

MACIASZEK, L.A. (2005):
Requirements Analysis and System Design, 2nd ed.
Addison Wesley, Harlow England, 504p.
ISBN 0 321 20464 6

Chapter 1
Software Process

© Pearson Education Limited 2005

Topics

- *The nature of software development*
- *System planning*
- *Systems for three management levels*
- *Software development lifecycle*
- *Problem statements for case studies (separate set of slides)*

The essence of software development

- *Only one out of three software projects complete on-time and on-budget (The Standish Group report, 2003)*
- *The **essence** of software development:*
 - *defined by the issues inherent in the software itself → software is a product of a creative act (not a result of a repetitive act of manufacturing)*
 - *difficulties not amenable to breakthroughs or 'silver bullets' → define software development **invariants***
 - *consequence of the inherent software*
 - *complexity,*
 - *conformity,*
 - *changeability, and*
 - *invisibility.*

The accidents of software development

- **'Accidental difficulties'** due to software production practices → amenable to human intervention
 - attributed mostly to the fact that an information system is a social system
 - must not be adding to the complexity and to the potential lack of supportability of the software product
 - supportability = understandability + maintainability + scalability (extensibility)
- **Related to:**
 - Stakeholders
 - Process
 - Modeling language and tools
- **Will be discussed soon**

Software development invariants

- *Software is inherently **complex***
 - *function of interdependencies between components of which the software product is composed → complexity in the wires*
 - *Data intensive applications (enterprise information systems) are particularly complex*
- *Software must **conform** to:*
 - *hardware/software platform on which it is built*
 - *pre-existing information systems*
- *Software must be built to accommodate **change***
- *Software is buried deeply in “**invisible**” programming statements, binary library code, and surrounding system software*

Stakeholders

- *People who have a stake in a software project:*
 - *Customers (users and system owners)*
 - *Developers (analysts, designers, programmers, etc.)*
- *Information systems are social systems → developed by people (developers) for people (customers)*
- *The main causes of software failure can be traced to the stakeholder factor*
 - *on the customer end, and*
 - *on the developer end*

