Software engineering – Lecture 10
The Software Process

Adapted after ©**Ian Sommerville**
Software Engineering, 2010, chapter **2**

The Software Process

Def. The software process = structured set of activities required to develop a software system.

Main activities:

- specification;
- design and implementation;
- verification and validation;
- evolution.

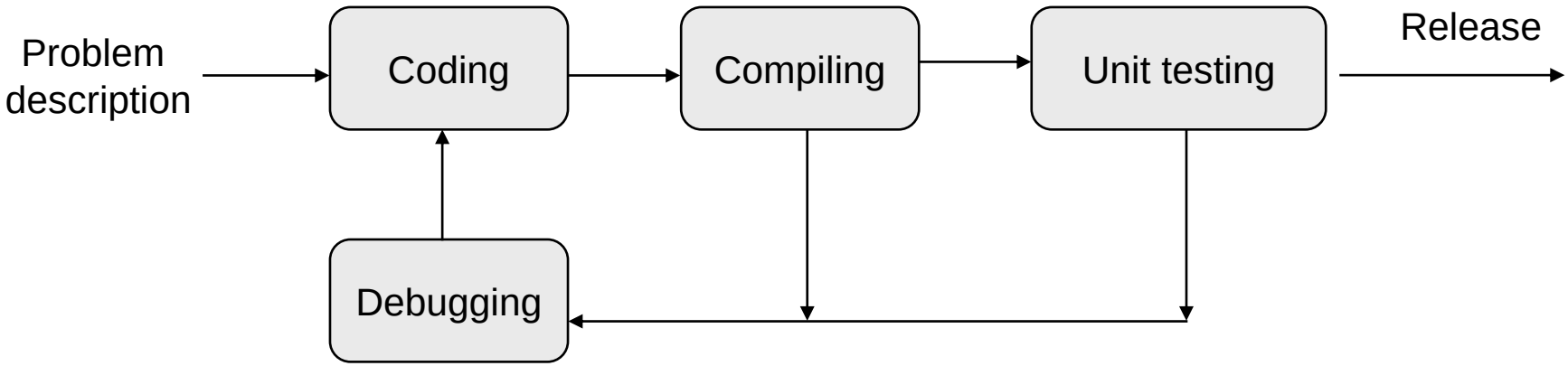
Def. Software process *model* = *abstract representation* of a software process.

Aim: *guidance* for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and the project objectives.

Presents a *description of a process* from some particular perspective.

Evolution of software process models

"Code and fix": *simple model*



Added tasks in *evoluated models*:

- Design
- Integration
- Process coordination

Different models applicable according to the project characteristics.

Plan-driven and agile processes

- **Plan-driven** processes : all of the process activities are *planned in advance* and progress is measured against this plan.

Used in *large, long-lifetime systems* (such as aircraft control systems) where *teams may be geographically dispersed* and work on the software for *several years*.

  - This approach is based on *controlled and rigorous software development processes* that include detailed project planning, requirements specification and analysis and system modelling.
  - However, plan-driven development *involves significant overheads and documentation* and it does not support the rapid development and delivery of software.

- **Agile** processes : *planning is incremental* and it is easier to change the process to reflect *changing* customer *requirements*.

Plan-driven and agile processes

- **Plan-driven** processes : all of the process activities are *planned in advance* and progress is measured against this plan.

- **Agile** processes : *planning is incremental* and it is easier to change the process to reflect *changing* customer *requirements*.

  These methods focus on the software rather than its documentation, develop software in a series of increments and aim to reduce process bureaucracy as much as possible.

In practice, most practical processes include elements of both plan-driven and agile approaches.

There are no right or wrong software processes but *appropriate* and not appropriate processes for a given set of requirements.

Topics covered

- **Generic models for the software process**
- Coping with change
- RUP (Rational Unified Process)
- Risk management

Generic software process models

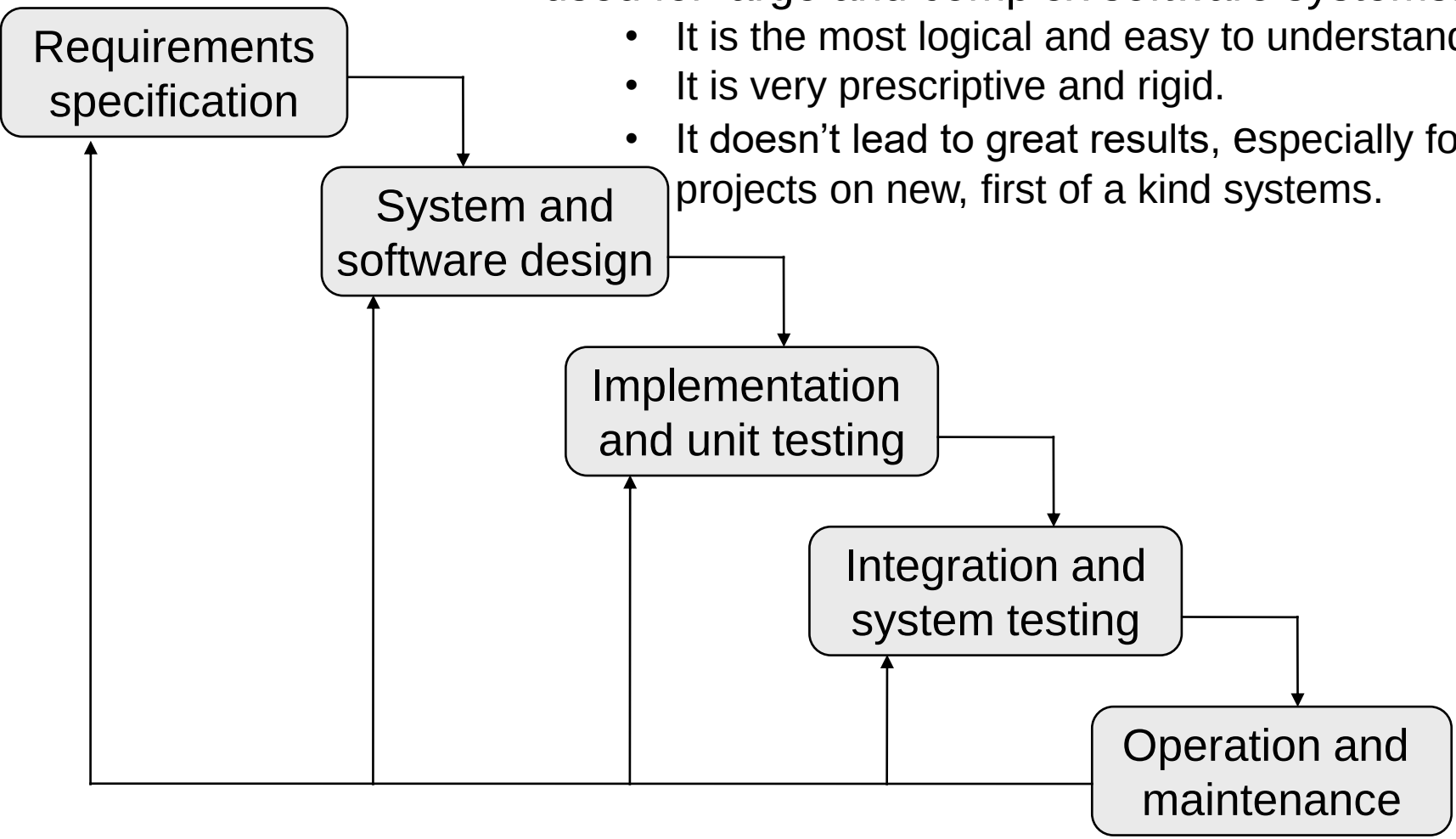
- **Waterfall model**
  - Separate and distinct phases of specification and development.
  - Plan-driven model.
- **Incremental development**
  - Specification, development and validation are interleaved.
  - May be plan-driven or agile.
- **Reuse-oriented development**
  - The system is assembled from existing components or services.
  - May be plan-driven or agile.

Obs.Not mutually exclusive and often used together.

Waterfall model

The most famous well defined process, defined to be used for large and complex software systems:

- It is the most logical and easy to understand.
- It is very prescriptive and rigid.
- It doesn't lead to great results, especially for projects on new, first of a kind systems.



Waterfall model problems

Main characteristic: Inflexible partitioning of the project into distinct stages: one phase has to be complete before moving onto the next phase.

