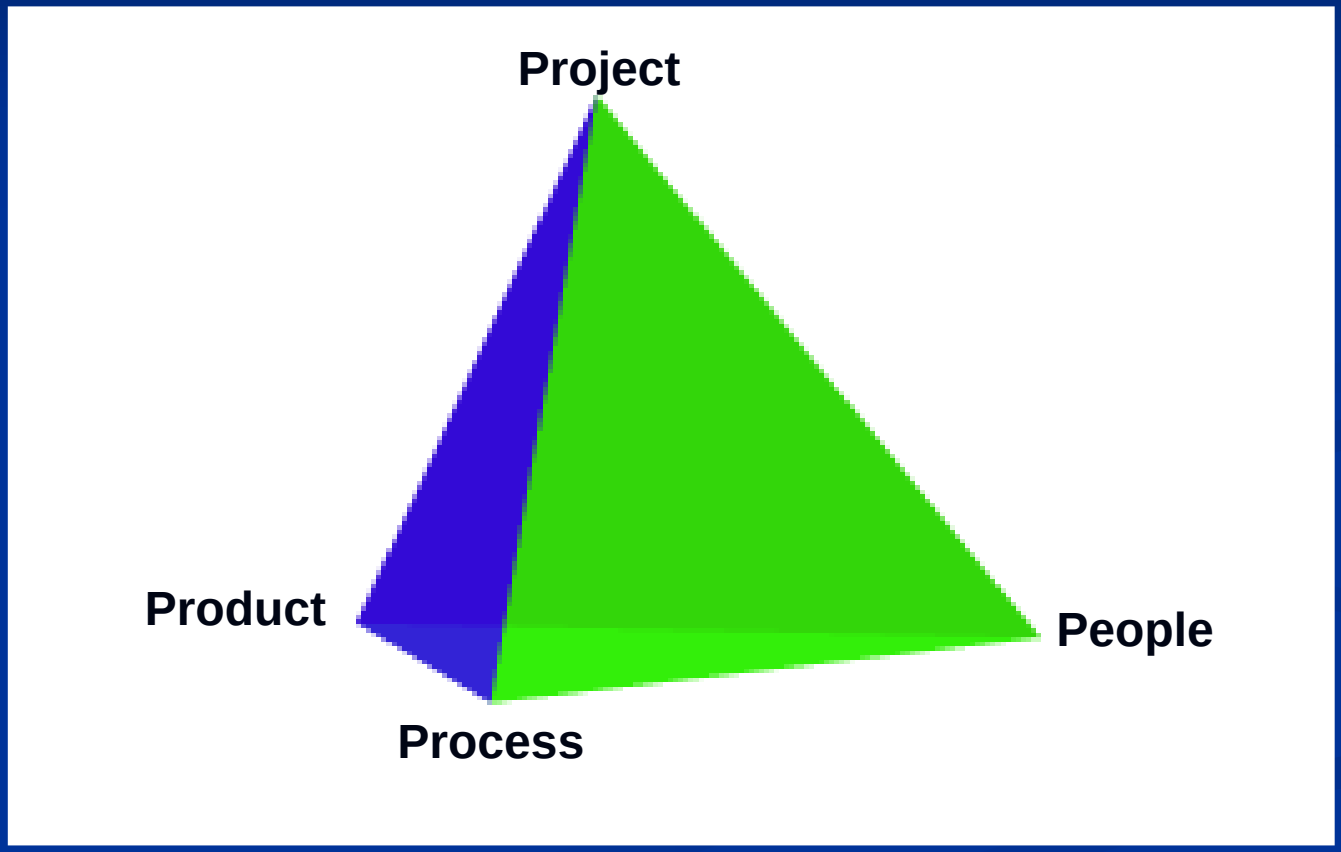
Introduction to Software Engineering

Perspective: Software Engineer and Project Management

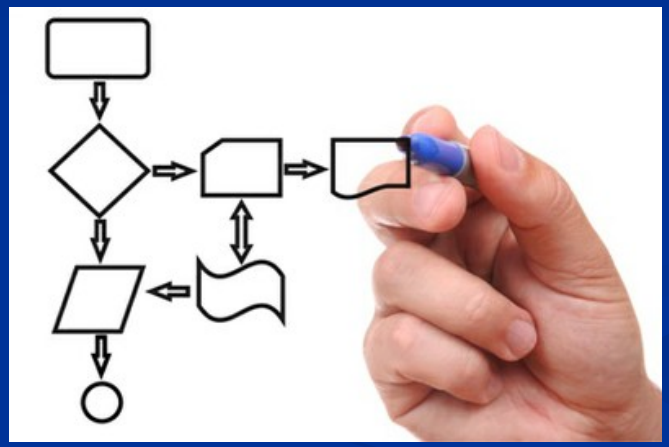
- What is Software Engineer?
- What is Software Project Management?
- Why is Software Development Process?
- Software Quality:
  - Quality of Product
  - Quality of Process