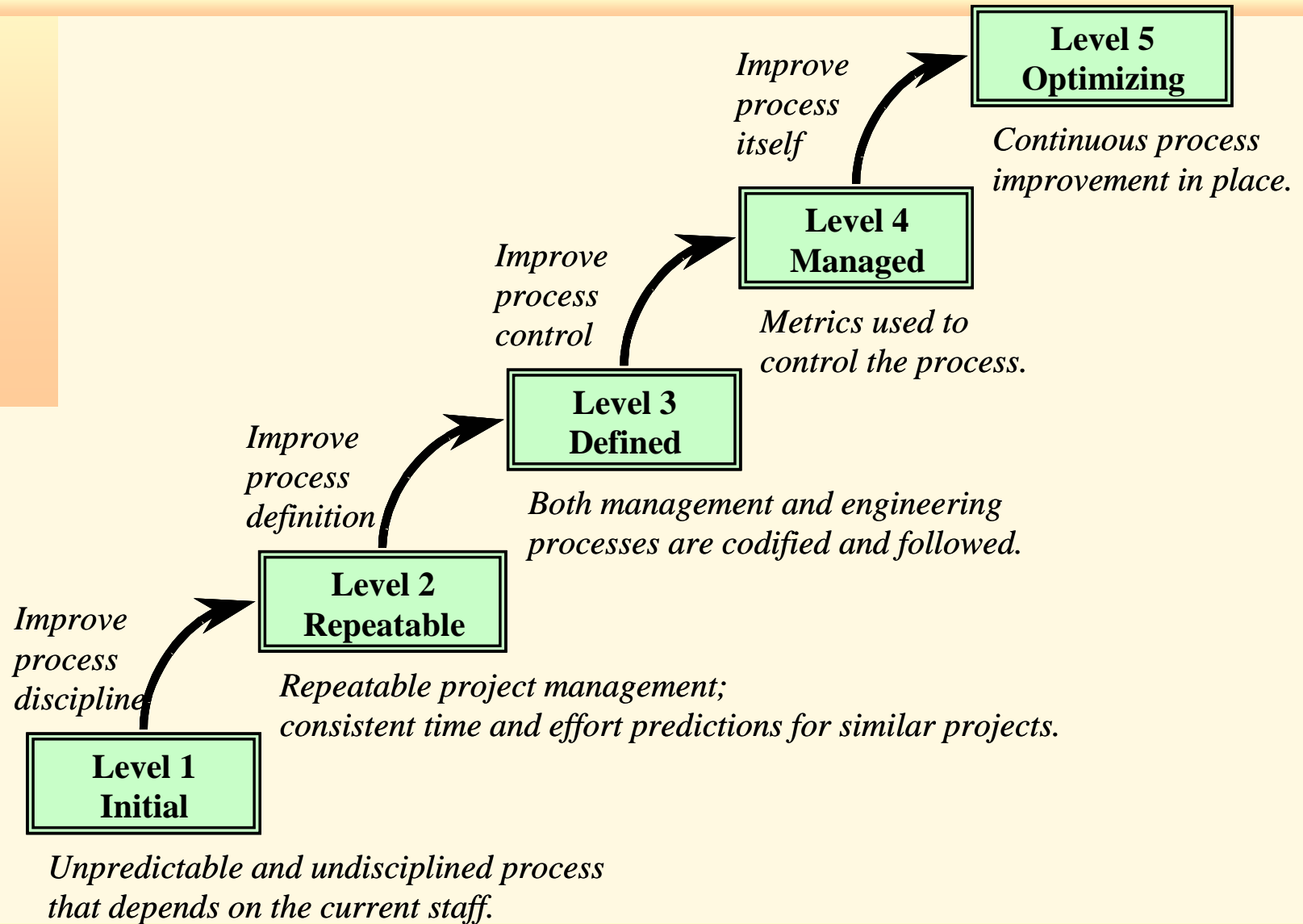
Software process

- *Defines activities and organizational procedures used in software production and maintenance*
- *A process model:*
 - *states an order for carrying out activities;*
 - *specifies what development artifacts are to be delivered and when;*
 - *assigns activities and artifacts to developers;*
 - *offers criteria for monitoring a project's progress, for measuring the outcomes, and for planning future projects.*
- *Is not susceptible to standardization*

Iterative and incremental process

- *'An iterative process is one that involves managing a stream of executable releases. An incremental process is one that involves the continuous integration of the system's architecture to produce these releases, with each new release embodying incremental improvements over the other.'* (RUP)
- *Some examples:*
 - *the spiral model*
 - *Rational Unified Process (RUP)*
 - *Model Driven Architecture (MDA)*
 - *the agile development process*
- *Iterative and incremental development*
 - *must be planned and controlled, and*
 - *must conform to a pre-defined architectural design framework*

Capability maturity model



ISO 9000 family of quality standards

- *Developed by the International Organization for Standardization*
- *The ISO standards*
 - *apply to the quality management and the process to produce a quality product*
 - *apply to any industry and all types of businesses, including software development*
- *The main premise*
 - *if the process is right then the process outcome (product or service) will also be right*
 - *but... the ISO standards do not enforce or specify processes → the standards provide models of what must be accomplished, not how activities must be performed*

Modeling language and tools

- *Modeling artifacts have to be*
 - *communicated (language) and*
 - *documented (tools)*
- *'The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system.'*
- *Computer-Assisted Software Engineering (CASE) tool enables storage and retrieval of models in a central repository and graphical and textual manipulation of models on a computer screen*

UML

- *UML is independent of*
 - *any software development process*
 - *A process that adopts UML must support an object-oriented approach to software production*
 - *implementation technologies (as long as they are object-oriented)*
 - *This makes UML somewhat deficient in supporting the detailed design phase of the development lifecycle*
- *The UML models can be categorized into three groups:*
 - *State models*
 - *describe the static data structures*
 - *Behavior models*
 - *describe object collaborations*
 - *State change models*
 - *describe the allowed states for the system over time*