⇓

The difficulty of responding to changing customer requirements after the process is underway.

⇓

Waterfall model appropriate when:

- the requirements are well-understood
- changes will be fairly limited during the design process.

Waterfall model is mostly used for *large systems* engineering projects, where a system is *developed at several sites*.

Incremental development

Incremental development = developing an *initial implementation*, exposing this to *user comment*, and *evolving* it through several versions until an adequate system has been developed.

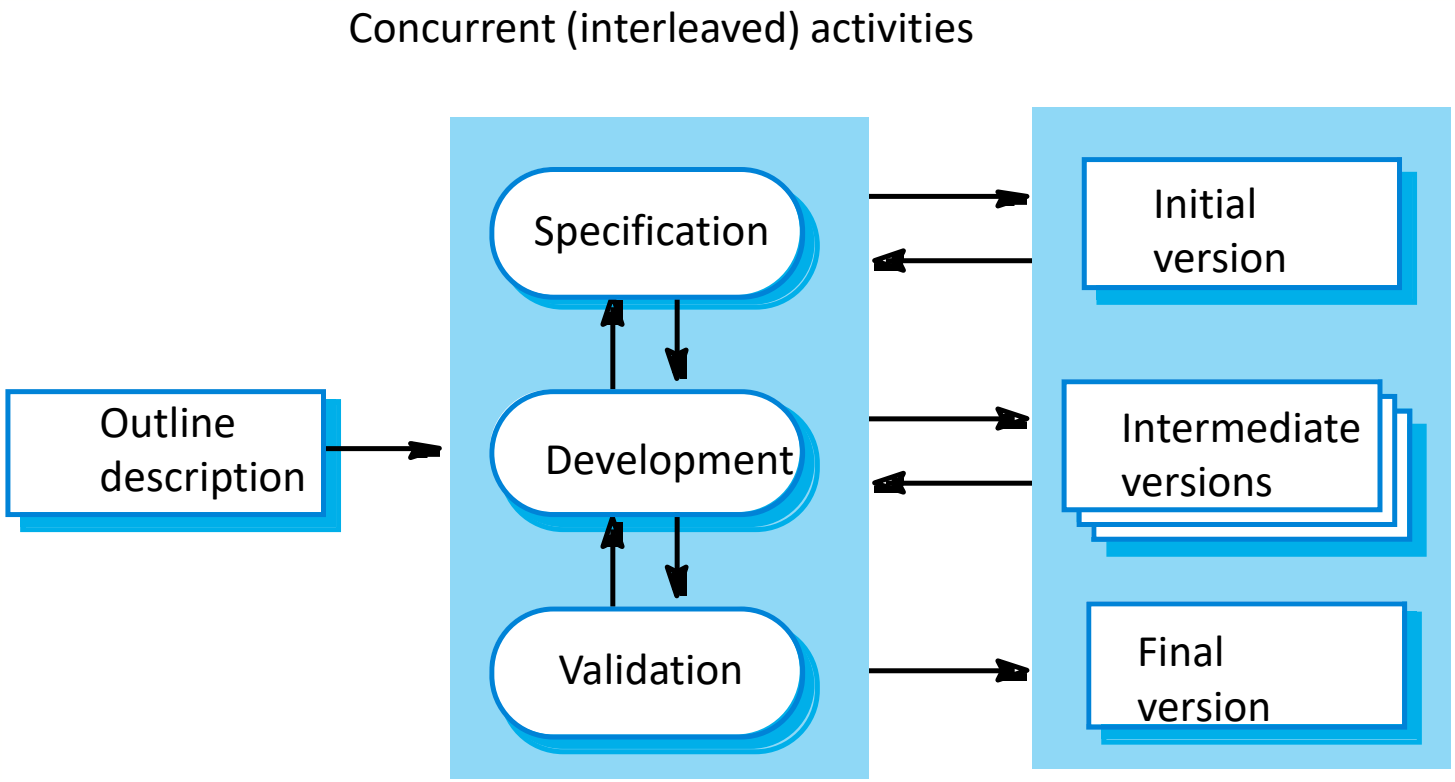
Plan-driven approach

- System increments are identified in advance.

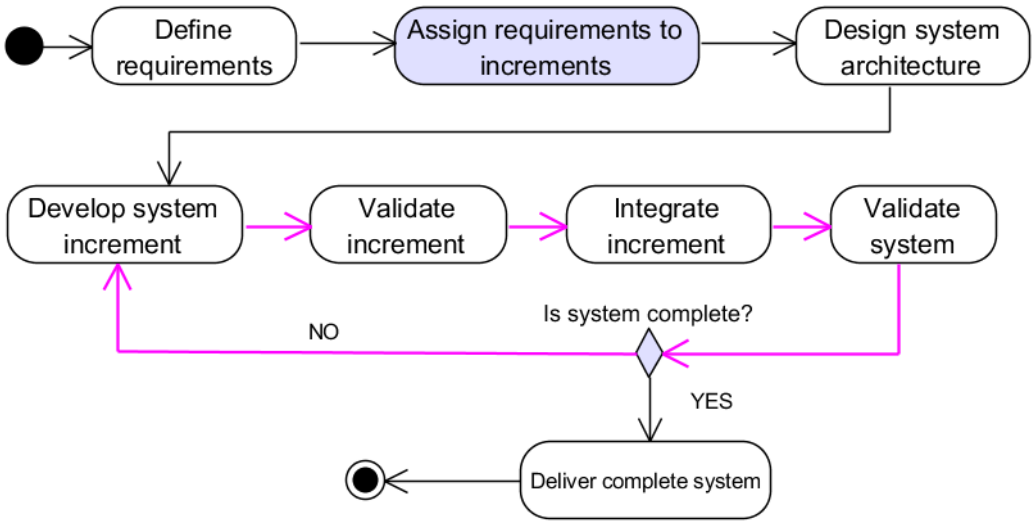
Agile appropach

- The early increments are identified.
- The development of later increments depends on progress and customer priorities.