The 4 Ps



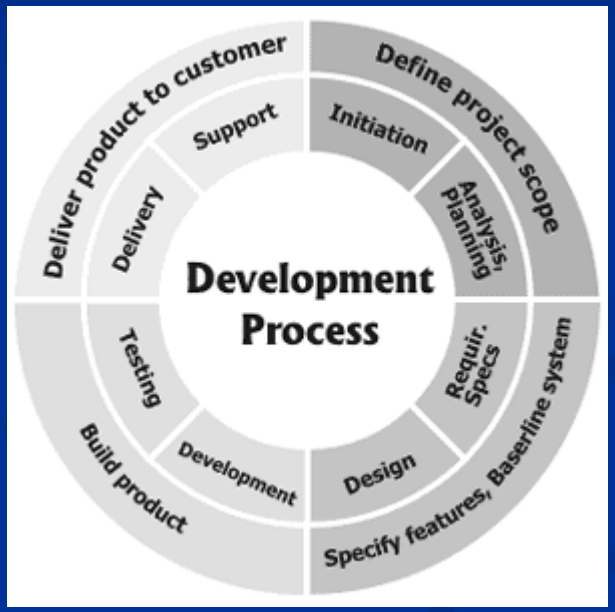
The 4 Ps

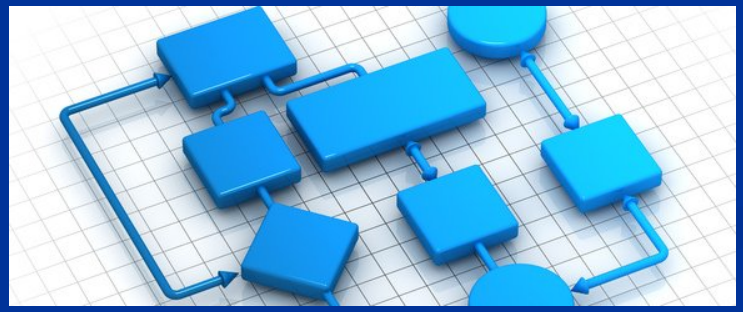
- People



The 4 Ps

- Process



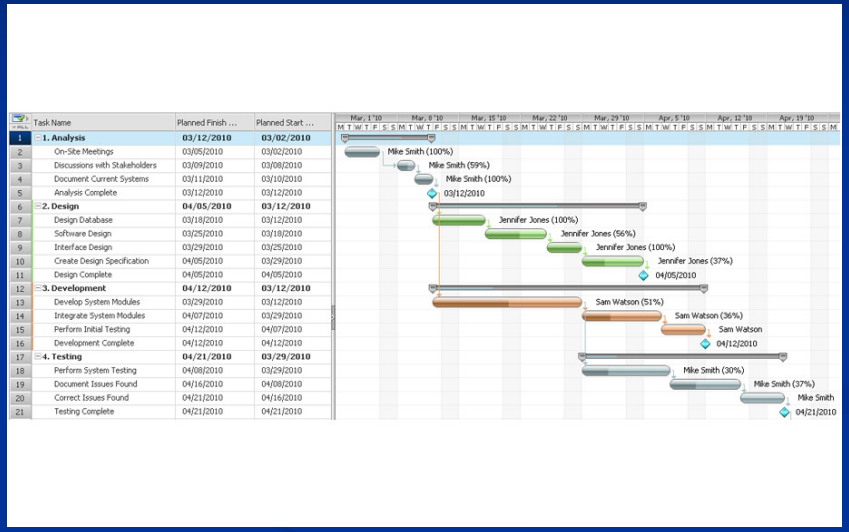


The 4 Ps

- Product

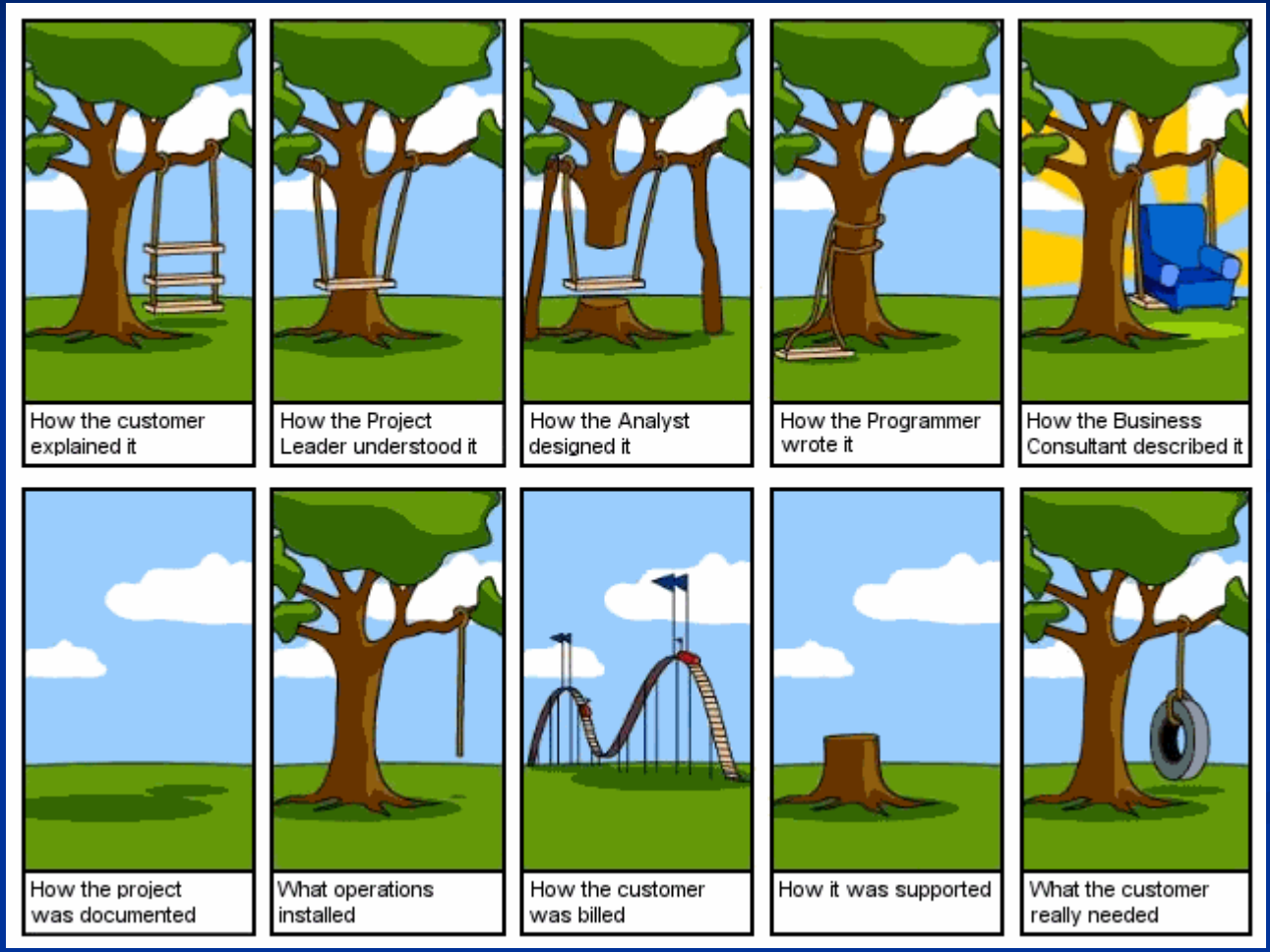
The 4 Ps

- Project



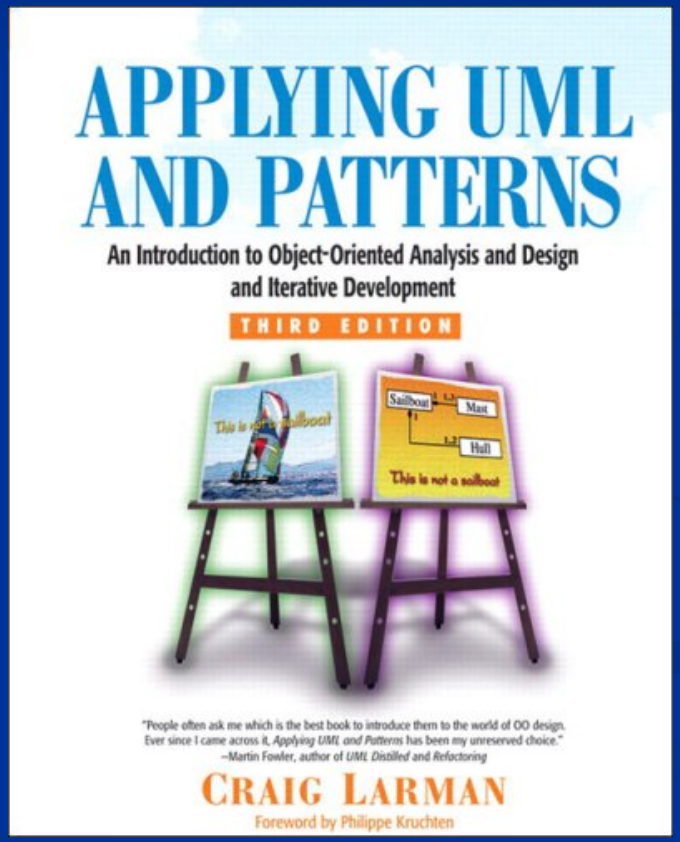
	Task Name	Planned Finish ...	Planned Start ...
1	**1. Analysis**	**03/12/2010**	**03/02/2010**
2	On-Site Meetings	03/05/2010	03/02/2010
3	Discussions with Stakeholders	03/09/2010	03/08/2010
4	Document Current Systems	03/11/2010	03/10/2010
5	Analysis Complete	03/12/2010	03/12/2010
6	**2. Design**	**04/05/2010**	**03/12/2010**
7	Design Database	03/18/2010	03/12/2010
8	Software Design	03/25/2010	03/18/2010
9	Interface Design	03/29/2010	03/25/2010
10	Create Design Specification	04/05/2010	03/29/2010
11	Design Complete	04/05/2010	04/05/2010
12	**3. Development**	**04/12/2010**	**03/12/2010**
13	Develop System Modules	03/29/2010	03/12/2010
14	Integrate System Modules	04/07/2010	03/29/2010
15	Perform Initial Testing	04/12/2010	04/07/2010
16	Development Complete	04/12/2010	04/12/2010
17	**4. Testing**	**04/21/2010**	**03/29/2010**
18	Perform System Testing	04/08/2010	03/29/2010
19	Document Issues Found	04/16/2010	04/08/2010
20	Correct Issues Found	04/21/2010	04/16/2010
21	Testing Complete	04/21/2010	04/21/2010