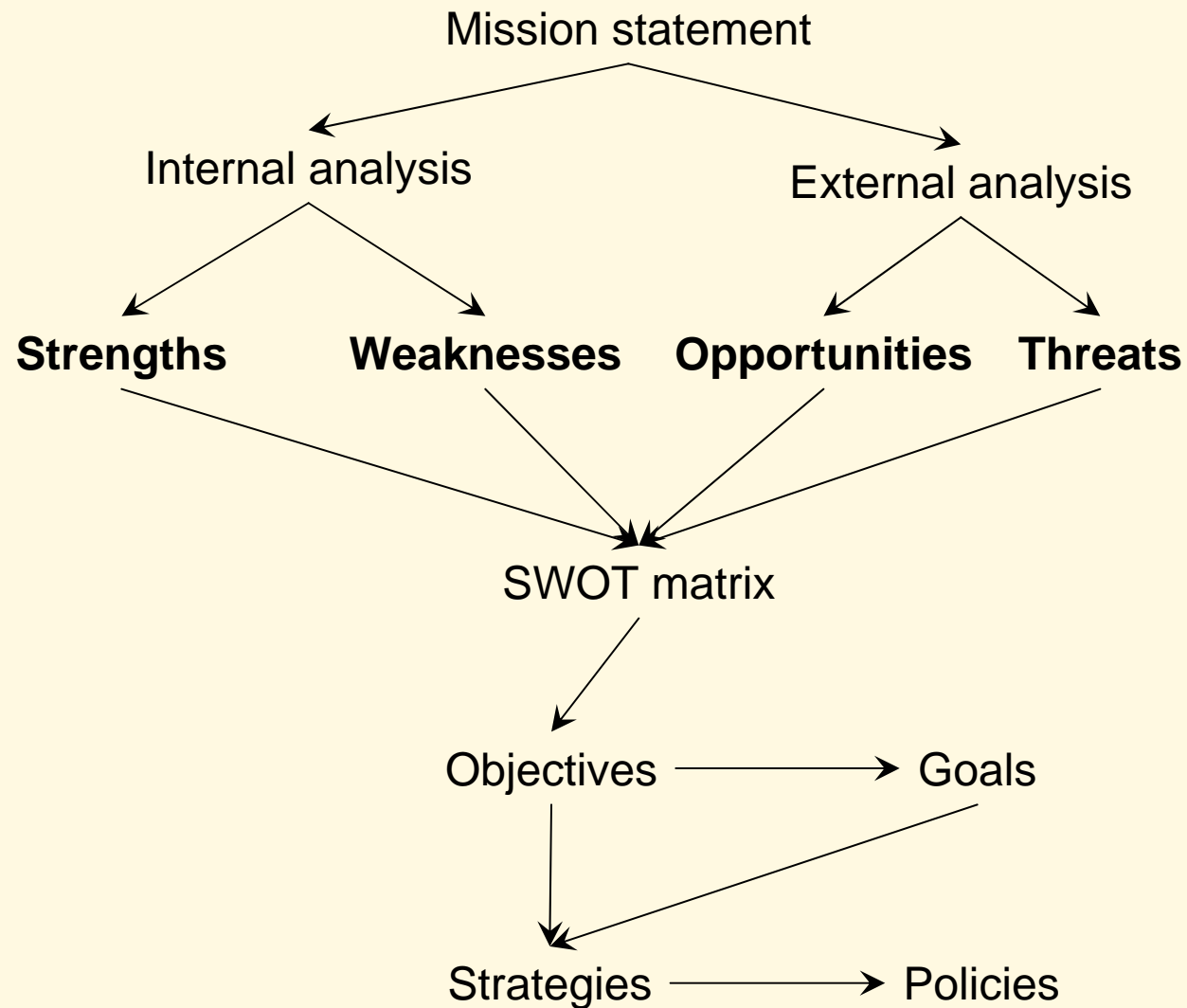
CASE and process improvement

- *Process improvement is much more than the introduction of new methods and techniques*
 - *the introduction of new methods and techniques to organization at a low level of process maturity can bring more harm than good*
- *An integrated CASE tool can allow multiple developers to collaborate and share design information in order to produce new design artifacts*
 - *the tool imposes processes on the development team*
 - *in “immature” organizations processes will not be followed (creating more mess than before)*
- *However, the same CASE methods and techniques would always bring personal productivity and quality improvements to individual developers*

System planning

- *The question is: Which IS technologies and applications will return the most value to the business?*
- *System planning can be carried out in a number of different ways:*
 - *SWOT – Strengths, Weaknesses, Opportunities, Threats*
 - *VCM – Value Chain Model*
 - *BPR – Business Process Reengineering*
 - *Information System Architecture (ISA)*
- *All system planning approaches have an important common denominator – they are concerned with effectiveness rather than efficiency*

SWOT approach



VCM approach

- *The VCM (Value Chain Model) assesses competitive advantage by analyzing the full chain of activities in an organization – from raw materials to final products sold and shipped to customers*
- *The question is: which value chain configurations will yield the greatest competitive advantage?*
 - *The IS development projects can then target those segments, operations, distribution channels, marketing approaches, etc. that give the most competitive advantage*
- *Organizational functions are categorized into:*
 - *primary activities*
 - *they create or add value to a final product*
 - *support activities*
 - *they are essential but they do not enrich the product*

BPR approach

- *The BPR (Business Process Reengineering) approach to system planning is based on the premise that today's organizations must reinvent themselves and abandon the functional decomposition, hierarchical structures and operational principles that they are now using*
 - *Most contemporary organizations are structured in vertical units focused on functions, products or regions*
 - *No one employee or department is responsible for a business process which is defined as ' . . . a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer'*
 - *'The most visible difference between a process enterprise and a traditional organization is the existence of process owners'*
- *The main objective of BPR is to radically redesign business processes in an organization (hence, **process redesign**)*
 - *The major hurdle lies in the need to embed a horizontal process in a traditional vertical management structure*
 - *BPR initiative requires changing the organization around the development teams as the primary organizational units*
 - *These teams are responsible for one or more end-to-end business processes*

ISA approach

- *ISA (Information Systems Architecture) is a bottom-up approach that offers a neutral architectural framework for IS solutions that can suit a variety of business strategies*
 - *it does not include a system planning methodology*
 - *it offers a framework that leverages most business strategies*
- *The ISA framework is represented as a table of thirty cells organized into five rows (labeled 1 through 5) and six columns (labeled A through F)*
 - *Rows represent the different perspectives used in the construction of a complex engineering product, such as an information system → five major 'players in the game'*
 - *Columns represent the six different descriptions or architectural models that each of the participants engages with*

Systems for three management levels

Level of decision making	Focus of decision making	Typical IS applications	Typical IT solutions	Pivotal concept
Strategic (executive and senior management levels)	Strategies in support of organizational long-term objectives	Market and sales analysis, Product planning, Performance evaluation	Data mining, Knowledge management	Knowledge
Tactical (line management level)	Policies in support of short-term goals and resource allocation	Budget analysis, Salary forecasting, Inventory scheduling, Customer service	Data warehouse, Analytical processing, Spreadsheets	Information
Operational (operative management level)	Day-to-day staff activities and production support	Payroll, Invoicing, Purchasing, Accounting	Database, Transactional processing, Application generators	Data

Software development lifecycle

- *Set of activities conducted and managed for development project*
- *The lifecycle identifies:*
 - *the applied modeling approach*
 - *the exact phases along which the software product is transformed – from initial inception to phasing it out*
 - *the method (methodology) and associated development process*

Development approach

- *Software has become much more **interactive***
 - *event-driven*
 - *user in control*
 - *software objects service random and unpredictable events*
- *Conventional software has been well served by the so-called **structured approach***
- *Modern GUI systems require object programming and the **object approach** is the best way to design such systems*