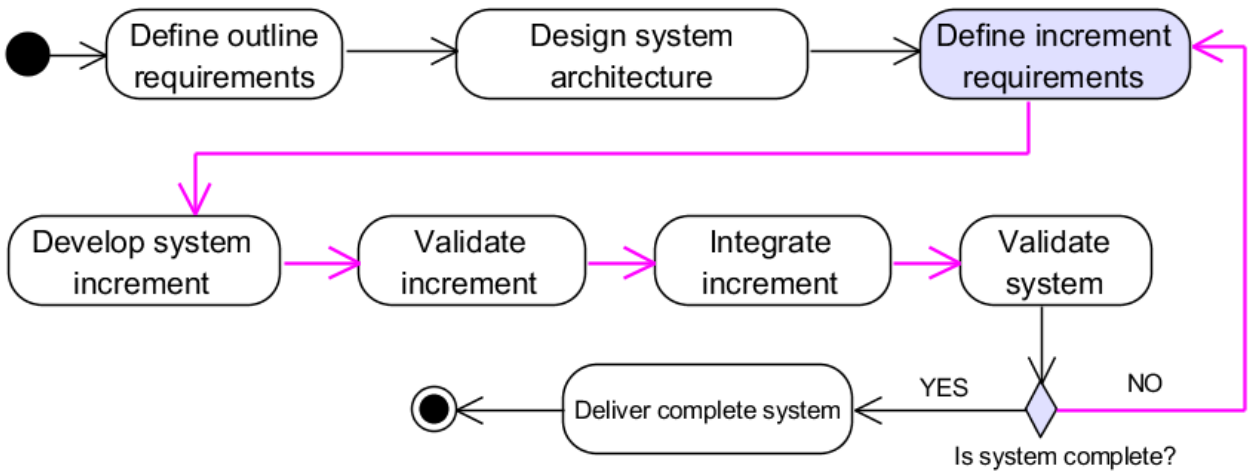
Incremental development



Incremental development



Plan-driven approach

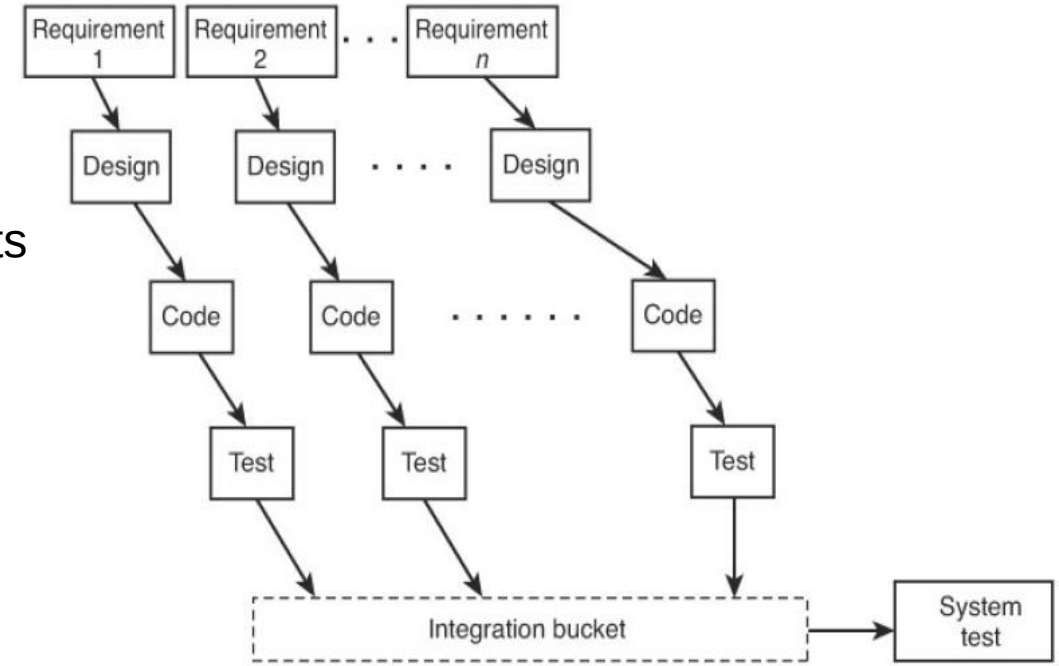


Agile approach

Incremental development

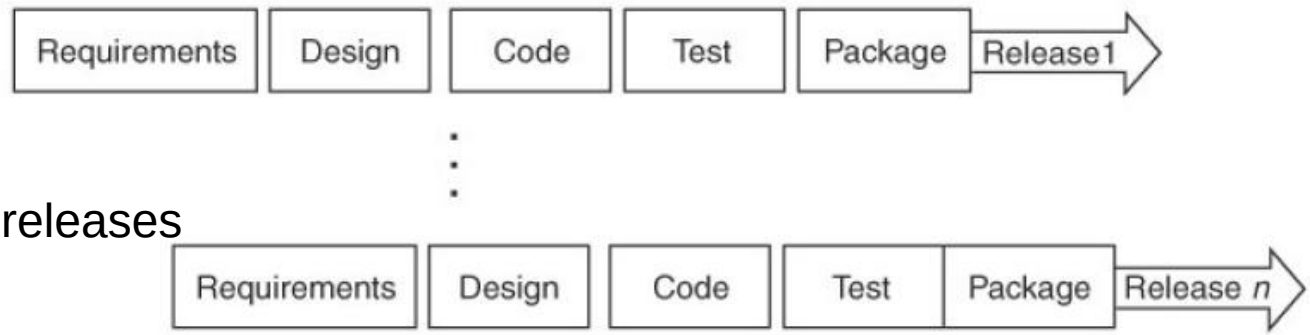
Incremental model

- system divided into components



Incremental model

- system with multiple releases



Incremental development - benefits

- The *cost* of accommodating *changing* customer requirements is *reduced*.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get *customer feedback* on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More *rapid delivery and deployment* of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development

Problems

- The process is not visible.
- System structure tends to degrade as new increments are added.

Applicability

- For small or medium-size interactive systems;
- For parts of large systems (e.g. the user interface);
- For short-lifetime systems. (**?**)

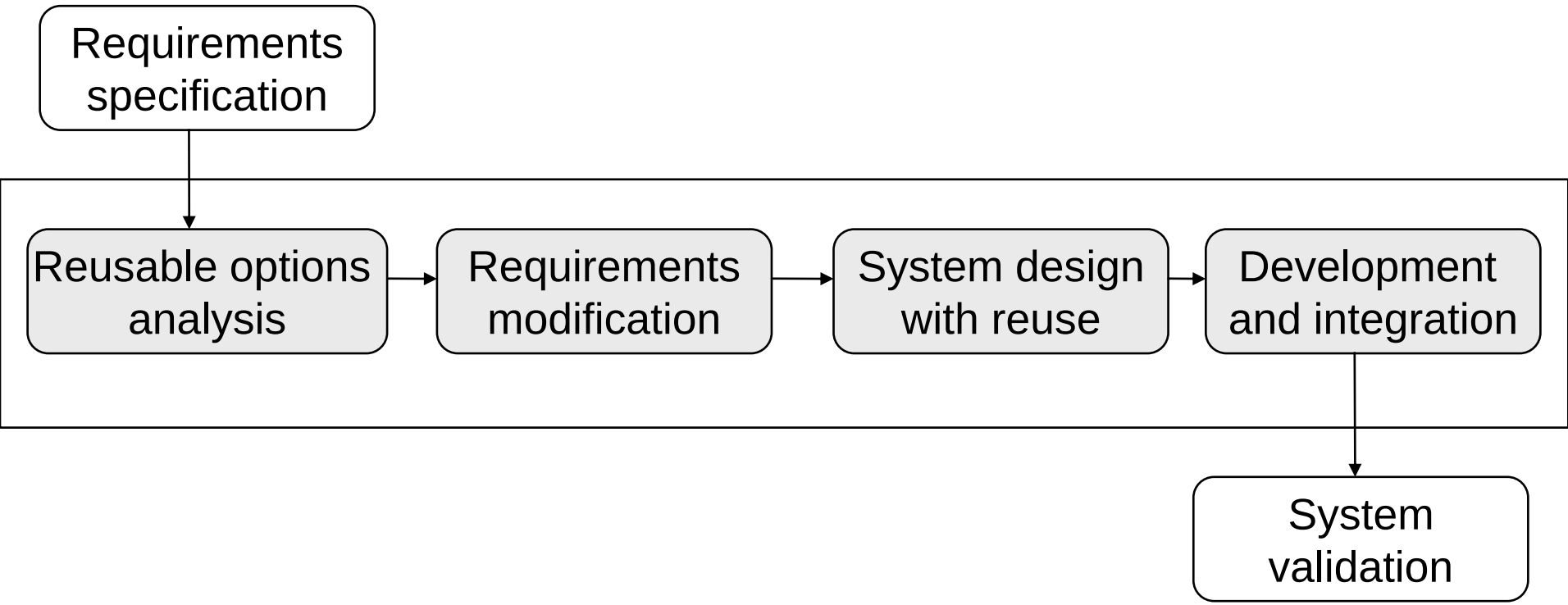
Reuse-oriented software engineering

Based on *systematic reuse*; systems are integrated from existing reusable software.

Main types of reusable software:

- *Components* (collections of objects that are developed as a package) to be integrated with a component framework such as .NET or J2EE.
- *Stand-alone software systems* (COTS) that are configured for use in a particular environment.
- *Web services* that are developed according to service standards and which are available for remote invocation.

Reuse-oriented development



This approach is becoming increasingly used, as component standards and web services have emerged.

Formative evaluation

1. Realize the correct mapping between the generic software process model and its essential characteristic.

2. Which of the generic software process models may be agile processes ?

https://forms.gle/R31orXppBFPgdmLF6

Topics covered

- Generic models for the software process
- **Coping with change**
- RUP (Rational Unified Process)
- Risk management

Coping with change

- Change is inevitable in all large software projects.
  - *Business changes* lead to new and changed system requirements.
  - *New technologies* open up new possibilities for improving implementations.
  - *Changing platforms* require application changes.
- Change leads to *re-work,* so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new/changed functionality.

Reducing the costs of rework

- **Change avoidance**: the software process includes activities that can *anticipate* possible changes before significant rework is required.

Example: a prototype system may be developed to show some key features of the system to customers.

- **Change tolerance**: the process is designed so that changes can be *accommodated* at relatively low cost.

Involves some form of *incremental development*. Proposed changes may be implemented in increments that have not yet been developed or only a single existing increment may have be altered to incorporate the change.

- **Combination** of change avoidance and change tolerance.