The Gap


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

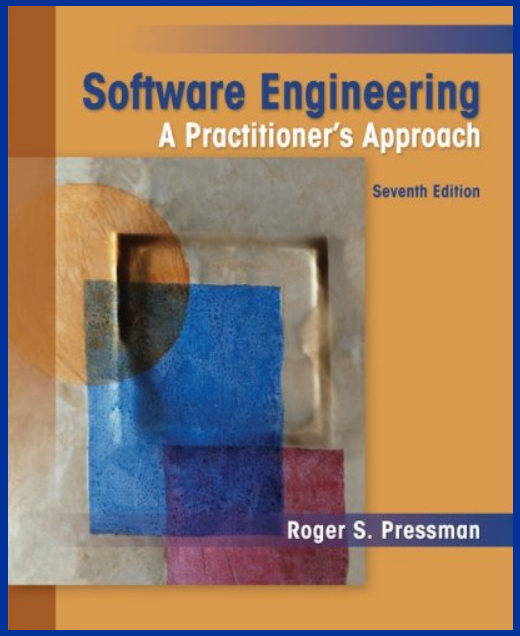
Textbook and Recommended References

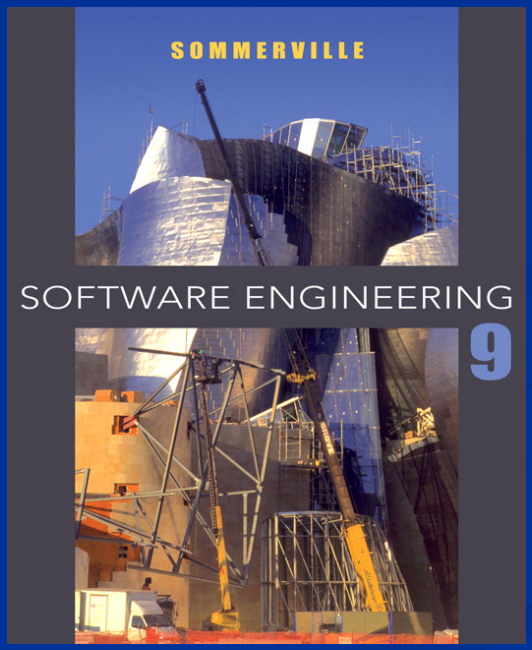
- Required book

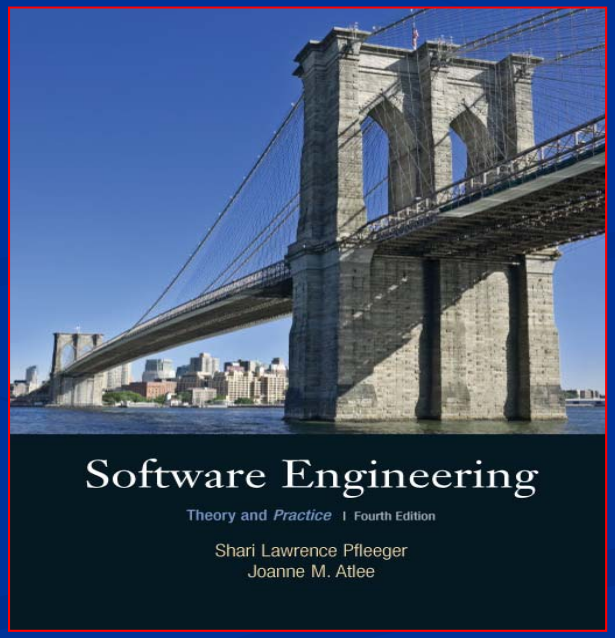


Textbook and Recommended References

- Recommended books/references







Building Software Products

Before the builder starts the development of a house, the blueprints must be approved and ready to be used

Do we have the equivalent of the blueprints in the software industry?

Building Software Products

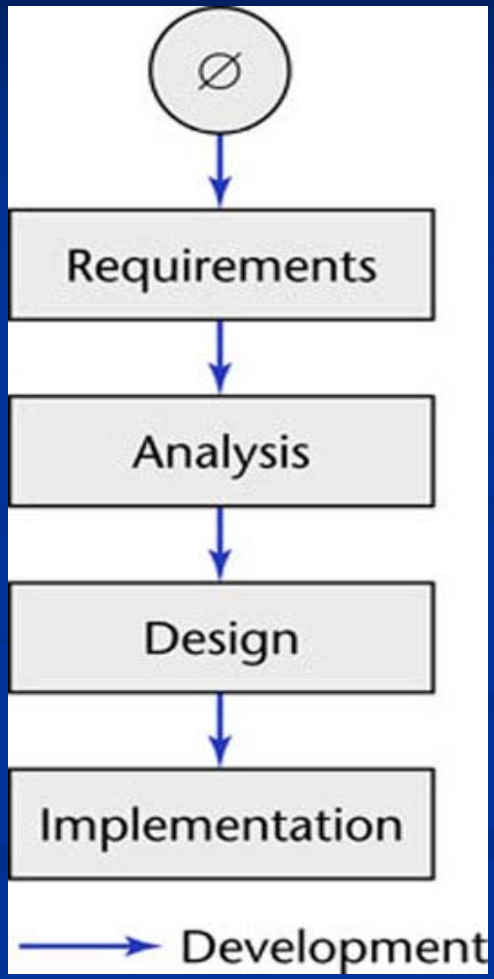
You can't start building the 3rd floor until you are done with the 2nd floor

Software Products are different

- Phases
- Features

Software Development in Theory

- Ideally, software is developed Linear



Software Development in Practice

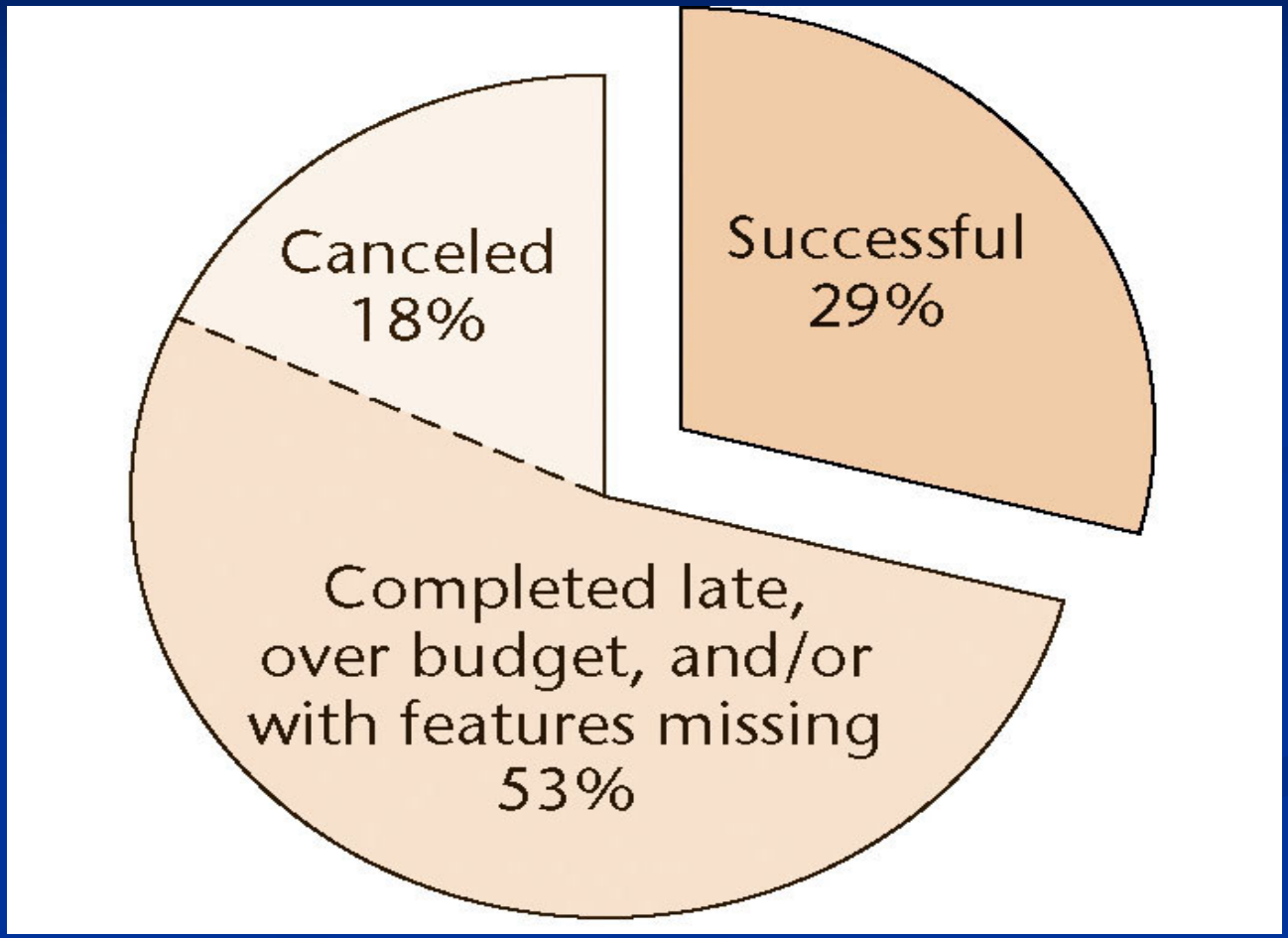
- In the real world, software development is totally different
  - We make mistakes
  - The client's requirements change while the software product is being developed

