developed in teams.
  (one or more small/large teams working on a project/subprojects)
- Forward, reverse, and round-trip engineering techniques.
  (to keep the design and implementation/code synchronised)

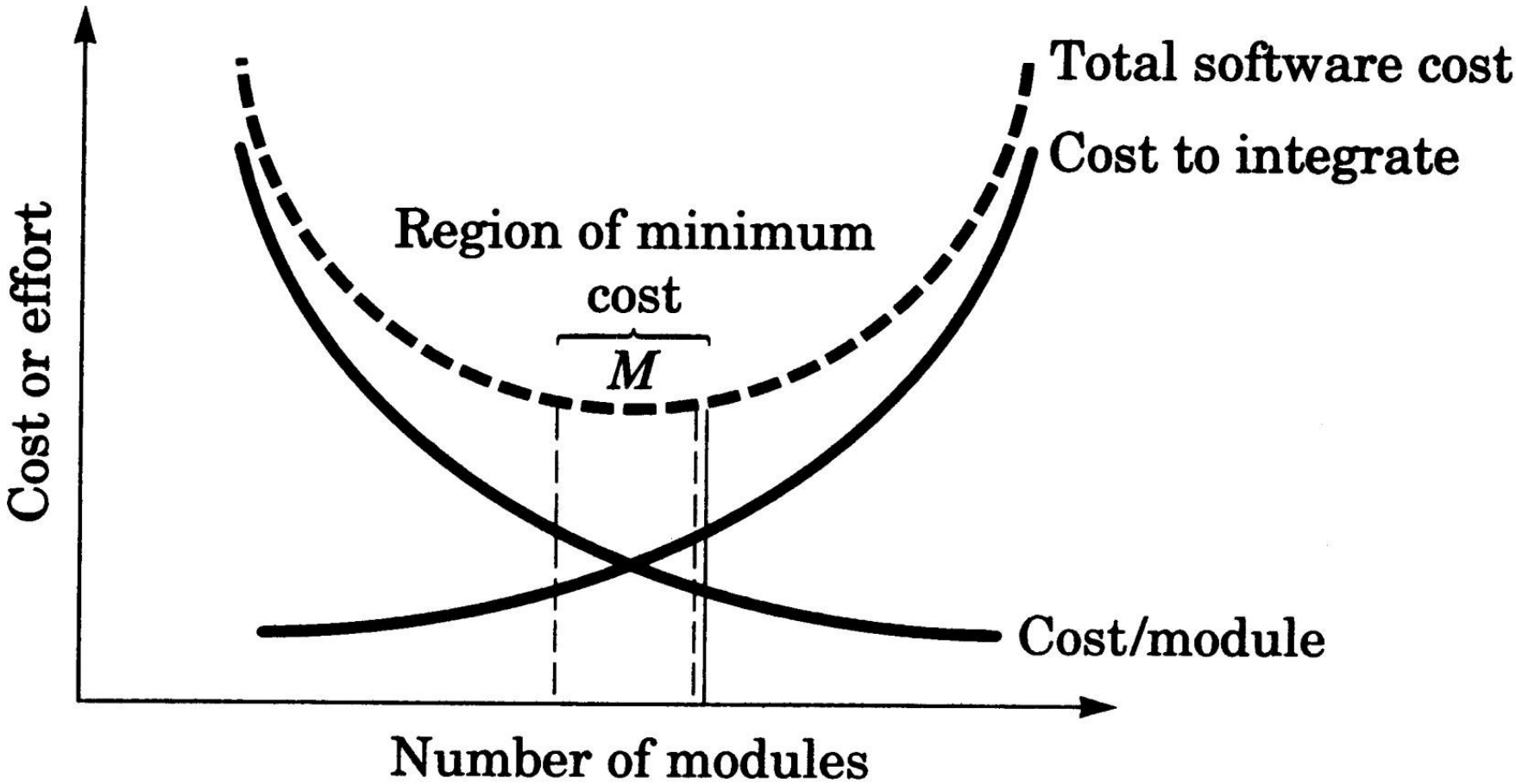
Software Engineering is about Modelling

- Models are abstractions of real-world objects.
  (simplified and generalised to support the understanding)
- Two types of models in software engineering:
  - models of the software process (for management)
  - models of the software product (for development)
- The models respect well-established paradigms.
  - functional-oriented
    (the function operate on data to transform input to output data)
  - object-oriented
    (the objects manage their data and provide actions to clients)