Spiral development model. In which changes are considered as result of project risks and includes explicit risk management activities to reduce these risks.

Software prototyping

Prototype = an initial version of a system used to

- demonstrate concepts and
- try out design options.

- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;

Software prototyping - benefits

- A closer match to users' real needs.
- Improved system usability.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

Prototype development

May be based on rapid prototyping languages or tools.

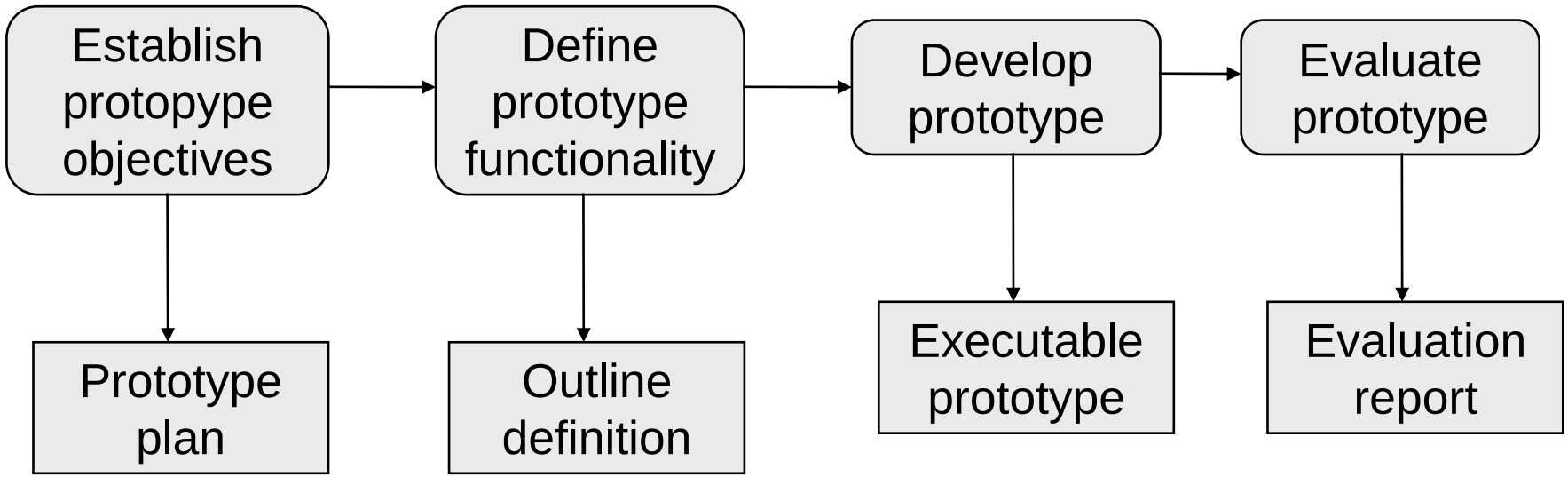
May involve leaving out functionality:

- Prototype should focus on areas of the product that are not well-understood;
- Error checking and recovery may not be included in the prototype;
- Focus on functional rather than extra-functional requirements such as reliability and security.

Prototype:

- Is generally undocumented
- Does not meet all system qualities
- May have a degraded structure
- Some requirements are relaxed

The process of prototype development



Incremental delivery

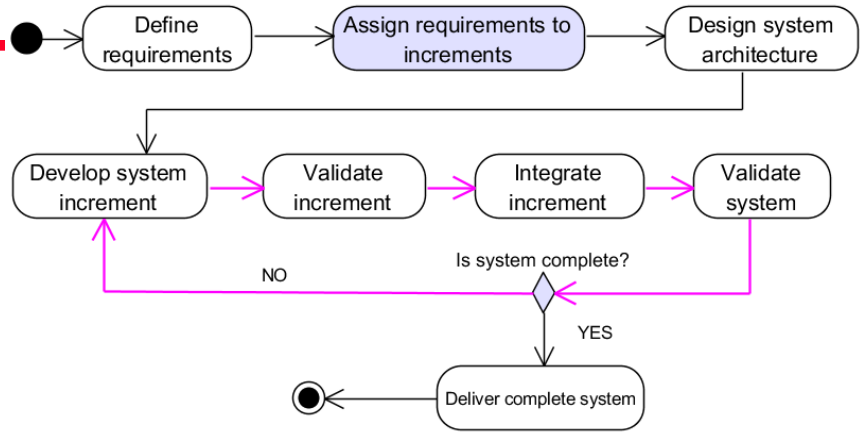
- Rather than deliver the system as a single delivery, the development and delivery is broken down into *increments*, with each increment delivering part of the required functionality.
- User requirements are *prioritised* and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are *frozen*, though requirements for later increments can continue to evolve.

Incremental development and delivery

- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer representative.

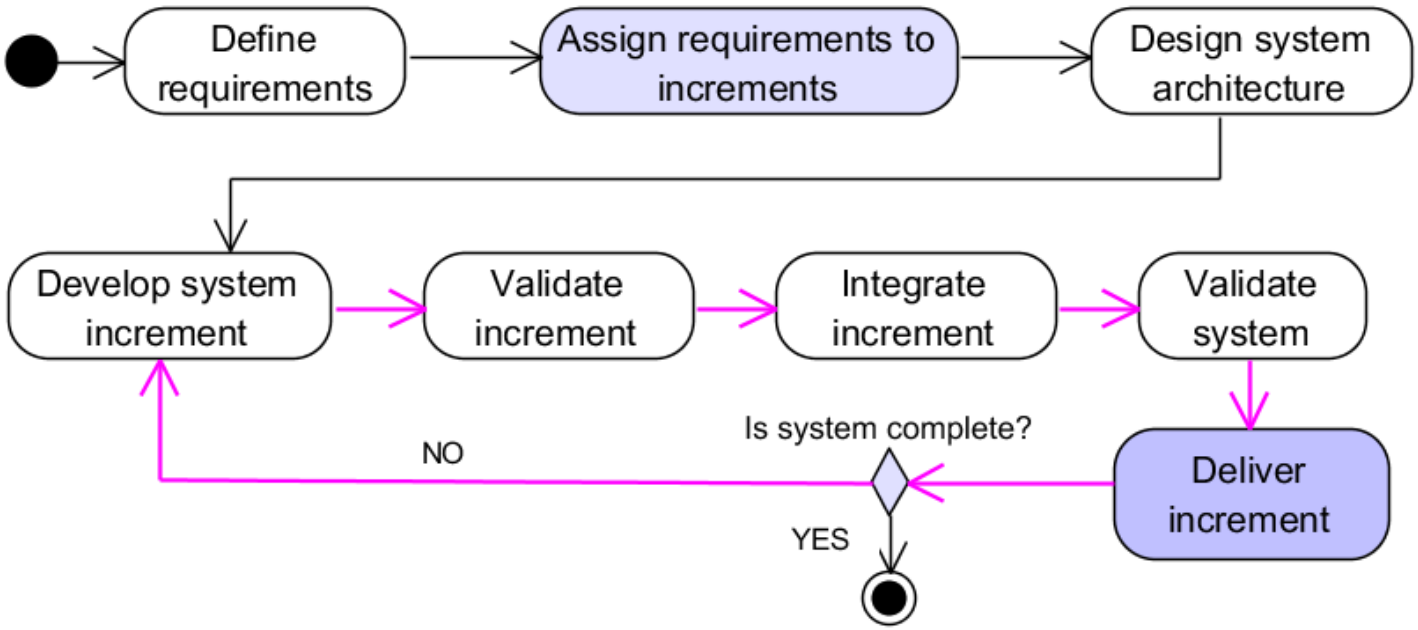
- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems, as increments have less functionality than the system being replaced.