Historical Aspects

- In the belief that software design, implementation, and maintenance could be put on the same footing as traditional engineering disciplines, in 1967 a NATO study group coined the term ***software engineering***
- The term endorsed in 1968 NATO Conference, Garmisch, Germany
- Aim: To solve the *software crisis*
- Software is delivered
  - Late
  - Over budget
  - With residual faults

Standish Group Data

- Data on 9236 projects completed in 2004



Economic Aspects

- Coding method CM_{new} is 10% faster than currently used method CM_{old}. Should it be used?
- Common sense answer
  - Of course!
- Software Engineering answer
  - Consider the cost of training
  - Consider the impact of introducing a new technology
  - Consider the effect of CM_{new} on maintenance

Maintenance Aspects

- Life-cycle model
  - The steps (*phases*) to follow when building software
  - A theoretical description of what should be done
- Life cycle
  - The actual steps performed on a specific product

Waterfall Life-Cycle Model

- Classical model (1970)

1. Requirements phase
2. Analysis (specification) phase
3. Design phase
4. Implementation phase
5. Postdelivery maintenance
6. Retirement

Figure 1.2

Typical Classical Phases

- Requirements phase
  - Explore the concept
  - Elicit the client's requirements
- Analysis (specification) phase
  - Analyze the client's requirements
  - Draw up the specification document
  - Draw up the software project management plan
  - "What the product is supposed to do"

Typical Classical Phases (contd)

- Design phase
  - Architectural design, followed by
  - Detailed design
  - “How the product does it”
- Implementation phase
  - Coding
  - Unit testing
  - Integration
  - Acceptance testing

Typical Classical Phases (contd)

- Postdelivery maintenance
  - Corrective maintenance
  - Perfective maintenance
  - Adaptive maintenance
- Retirement

Classical and Modern Views of Maintenance

- Classical maintenance
  - Development-then-maintenance model
- This is a temporal definition
  - Classification as development or maintenance depends on when an activity is performed

Classical Maintenance Defn — Consequence 1

- A fault is detected and corrected one day after the software product was installed
  - Classical maintenance
- The identical fault is detected and corrected one day before installation
  - Classical development

Classical Maintenance Defn — Consequence 2

- A software product has been installed
- The client wants its functionality to be increased
  - Classical (perfective) maintenance
- The client wants the identical change to be made just before installation (“moving target problem”)
  - Classical development

Classical Maintenance Definition

- The reason for these and similar unexpected consequences
  - Classically, maintenance is defined in terms of the time at which the activity is performed
- Another problem:
  - Development (building software from scratch) is rare today
  - Reuse is widespread

Modern Maintenance Definition

- In 1995, the International Standards Organization and International Electrotechnical Commission defined maintenance *operationally*

- Maintenance is nowadays defined as
  - The process that occurs when a software artifact is modified because of a problem or because of a need for improvement or adaptation

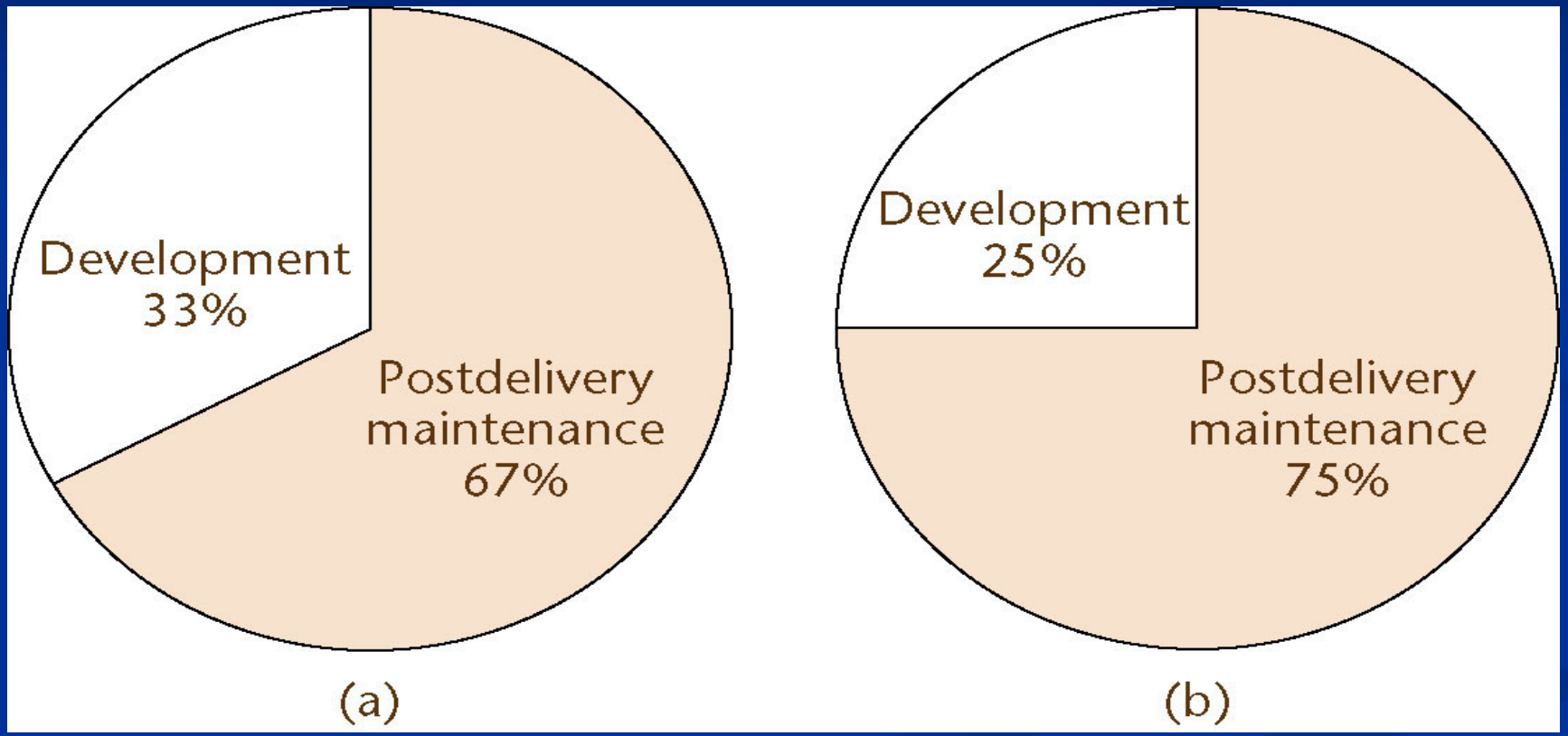
Modern Maintenance Definition (contd)

- In terms of the ISO/IEC definition
  - Maintenance occurs whenever software is modified
  - Regardless of whether this takes place before or after installation of the software product
- The ISO/IEC definition has also been adopted by IEEE and EIA

The Importance of Postdelivery Maintenance

- Bad software is discarded
- Good software is maintained, for 10, 20 years or more
- Software is a model of reality, which is constantly changing

Time (= Cost) of Postdelivery Maintenance



(a) Between 1976 and 1981

(b) Between 1992 and 1998

Consequence of Relative Costs of Phases

- Return to CT_{old} and CT_{new}
- Reducing the coding cost by 10% yields at most a 0.85% reduction in total costs
  - Consider the expenses and disruption incurred
- Reducing postdelivery maintenance cost by 10% yields a 7.5% reduction in overall costs

Requirements, Analysis, and Design Aspects

- The earlier we detect and correct a fault, the less it costs us
- Why?
  - You will have a cascade of changes for all artifacts involved
  - See next slide.