Structured approach

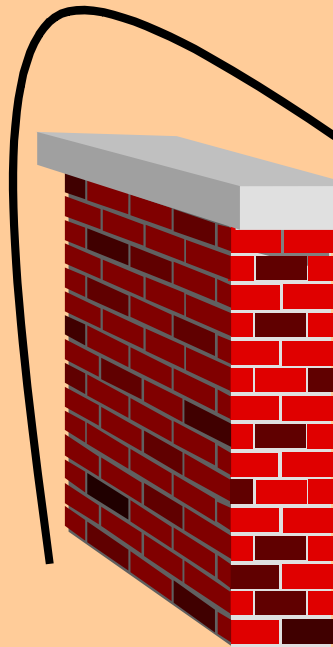
ANALYSIS

DFD

(processes, flows, stores, external entities)

ERD (entities, relationships, attributes)

User interface prototype



DESIGN

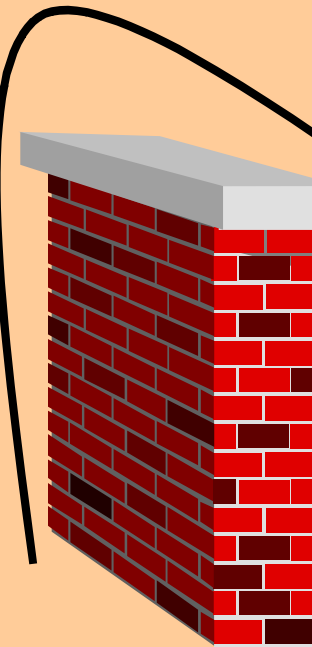
Architectural design

Schema design

(tables, views, referential integrity, indexes, etc.)

Server program design (stored procedures, triggers)

4GL application design



IMPLEMENTATION

Server programming,

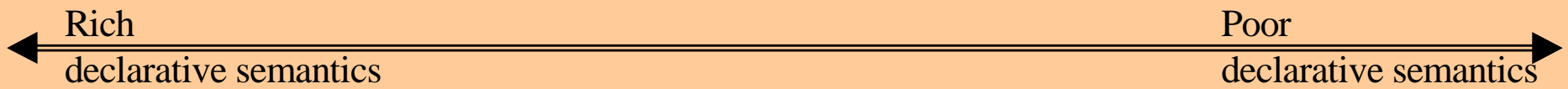
Client programming,

Testing,

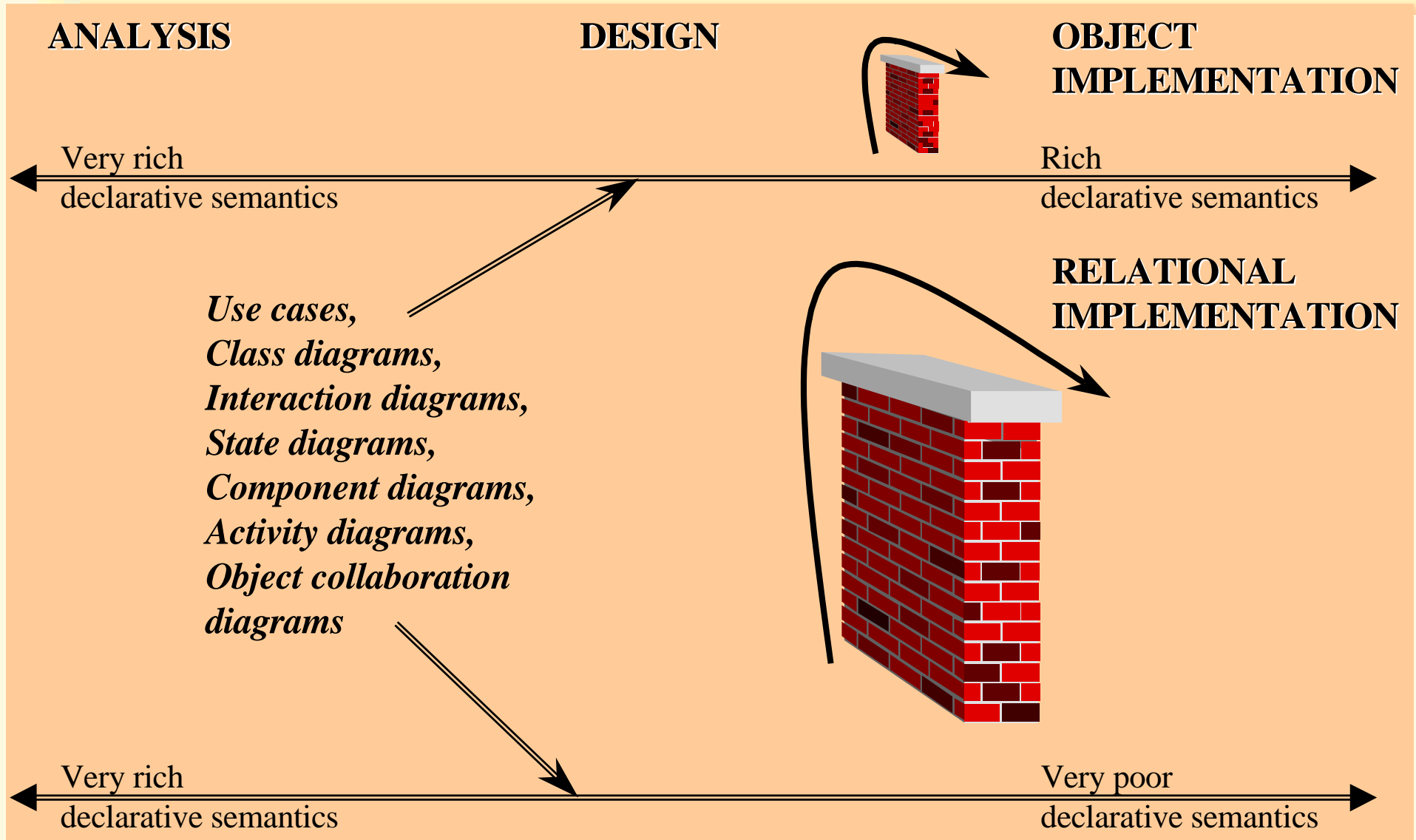
Performance tuning,

Three-tier deployment,

etc.