Incremental delivery
Plan-driven approach

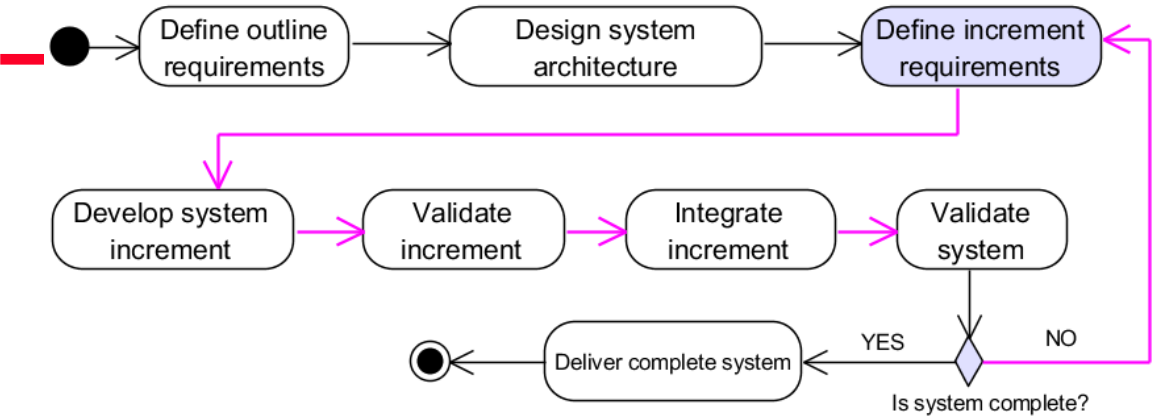


Incremental development

Incremental delivery

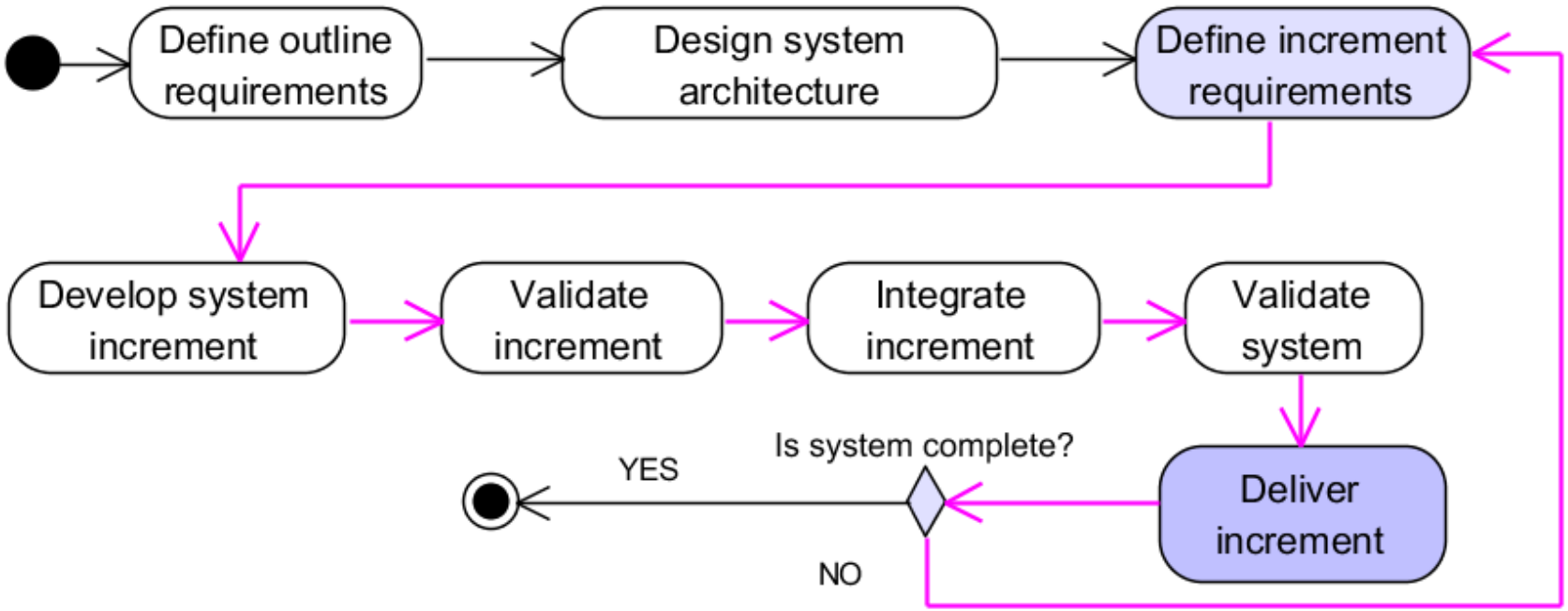


Incremental delivery
Agile approach

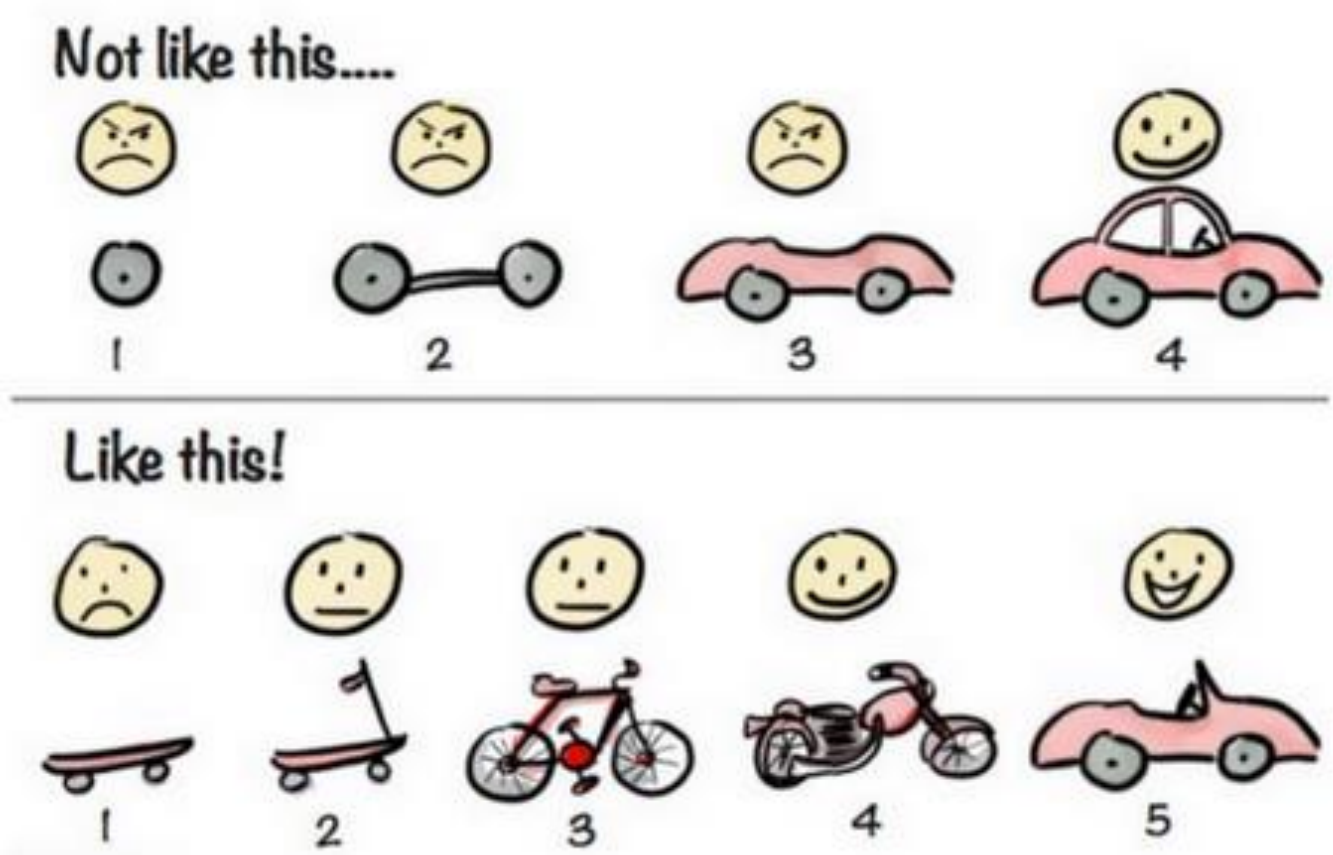


Incremental development

Incremental delivery



Incremental development and delivery



https://twitter.com/michelnadeau7/status/527798932269973504

Advantages of incremental delivery

- Accelerated delivery of customer services.

Customer value can be delivered with each increment so system functionality is available earlier. Moreover, each increment delivers the highest priority functionality to the customer.

- User engagement with the system.