Requirements, Analysis, and Design Aspects (contd)

- To correct a fault early in the life cycle
  - Usually just a document needs to be changed
- To correct a fault late in the life cycle
  - Change the code and the documentation
  - Test the change itself
  - Perform regression testing
  - Reinstall the product on the client's computer(s)

Requirements, Analysis, and Design Aspects (contd)

- Between 60 and 70% of all faults in large-scale products are requirements, analysis, and design faults

Find Faults Early

- It is vital to improve our requirements, analysis, and design techniques
  - To find faults as early as possible
  - To reduce the overall number of faults (and, hence, the overall cost)

Planning Activities of the Classical Paradigm

- Preliminary planning of the requirements and analysis phases at the start of the project
- The Software Project Management Plan is drawn up when the specifications have been signed off by the client
- Management needs to monitor the SPMP throughout the rest of the project

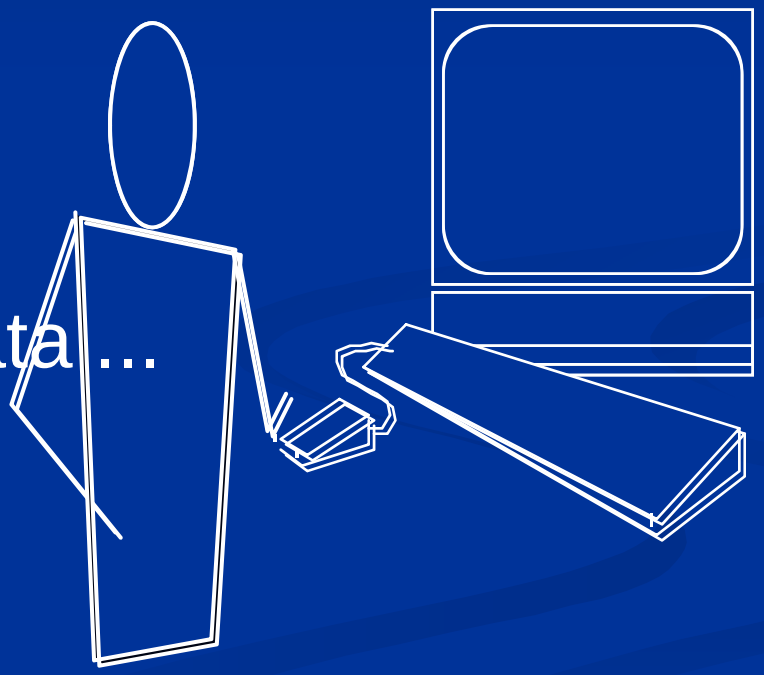
Documentation Must Always be Current

- Key individuals may leave before the documentation is complete
- We cannot perform a phase without having the documentation of the previous phase
- We cannot test without documentation
- We cannot maintain without documentation

What is Software Product?

Software Product is a set of items or objects that form a “configuration” that includes

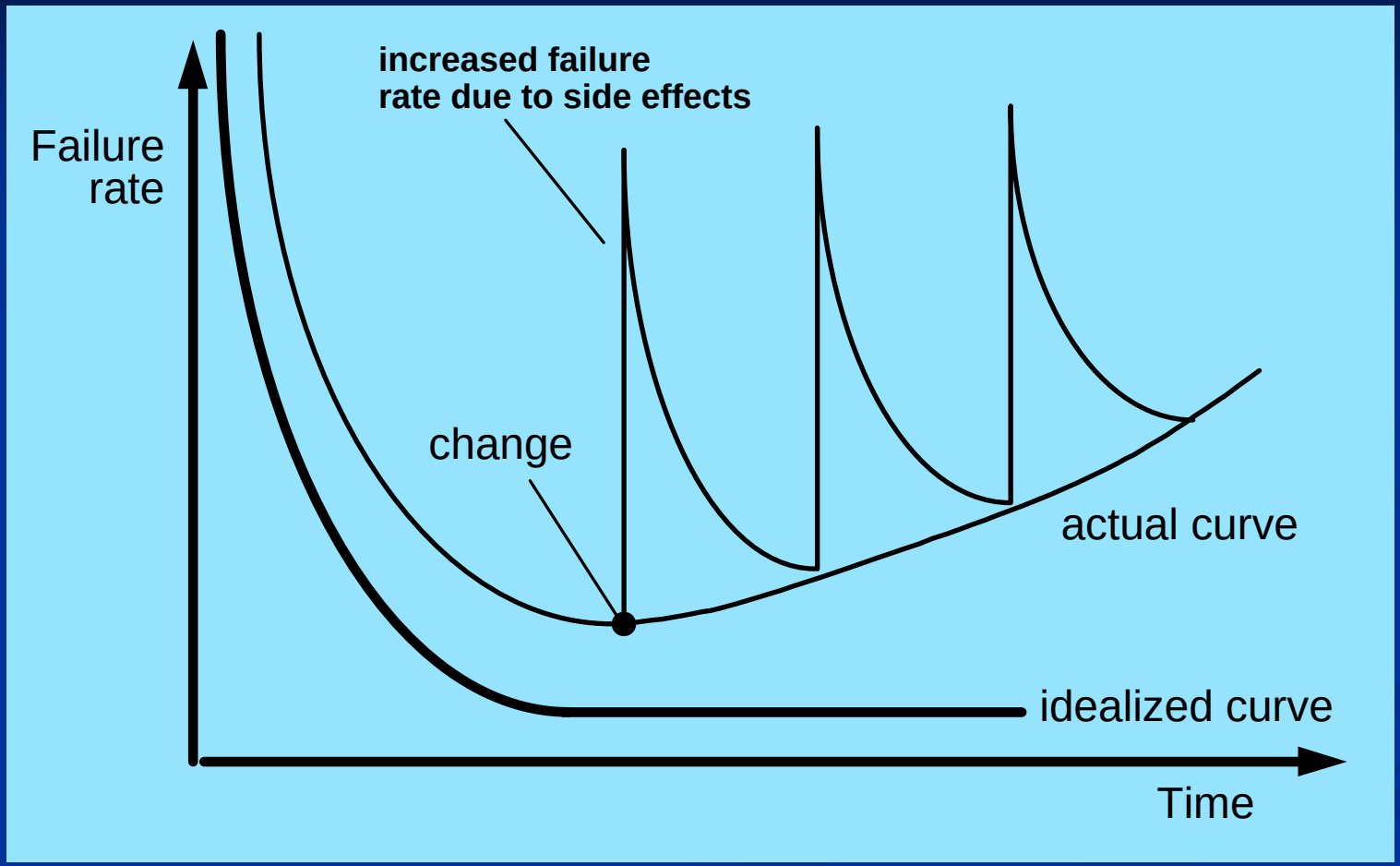
- programs
- documents
- configuration data …



What is the Software Product?

- Software product is engineered
- Software product doesn’t wear out but it does suffer from aging

Wear vs. Deterioration



Software Applications

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

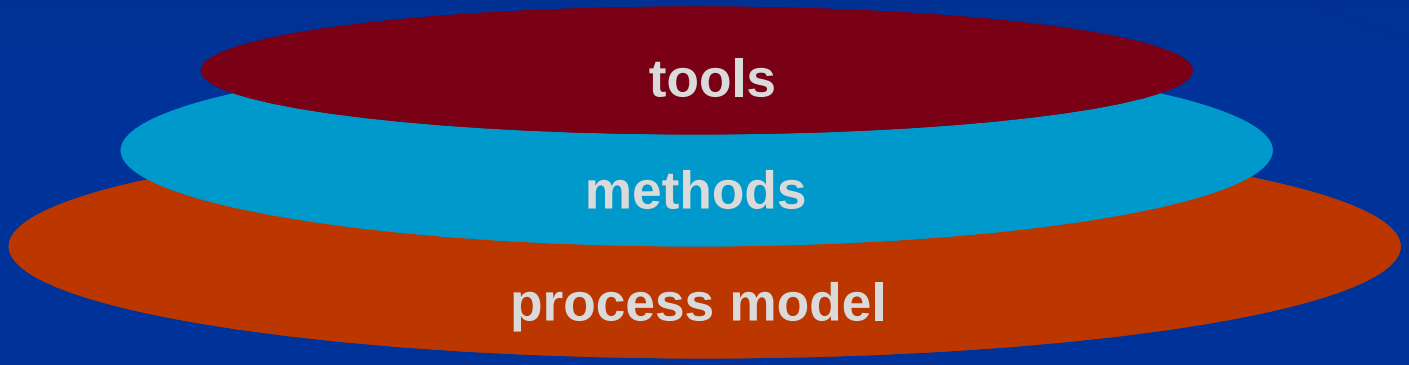
Legacy Software

Why must it change?

- software must be adapted to meet the needs of new computing environments or technology.
- software must be enhanced to implement new business requirements.
- software must be extended to make it interoperable with other more modern systems or databases.
- software must be re-architected to make it viable within a network environment.