Software is Structured (has Architecture)

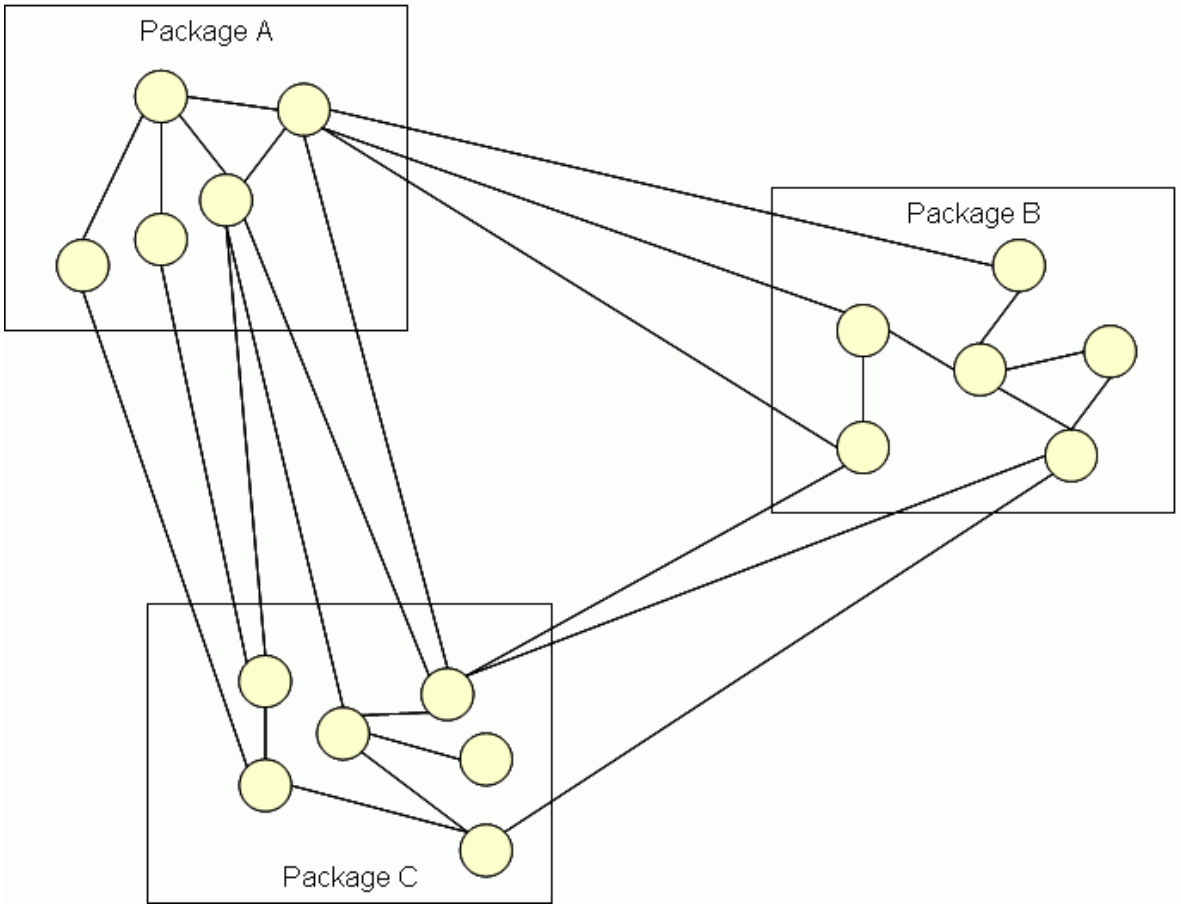
- Hierarchical (de)composition to cope with the complexity.
  (it is easier to design and implement smaller components)
- However, there can be problematic integration.
- Software architecture must have
  - high cohesion
    (the degree to which the elements of a component belong together)
  - low coupling
    (the degree to which the different components depend on each other)
- Only encapsulated code can be modified/reimplemented.
  (transparently, without breaking depended components)
- Each upgrade usually increase complexity and risk of failures.
  (software architecture erosion; the need for refactoring)