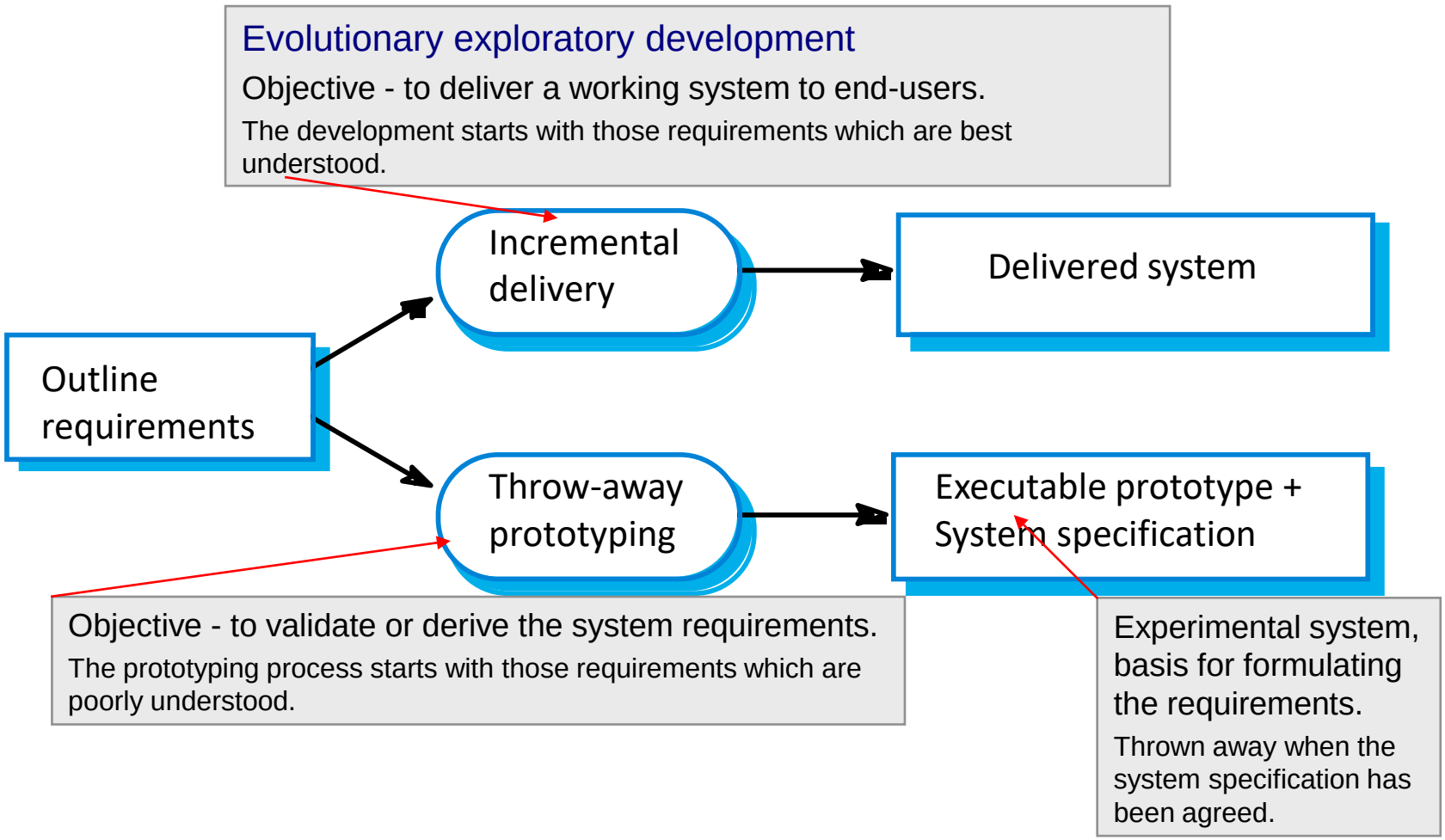
Users have to be involved in the development, which means the system is more likely to meet their requirements and the users are more committed to the system.

- Software process improvement.
  - Early increments act as a prototype to help elicit requirements for later increments.
  - Lower risk of overall project failure.
  - The highest priority system services tend to receive the most testing.

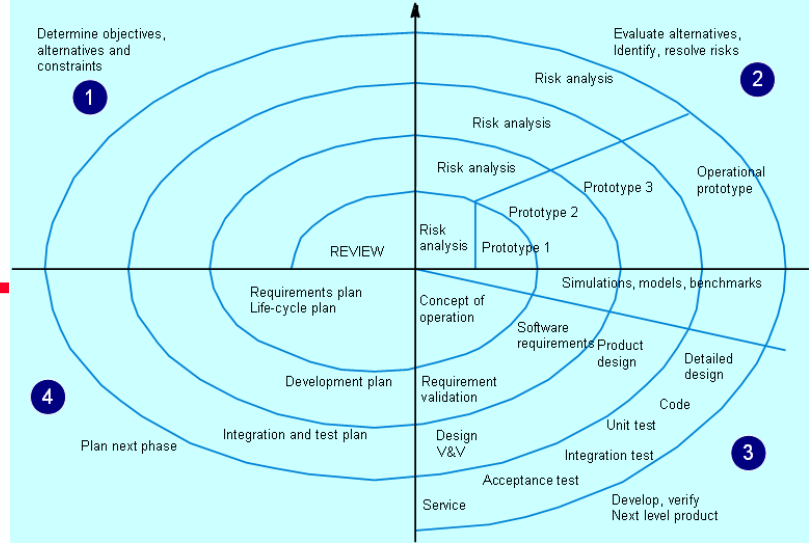
Problems of incremental delivery

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Incremental delivery and throw-away prototyping



Spiral development (Bohem)



- Process is represented as a spiral

(rather than as a sequence of activities with backtracking).

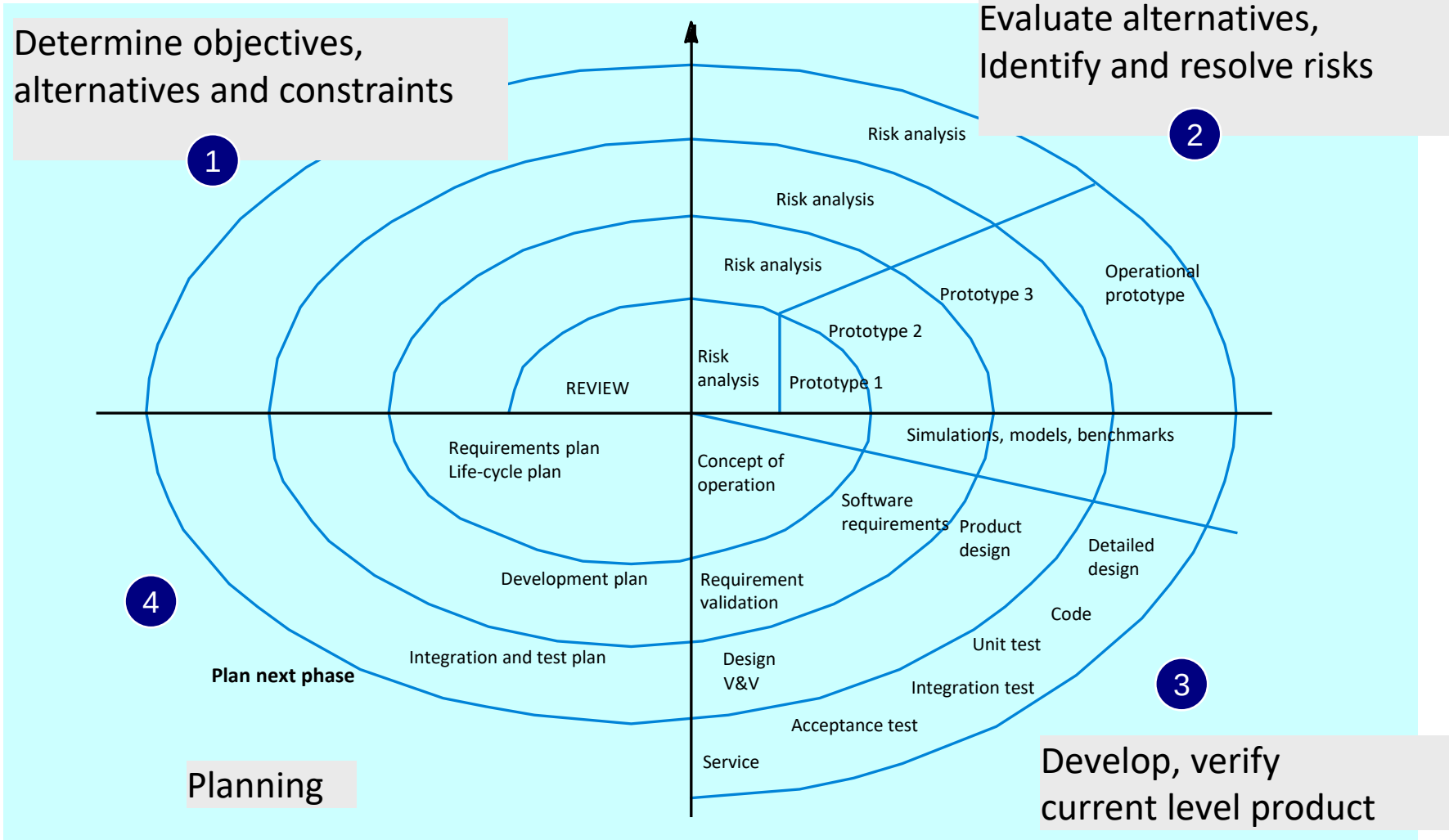
- Each loop in the spiral represents a phase in the process.