A Layered Technology

Software Engineering



a “quality” focus

Process

- What is the software process?

A Process Framework

Process framework

- **Framework activities**
  - **work tasks**
  - **work products**
  - **milestones & deliverables**
  - **QA checkpoints**
- **Umbrella Activities**

Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

The Process Model: Adaptability

- the framework activities will always be applied on every project ... BUT
- the tasks (and degree of rigor) for each activity will vary based on:
  - the type of project
  - characteristics of the project
  - common sense judgment; concurrence of the project team

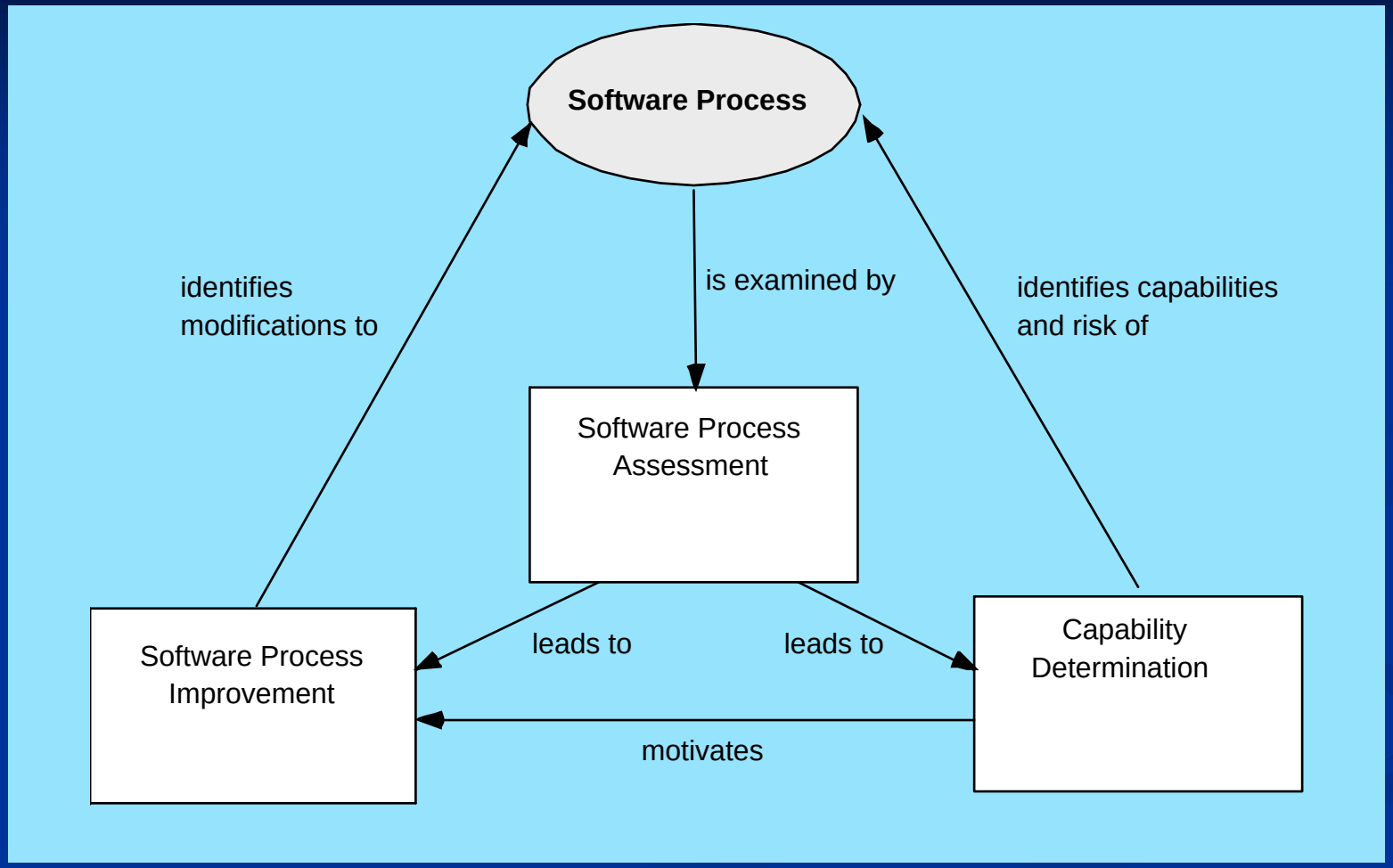
Process Patterns

- Process patterns define a set of activities, actions, work tasks, work products and/or related behaviors
- A template is used to define a pattern
- Typical examples:
  - Customer communication (a process activity)
  - Analysis (an action)
  - Requirements gathering (a process task)
  - Reviewing a work product (a process task)
  - Design model (a work product)

Process Assessment

- The process should be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful software engineering.
- Many different assessment options are available

Assessment and Improvement



The Primary Goal of Any Software Process: *High Quality*

Remember:

High quality = project timeliness

Why?

Less rework!

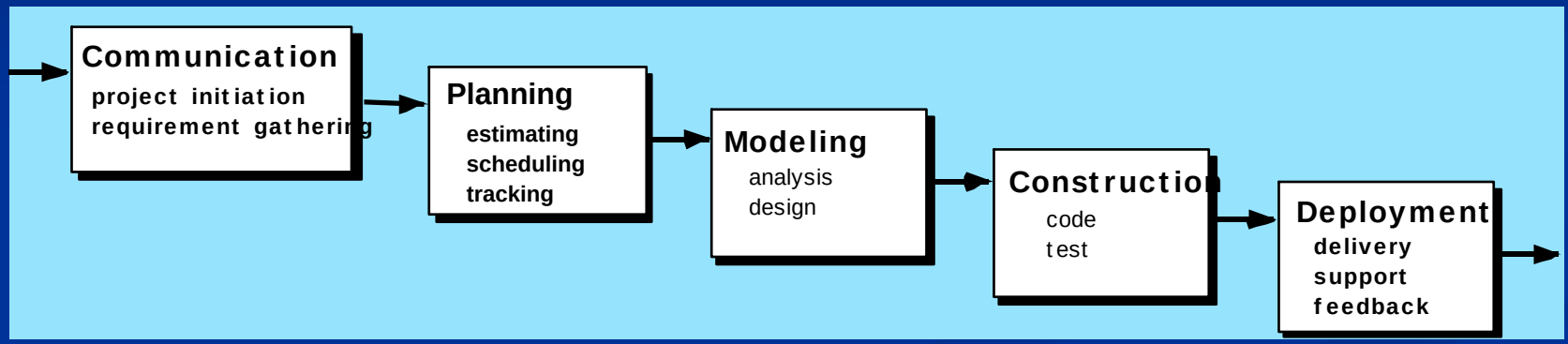
Prescriptive Process Models

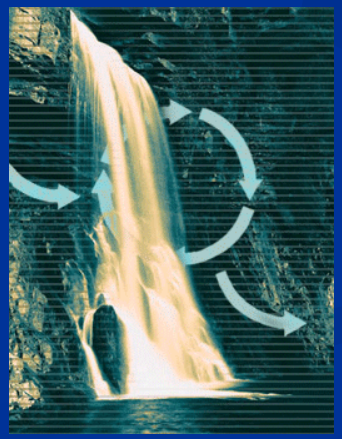
- Prescriptive process models advocate an orderly approach to software engineering