Object-oriented approach



Lifecycle phases

- *Business Analysis*
 - *functional and non-functional requirements*
- *System Design*
 - *architectural design*
 - *detailed design*
- *Implementation*
 - *coding*
 - *round-trip engineering*
- *Integration and Deployment*
- *Operation and Maintenance*

Business analysis

- **Also requirements analysis**
- *Activity of determining and specifying customer requirements*
 - *business analyst determines requirements*
 - *system analyst specifies (or models) requirements*
- *Business analysis is linked to **business process reengineering (BPR)***
 - *aim of BPR is to propose new ways of conducting business and gaining competitive advantage*
- *Business analysis becomes increasingly an act of **requirements engineering***

Requirements determination

- **Requirement** – ‘a statement of a system service or constraint’
- **Service statement**
 - a business rule that must be obeyed at all times (e.g. ‘fortnightly salaries are paid on Wednesdays’)
 - a computation that the system must carry out (e.g. ‘calculate salesperson commission based on the sales in the last fortnight using a particular formula’)
- **Constraint statement**
 - a restriction on the system’s behavior (‘only direct managers can see the salary information of their staff’)
 - a restriction on the system’s development (‘we must use Sybase development tools’)

Requirements specification

- *Begins when the developers start modeling the requirements using a particular method (such as UML)*
 - *CASE tool is used to enter, analyze and document the models*
 - *Requirements Document is enriched with graphical models and CASE-generated reports*
 - *Specifications document (the specs in the jargon) replaces the requirements document*
- *Most important specification techniques*
 - *class diagrams*
 - *use case diagrams*
- *Ideally, the specification models should be independent from the hardware/software platform on which the system is to be deployed*

Architectural design

- *The description of the system in terms of its modules (components)*
- *Concerned with*
 - *selection of a solution strategy*
 - *to resolve client (user interface) and server (database) issues as well as any middleware needed to ‘glue’ client and server processes*
 - *modularization of the system*
 - *relatively independent from a solution strategy but the detailed design of components must conform to a selected client/server solution*
- *Client/server models are frequently extended to provide a three-tier architecture where application logic constitutes a separate layer*
- *Good architectural design produces supportable systems, i.e. systems that are understandable, maintainable, and scalable (extensible)*

Detailed design

- *Description of the internal workings of each software component*
- *Develops detailed algorithms and data structures for each component*
- *Dependent on the underlying implementation platform*
 - *client*
 - *server*
 - *middleware*

Implementation

■ *Involves*

- *installation of purchased software*
- *coding of custom-written software*
- *other important activities, such as loading of test and production databases, testing, user training, hardware issues, etc.*

■ *Client programs*

■ *Server programs*

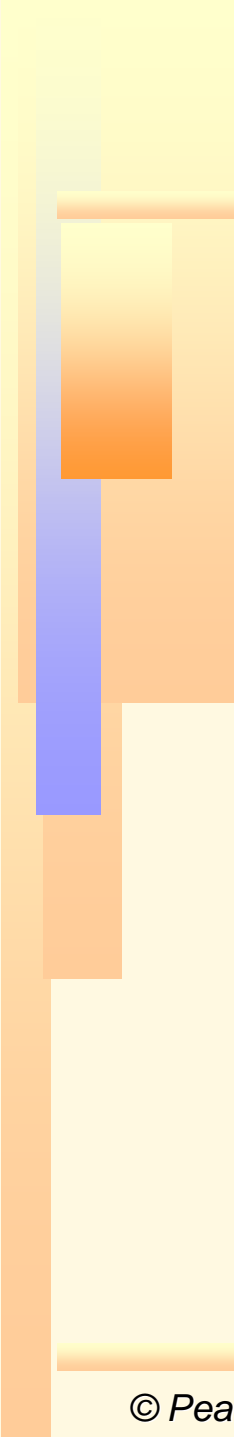
Integration and deployment

- ***Module integration*** can take more time and effort than any one of the earlier lifecycle phases, including implementation
 - *'The whole is more than the sum of the parts'* (Aristotle)
 - *must be carefully planned from the very beginning of the software lifecycle*
- ***Deployment*** must be carefully managed and allow, if at all possible, fallback to the old solution, if problems encountered

Operation and maintenance

- **Operation** signifies change over from the existing business solution, whether in software or not
- **Maintenance** is not only an inherent part of the software lifecycle – it accounts for most of it as far as IT personnel time and effort is concerned
 - Housekeeping
 - Adaptive maintenance
 - Perfective maintenance
- **Phasing out** would normally happen due to reasons that have little to do with the usefulness of the software

Activities spanning the lifecycle



- *Project planning*

- *Metrics*

- *Testing*

Project planning

- *If you can't plan it, you can't do it*
- *Activity of estimating the project's deliverables, costs, time, risks, milestones, and resource requirements*
- *Includes the selection of development methods, processes, tools, standards, team organization, etc.*
- *A moving target*
- *Typical constraints are time and money*