Ex.
Loop 1 (innermost) - system feasibility
Loop 2 - requirements definition
Loop 3 - system design,
etc.

- No fixed phases (like specification or design) – loops in the spiral are chosen depending on what is required.

- Risks are explicitly assessed and resolved throughout the process.

Spiral model of the software process



Spiral model sectors

Each loop in the spiral is split into four sectors:

1. Objective setting
   - Specific objectives for the phase are identified.
2. Risk assessment and reduction
   - Risks are assessed and activities put in place to reduce the key risks.
3. Development and validation
   - A development model for the system is chosen, which can be any of the generic models.
4. Planning
   - The project is reviewed and the next phase of the spiral is planned.

Formative evaluation

1. Realize the correct mapping between the activity diagram and what is represented on it.

2. Explain why incremental delivery implies that the highest priority system services tend to receive the most testing ?

https://forms.gle/dJzcFavKJLN74WfUA

Topics covered

- Generic models for the software process
- Coping with change
- **RUP (Rational Unified Process)**
- Risk management

UP (Unified Process)

Modern model for the software process.

Features:

- Extensible framework which can be customized for different organizations and projects.
- Iterative and incremental
- Use case driven
- Architecture centric
- Focused on risk

Representative refinements and variations:

- Rational UP (Basic UP, Enterprise UP)
- Open UP
- Agile UP
- Oracle Unified Method

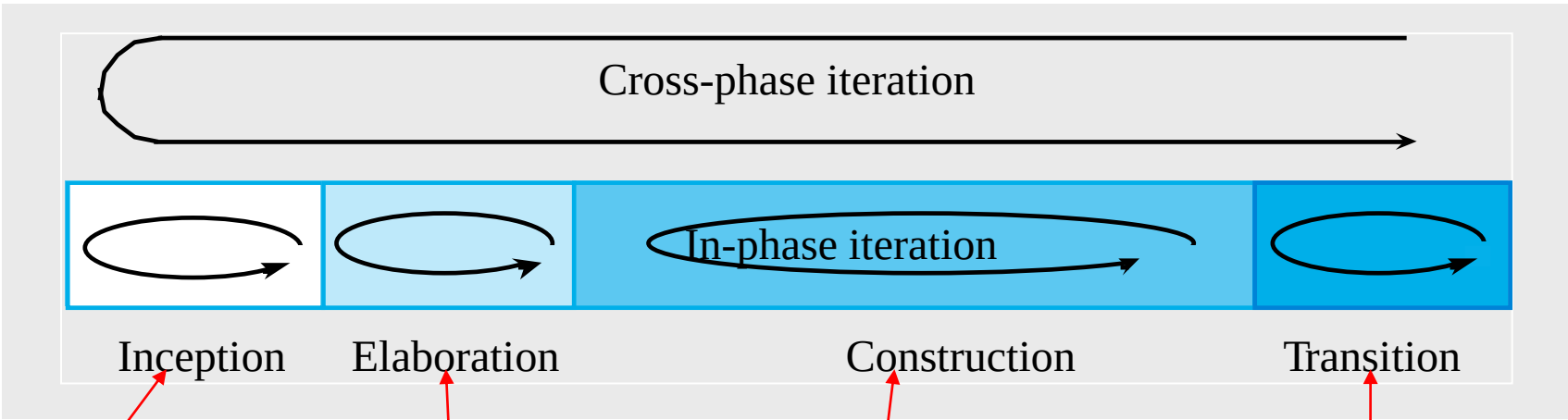
The Rational Unified Process

RUP – IBM Rational Unified Process derived from the work on the UML and associated process.

Normally described from 3 perspectives:

- ***static*** perspective - shows process activities;
- ***dynamic*** perspective - shows phases over time;
- ***practice*** perspective - suggests good practice.

RUP phase model



Establish the business case for the (sub)system.

Develop an understanding of the problem domain and the (sub)system architecture.

(sub)System design, programming and testing.

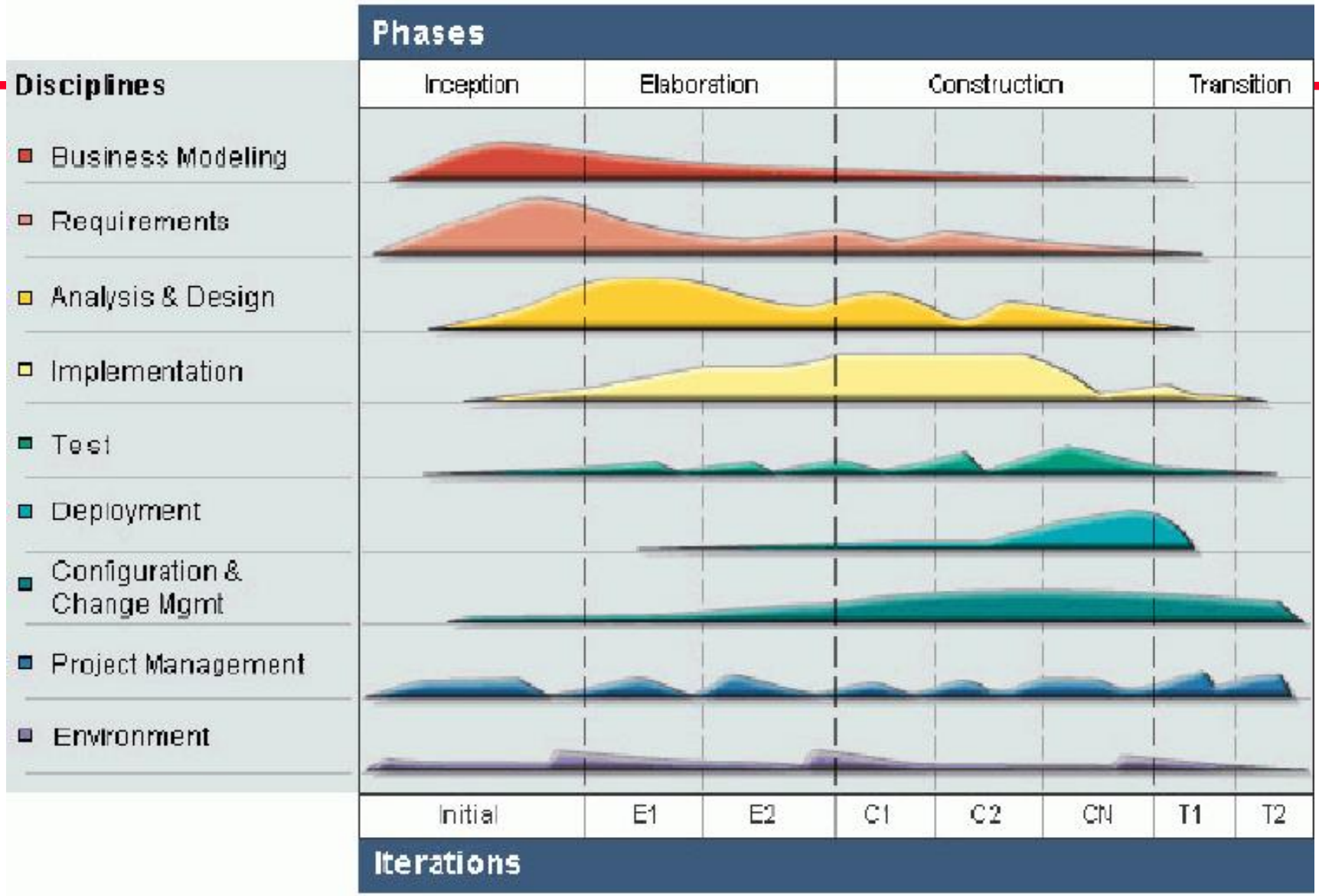
Deploy the (sub)system in its operating environment.