Modularity and Software Cost



(adopted from Software Engineering: A Practitioner's Approach by R. Pressman)

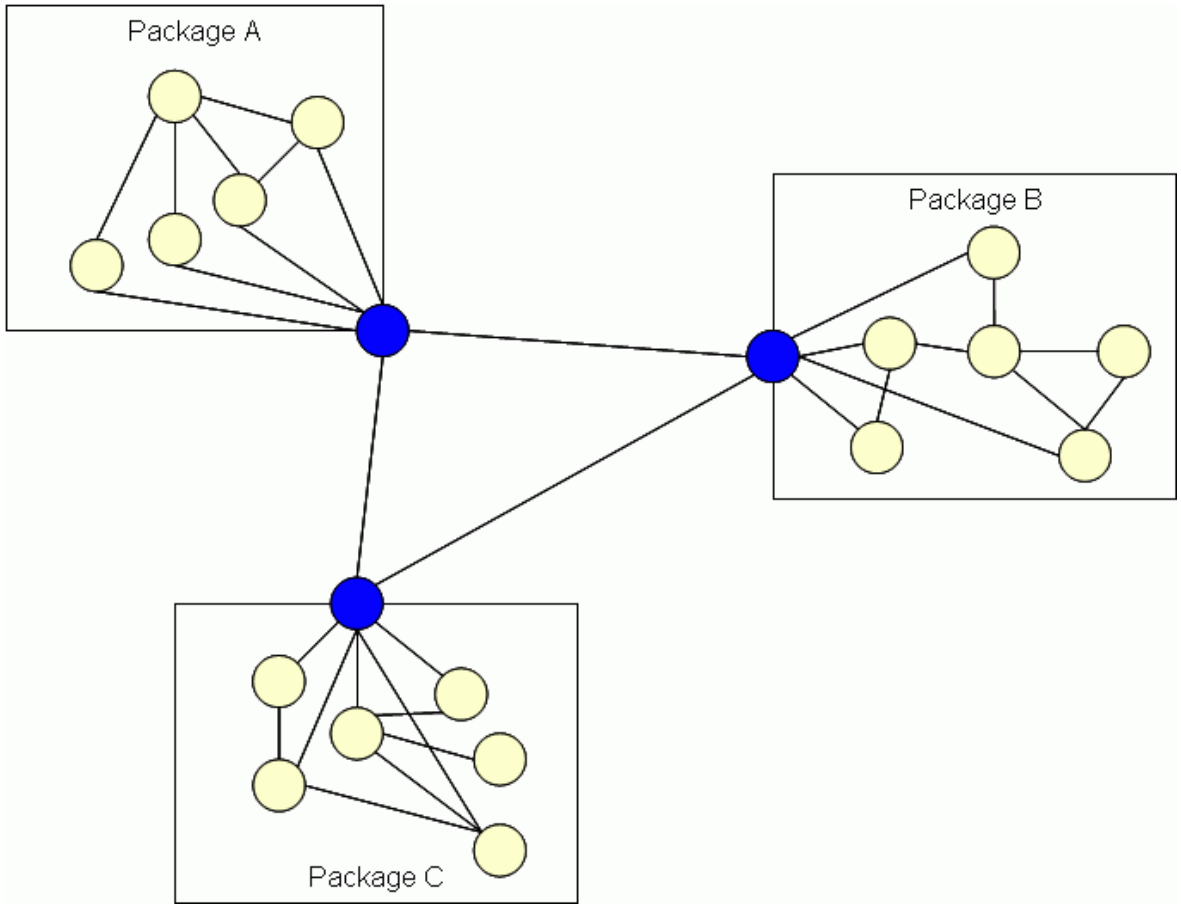
Software Complexity – High Coupling



(adopted from [Maciaszek and Liong(2004)])

Reducing Software Complexity – Low Coupling

(e.g., by application of “facade” design pattern)