Metrics

- *Measuring development time and effort*
- *Without measuring the past, the organization is not able to plan accurately for the future*
- *Metrics are usually discussed in the context of software quality and complexity – they apply to the quality and complexity of the software product*
- *Equally important application of metrics is measuring the development models (development products) at different phases of the lifecycle → to assess the effectiveness of the process and to improve the quality of work at various lifecycle phases*

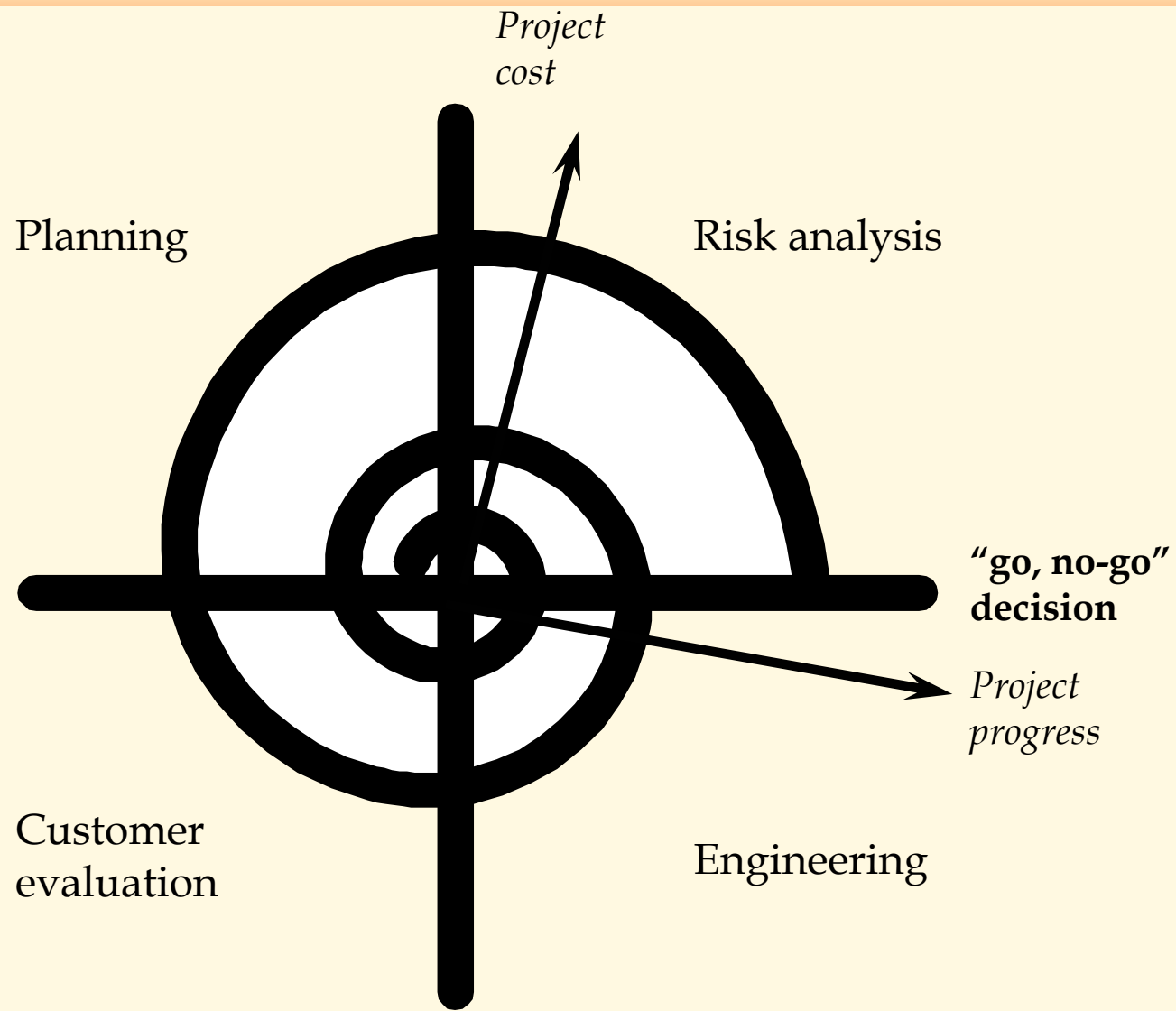
Testing

- **Test cases** should be defined for each functional module (use case) described in the requirements document
- Desktop testing by developers not sufficient
- Methodical testing by Software Quality Assurance (SQA) group necessary
- Requirements, specifications and any documents (including program source code) can be tested in formal reviews (so-called walkthroughs and inspections)
- **Execution-based testing:**
 - Testing to specs (black-box testing)
 - Testing to code (white-box or glass-box testing)