RUP: Activities

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow manages changes to the system.
Project management	This supporting workflow manages the system development.
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Relative weight of activities during the software process

www.ibm.org/developerworks



Control of the RUP activities

Based on criteria well defined in advance.

Input criteria: - to initiate the activity

- Needed artifacts
- Needed personnel
- Needed tools
- Activity description

Output criteria: - to declare the activity complete

- Produced artifacts
- Conditions (ex. All artefacts are revied, all (x%) error are fixed, etc.)

RUP good practice

Develop software iteratively

Plan increments based on customer priorities and deliver highest priority increments first.

Manage requirements

Explicitly document customer requirements and keep track of changes to these requirements.

Use component-based architectures

Organize the system architecture as a set of reusable components.

RUP good practice

Visually model software

Use graphical UML models to present static and dynamic views of the software.

Verify software quality

Ensure that the software meets organizational quality standards.

Control changes to software

Manage software changes using a change management system and configuration management tools.

Topics covered

- Generic models for the software process
- Coping with change
- **RUP (Rational Unified Process)**
- Risk management

Risk management

Risk management is concerned with *identifying* risks and drawing up *plans to minimise* their effect on a project.

A risk is a probability that some adverse (or favourable) circumstance will occur :

Project risks affect schedule or resources;

Product risks affect the quality or performance of the software being developed;

Business risks affect the organisation developing or procuring the software.

Software risks - examples

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

•Risk identification
•Risk analysis
•Risk planning
•Risk monitoring

The risk management process

Risk identification

Identify project, product and business risks;

Risk analysis

Assess the likelihood and consequences of these risks;

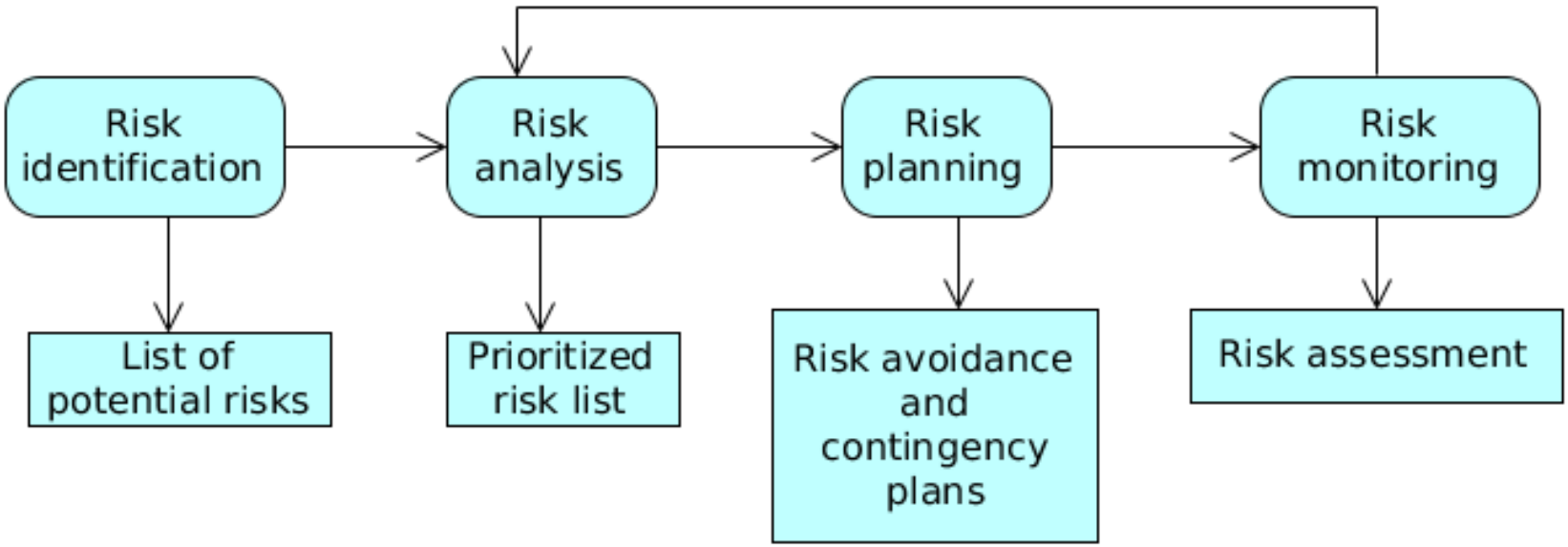
Risk planning

Draw up plans to avoid or minimise the effects of the risk;

Risk monitoring

Monitor the risks throughout the project;

The risk management process



Risk identification

Types of risks:

Technology risks.

People risks.

Organisational risks.

Tools risks.

Requirements risks.

Estimation risks.

Risks and risk types

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)

Risk analysis

Assess probability and effects of each risk.

Probability may be very low, low, moderate, high or very high.

Risk ***effects*** might be catastrophic, serious, tolerable or insignificant.

Risk analysis (1)
Example

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (7).	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project (3).	High	Catastrophic
Key staff are ill at critical times in the project (4).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused. (2).	Moderate	Serious
Changes to requirements that require major design rework are proposed (10).	Moderate	Serious
The organization is restructured so that different management are responsible for the project (6).	High	Serious
The database used in the system cannot process as many transactions per second as expected (1).	Moderate	Serious

Risk analysis (2)
Example

Risk	Probability	Effects
The time required to develop the software is underestimated (12).	High	Serious
Software tools cannot be integrated (9).	High	Tolerable
Customers fail to understand the impact of requirements changes (11).	Moderate	Tolerable
Required training for staff is not available (5).	Moderate	Tolerable
The rate of defect repair is underestimated (13).	Moderate	Tolerable
The size of the software is underestimated (14).	High	Tolerable
Code generated by code generation tools is inefficient (8).	Moderate	Insignificant

Risk planning

Consider each risk and develop a strategy to manage that risk.

Main strategies:

Avoidance strategies

Reduce the *probability* that the risk will arise;

Minimisation strategies

Reduce the *impact* of the risk on the project or product;

Contingency plans

If the risk arises, contingency plans are plans to deal with that risk;

Risk management strategies (1)
Examples

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

Risk management strategies (2)
Examples

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Risk monitoring

Assess each identified risks regularly to decide whether or not it is becoming less or more *probable*.

Also *assess* whether the *effects* of the risk have changed.

Each key risk should be discussed at management progress meetings.

Risk indicators

Risk type	Potential indicators
Technology	Late delivery of hardware or support software; many reported technology problems.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Organizational	Organizational gossip; lack of action by senior management.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.
Requirements	Many requirements change requests; customer complaints.
Estimation	Failure to meet agreed schedule; failure to clear reported defects.

Formative evaluation

1. What is repezented on this diagram ?

2. Elaborate on the building and maintaining the prioritized risks list.

https://forms.gle/Dbh7rzYfNodDJxND8

Key points

- Software process is a structured set of the activities involved in producing a software system. A software process model is an abstract representation of the software process.
- Generic software process models include the ‘waterfall’ model, incremental development, and reuse-oriented development.
- Coping with change may involve a prototyping phase that helps avoid poor decisions on requirements and design.
- Processes may be structured for incremental development and delivery so that changes may be made without disrupting the system as a whole.
- The Rational Unified Process is a modern process model that is organized into phases (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.
- Risk management is concerned with identifying risks which may affect the project and planning to ensure that these risks do not develop into major threats.

Waterfall and Agile

https://www.theserverside.com/opinion/Why-the-Waterfall-or-Agile-debate-will-be-around-forever?track=NL-1839&ad=919881&src=919881&asrc=EM_NLN_92368225&utm_medium=EM&utm_source=NLN&utm_campaign=20180319_Waterfall%2520vs.%2520Agile:%2520The%2520never-ending%2520debate