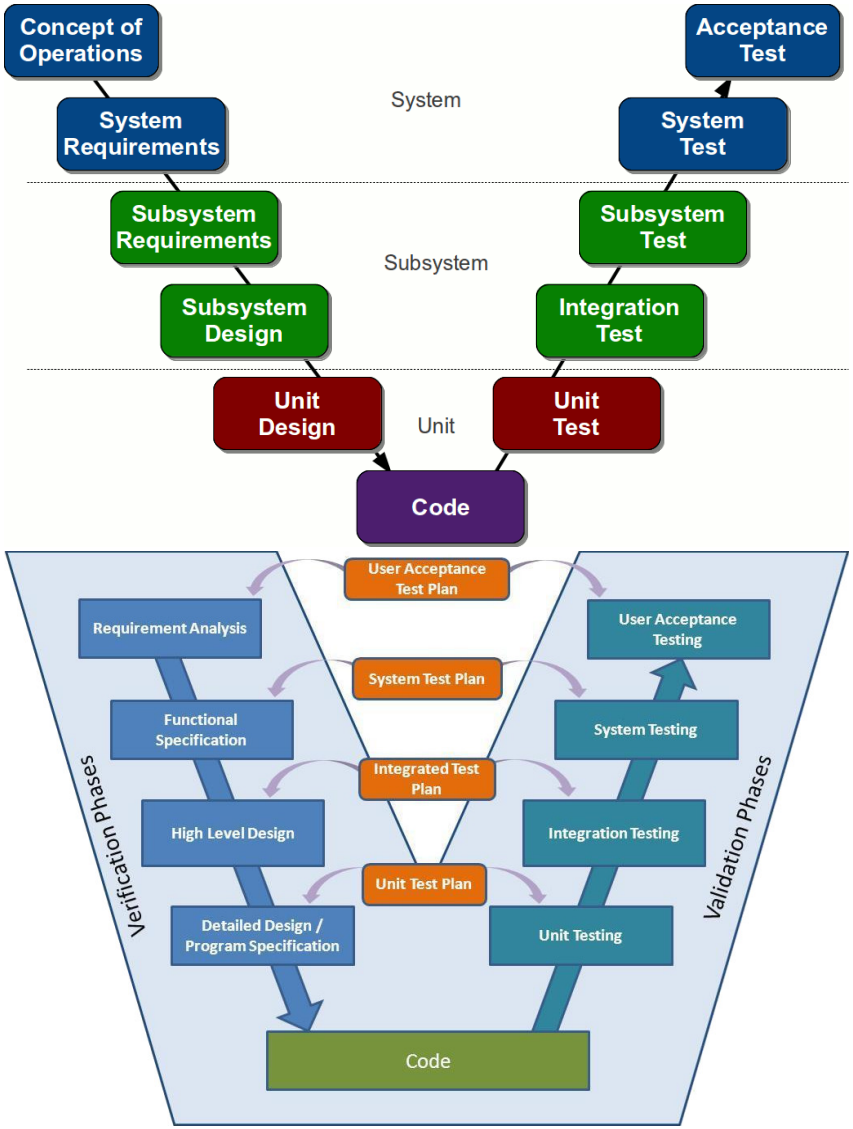
(adopted from [Maciaszek and Liong(2004)])