That leads to a few questions …

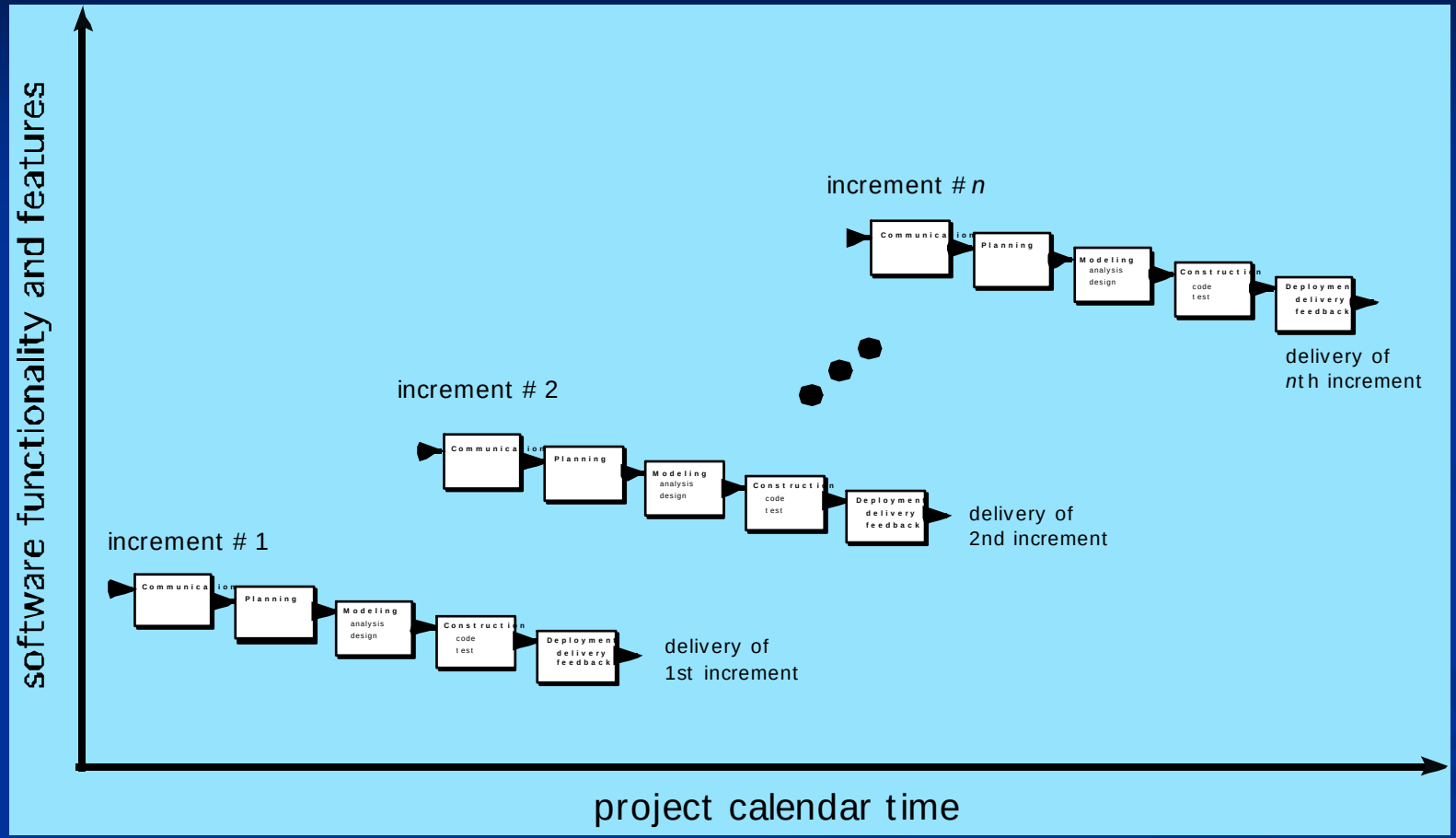
- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