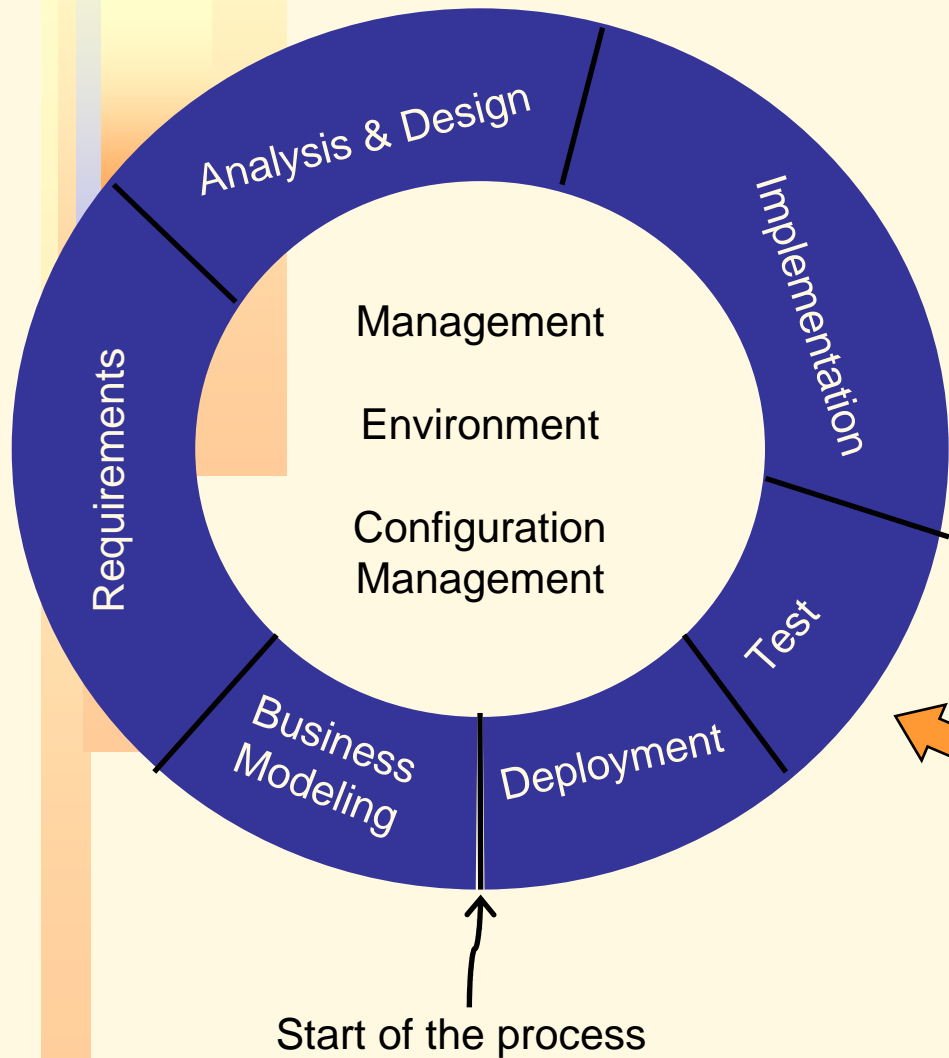
Development models

- *Called also lifecycle models*
- *Model defines an approach to software production*
- *Has an associated process – iterative and incremental*
- *Process is unique for each organization*
- *Representative models for iterative and incremental development:*
 - *the spiral model*
 - *the IBM Rational Unified Process (RUP)*
 - *the Model Driven Architecture (MDA)*
 - *the agile software development*