Software (Development) Process Models

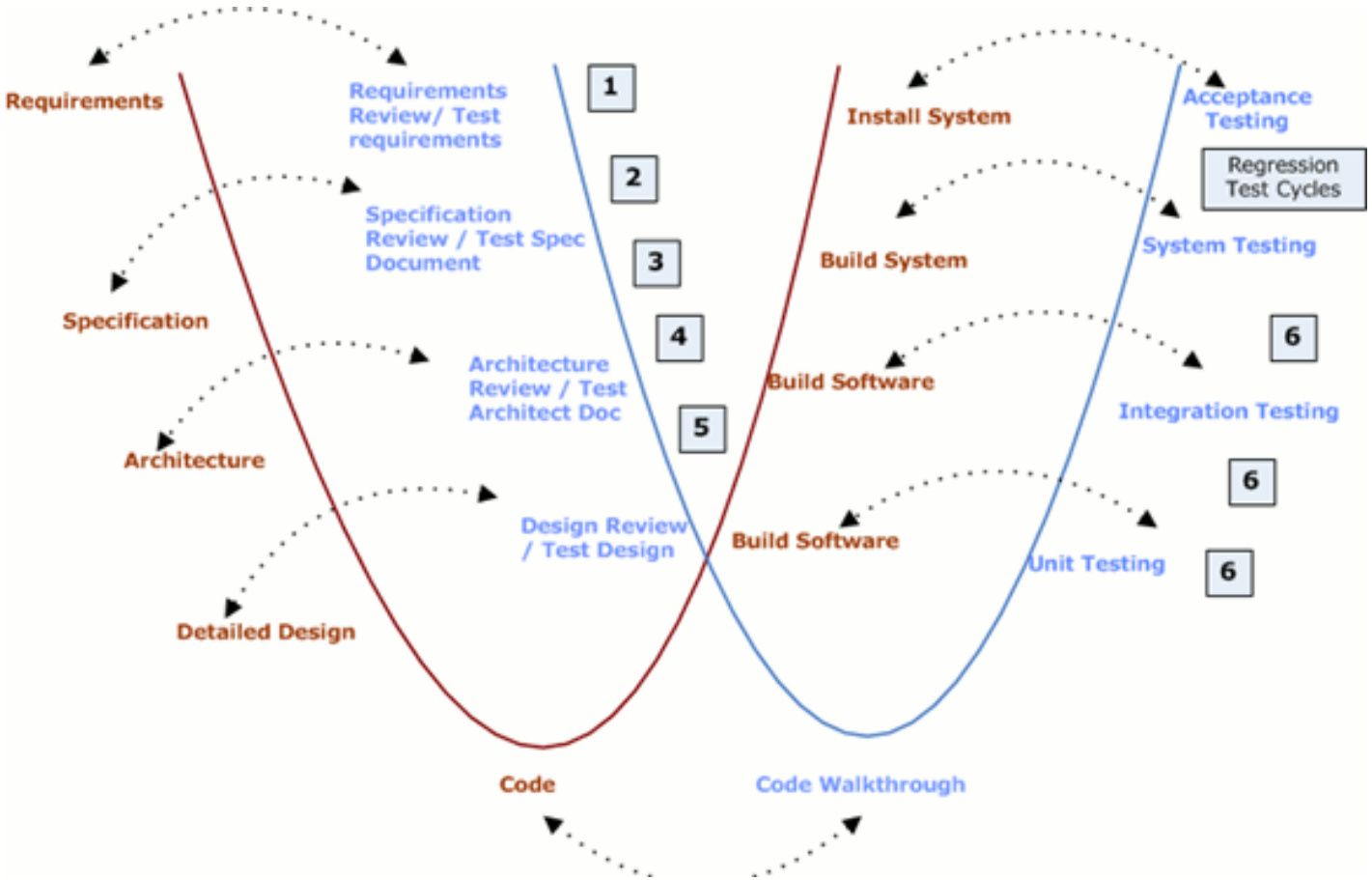
- The models describe the process of software development.
  (including its specification, design, implementation, integration, maintenance, etc.)
- Suitability of a development process model depends on:
  - knowledge and skills of the project teams
  - knowledge and understanding of the application domain
  - type of the application domain
  - stability of internal/external environment
  - size and complexity of the software project
- There are linear (sequence) and iterative development models.

linear waterfall, V model, W model
iterative spiral, specification/evolutionary prototyping

V Model


Concept of Operations
System Requirements
Subsystem Requirements
Subsystem Design
Unit Design
Code
Unit Test
Integration Test
Subsystem Test
System Test
Acceptance Test
System
Subsystem
Unit
User Acceptance Test Plan
System Test Plan
Integrated Test Plan
Unit Test Plan
Requirement Analysis
Functional Specification
High Level Design
Detailed Design / Program Specification
Code
Unit Testing
Integration Testing
System Testing
User Acceptance Testing
Verification Phases
Validation Phases