The Waterfall Model



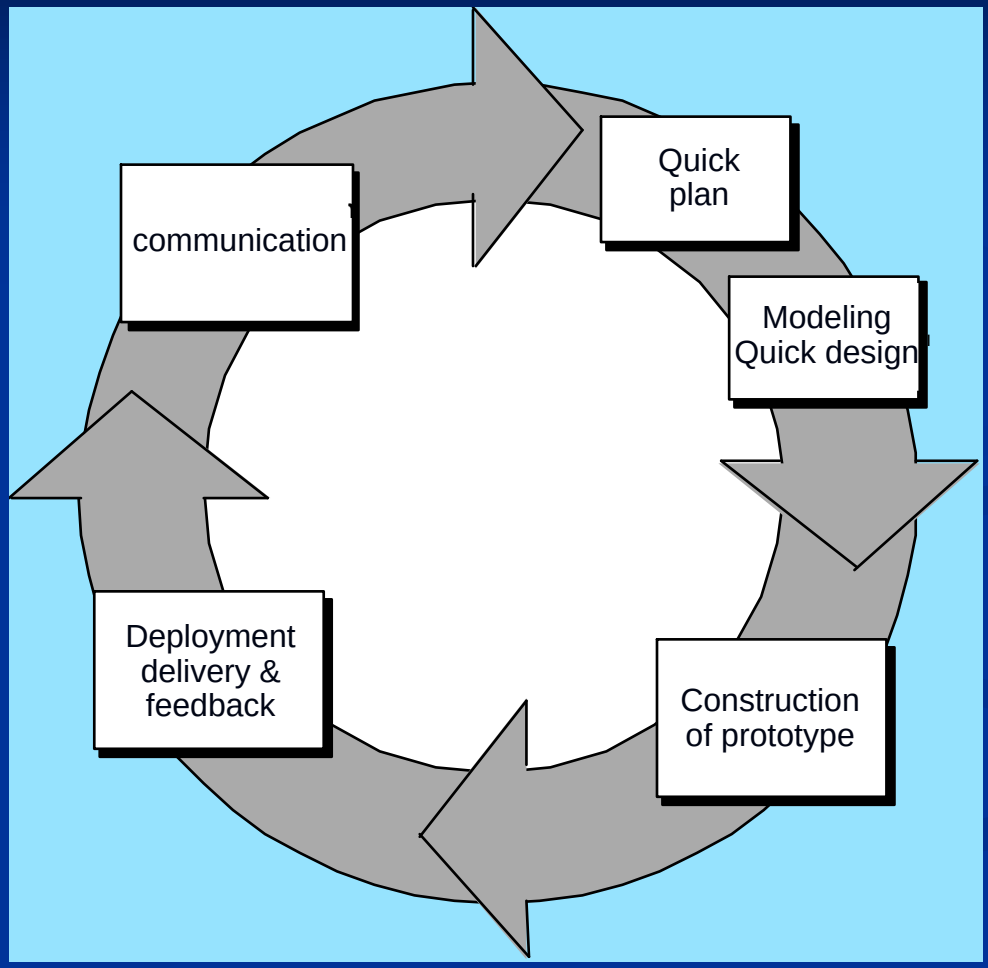


The Incremental Model



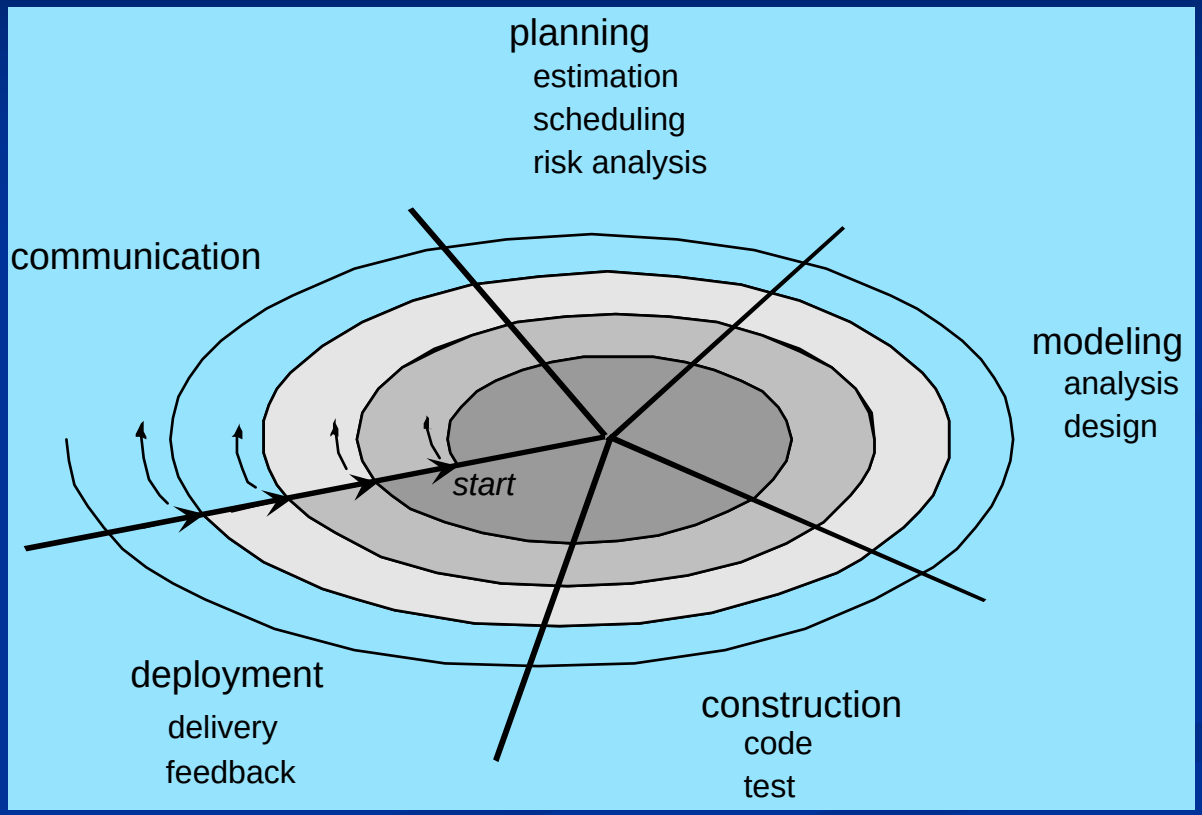
Evolutionary Models: Prototyping

Customer provides general objectives; NO detailed requirements



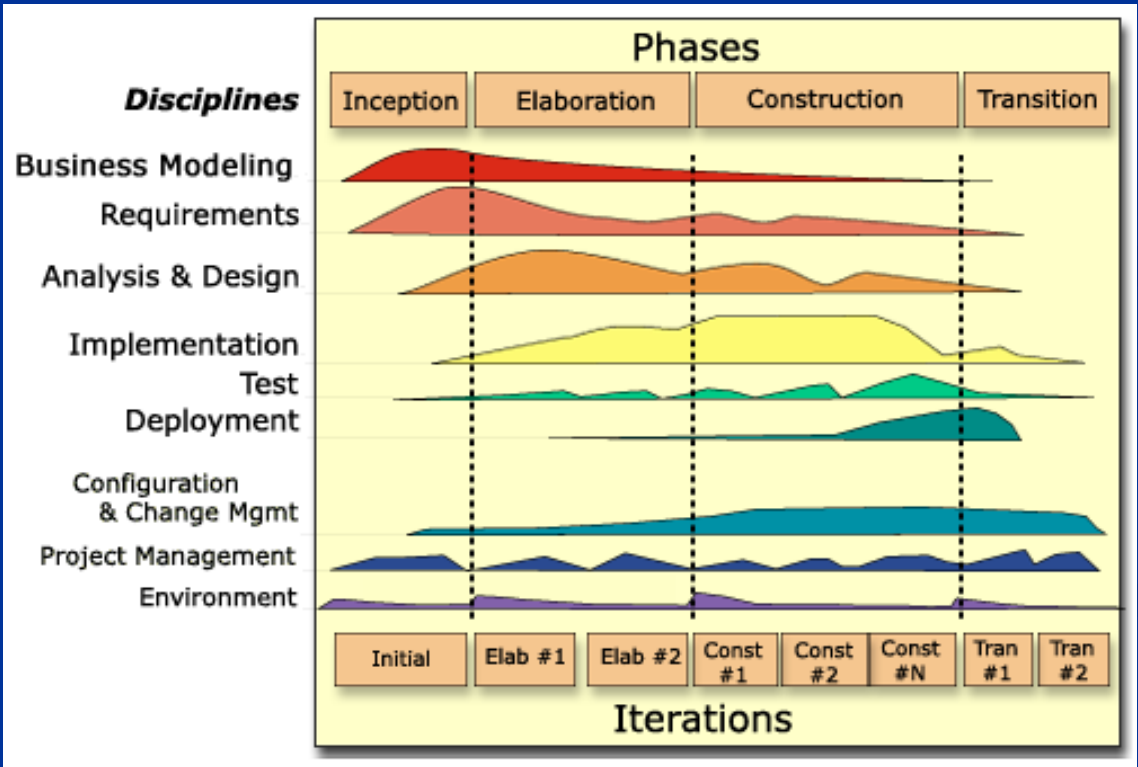
Evolutionary Models: The Spiral

Iterative "prototyping" in a controlled waterfall model

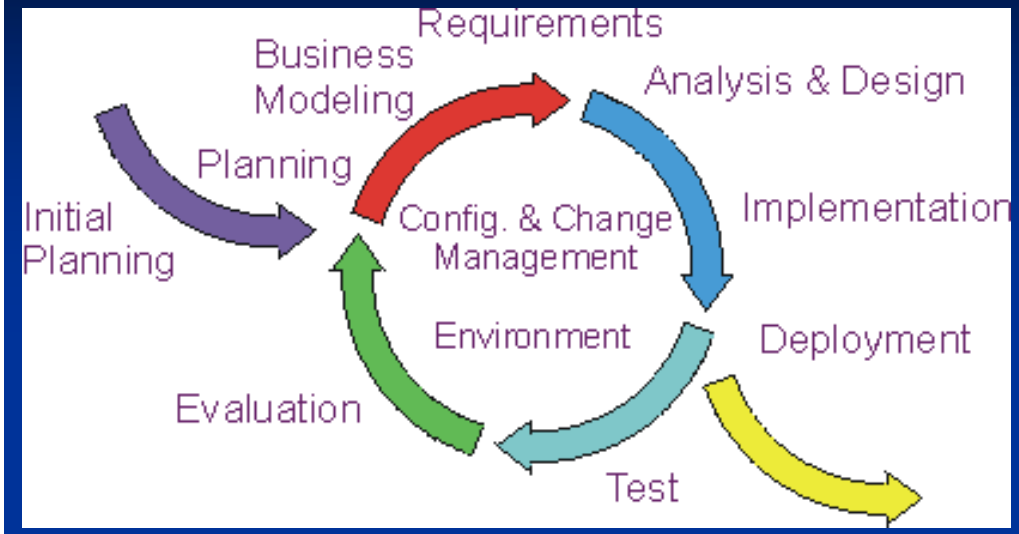


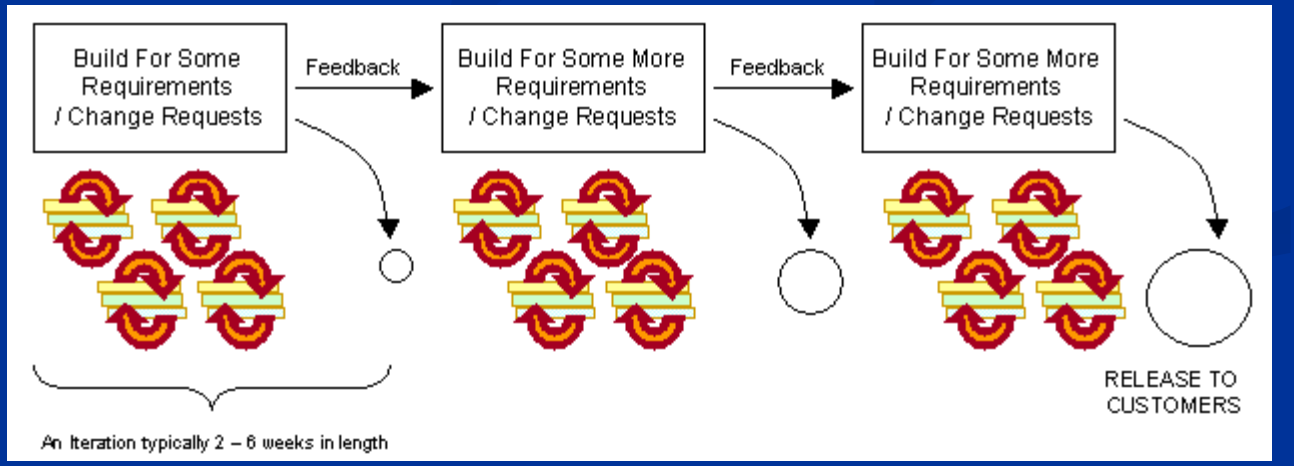
Unified Process

- Unified Process—a “use-case driven, architecture-centric, iterative and incremental” software process closely aligned with the Unified Modeling Language (UML)



The Unified Process (UP)





UP Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan,
  - phases and iterations.
- Business model,
  - if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements
  - including non-functional
- Analysis model
- Software architecture
  - Description.
- Executable architectural
  - prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software
  - increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current
    - increment

Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

The Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more."

Kent Beck et al

What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of “user stories”
  - Agile team assesses each story and assigns a cost
  - Stories are grouped for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment “project velocity” is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)