Spiral model

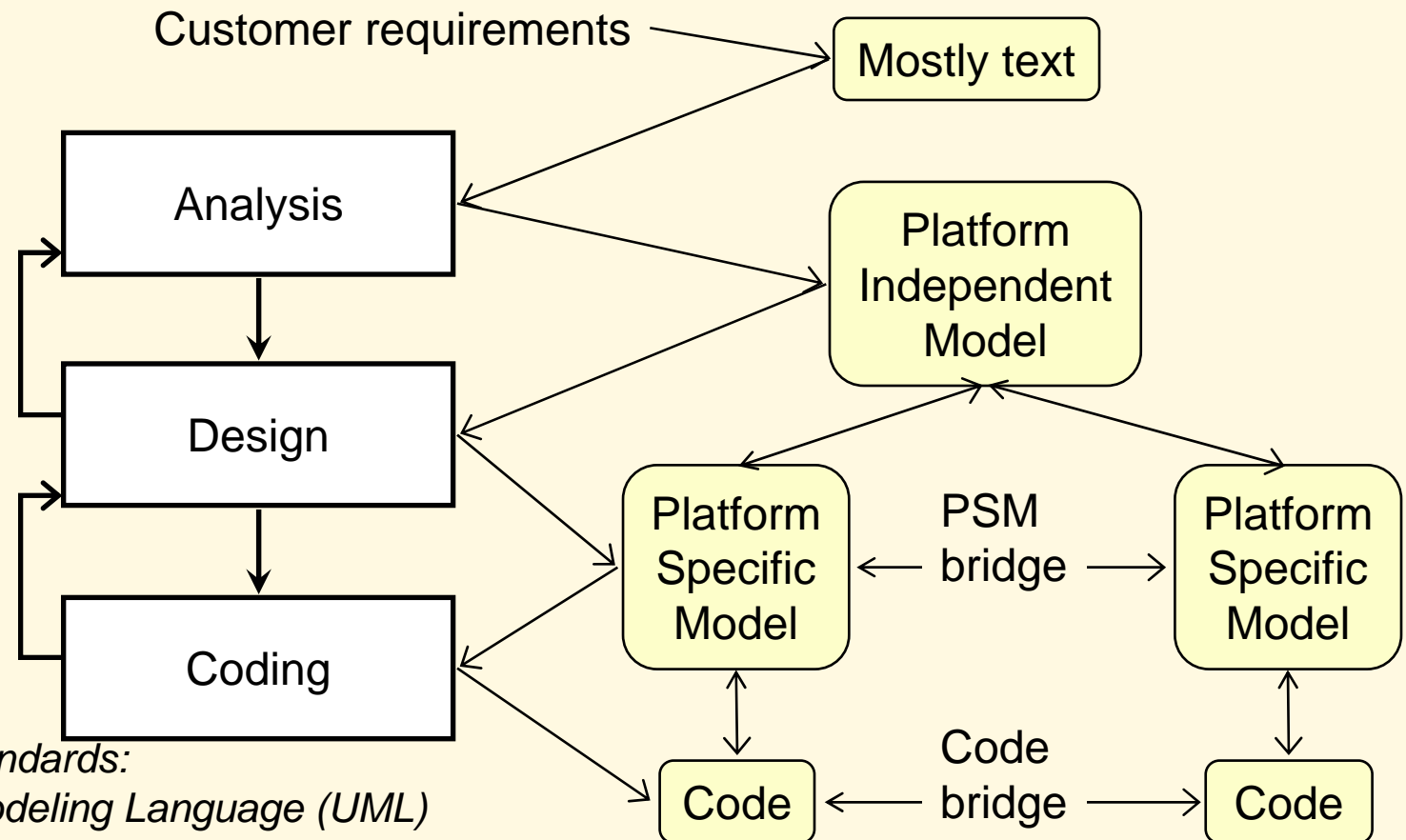


IBM Rational Unified Process (RUP)



- RUP organizes projects in two-dimensional terms
- The **horizontal dimension** represents the successive phases of each project iteration:
 - inception,
 - elaboration,
 - construction, and
 - transition.
- The **vertical dimension** represents seven software development disciplines and supporting activities of configuration and change management, project management, and environment.

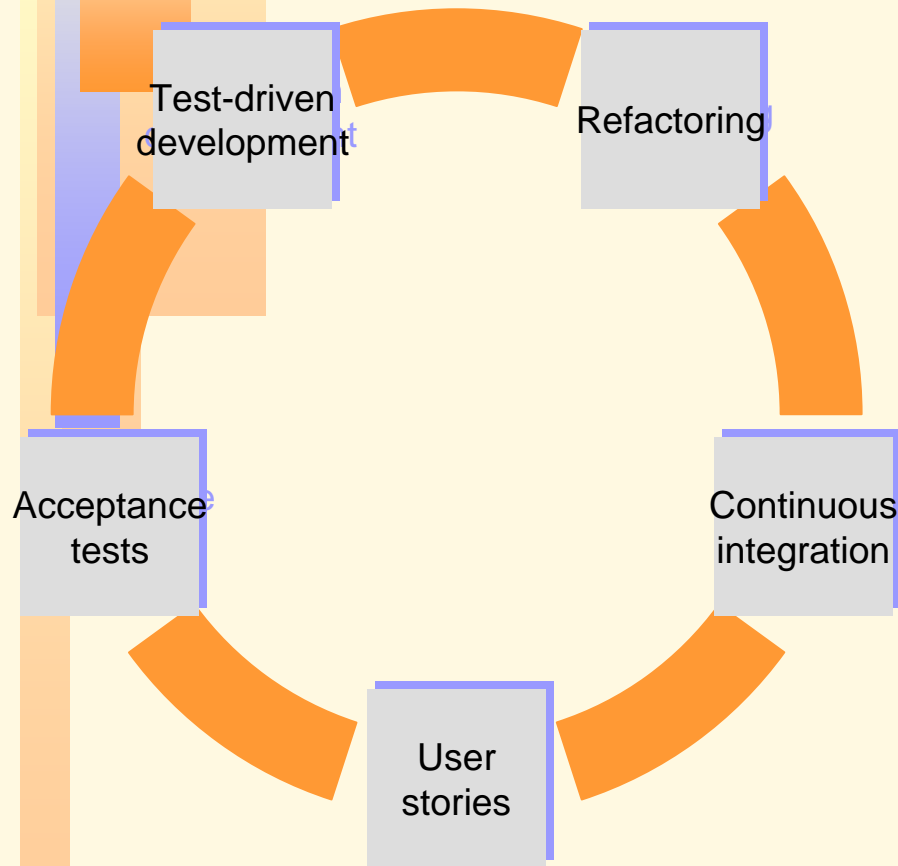
Model Driven Architecture (MDA)



Applicable standards:

- Unified Modeling Language (UML)
- Meta-Object Facility (MOF) for using a standard meta-model repository so that derived specifications can work together
- XML Meta-Data Interchange (XMI) for mapping UML to XML for interchange purposes,
- Common Warehouse Meta-model (CWM) for mapping of MDA to database schemas and permitting flexible data mining.

Agile software development



Key points of agility in software production:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

The best known representatives:

- *eXtreme Programming (XP)*
- *Aspect Oriented Software Development*
- *Feature-Driven Development*
- *Lean Development*

Case studies (separate set of slides)

- *University Enrolment (UE)*
- *Video Store (VS)*
- *Contact Management (CM)*
- *Telemarketing (TM)*
- *Advertising Expenditure (AE)*
- *Time Logging (TL)*

Summary

- *The nature of software development is that of a **craft** or even **art**.*
- *The triangle for success includes the **stakeholders'** factor, a sound **process** and the support of a modeling **language and tools**.*
- ***System planning** precedes software development and determines which products can be most effective to the organization.*
- *Modern software products are **object-oriented**.*
- *Software development follows a **lifecycle**.*
- *Modern development lifecycles and processes are **iterative and incremental**.*