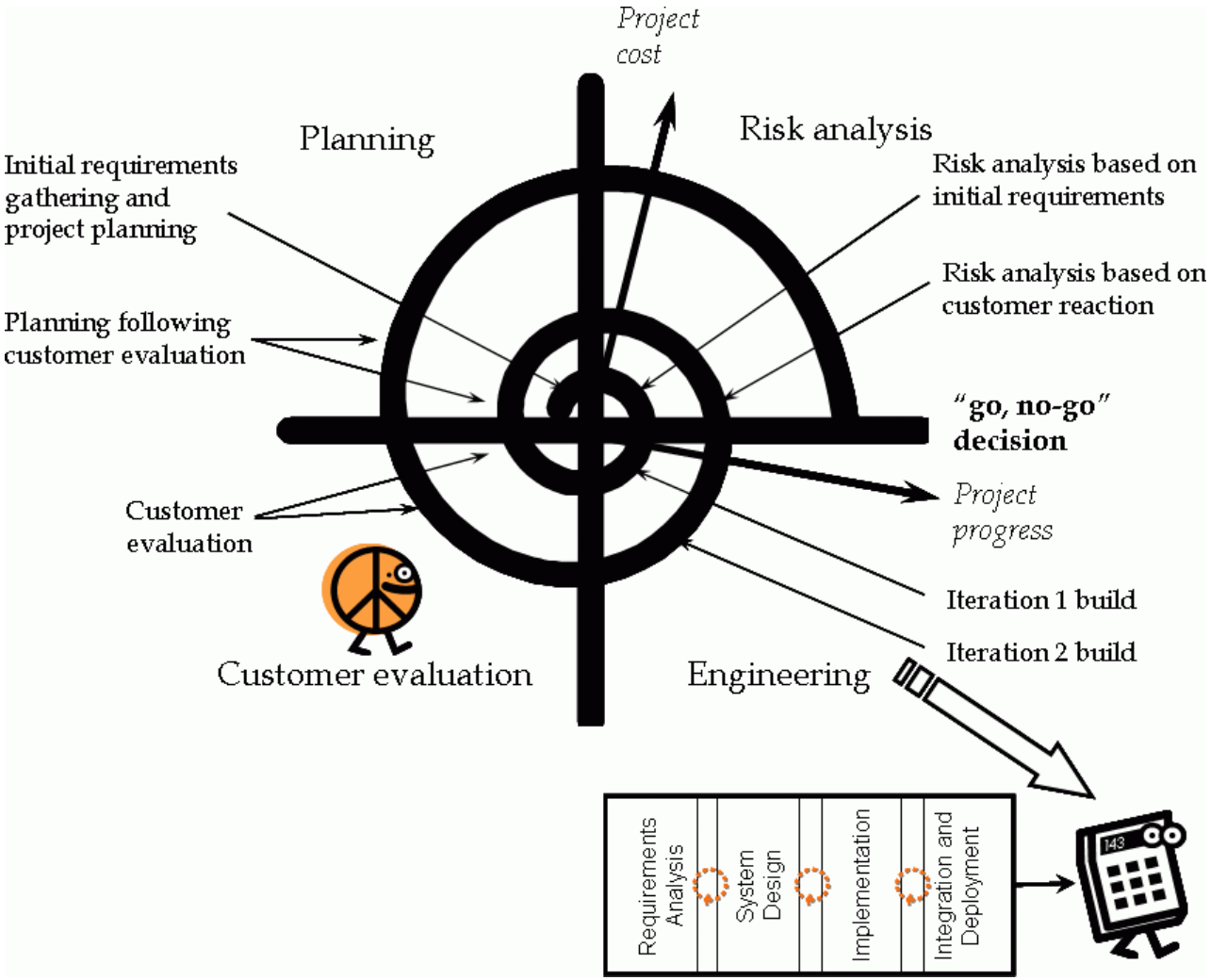
W Model



(adopted from V Model to W Model, W Model in SDLC Simplified)

Spiral Model

(e.g., by application of “facade” design pattern)



(adopted from [Maciaszek and Liong(2004)])

Thank you for your attention!

Marek Rychlý
<rychly@fit.vutbr.cz>