About Software Engineering

pierre-alain.muller@uha.fr

What is Software Engineering?

- Software Engineering
  - Software development
  - Engineering

- Let's have a look at ICSE
  - International Conference on Software Engineering

Challenges

- Software permeates every aspect of our society—at home and at work, in business and for pleasure, to support our daily chores, and to plan and manage our future. We increasingly expect it to be available, reliable, safe, secure, and usable, despite our own mobility, unpredictability, and changing needs.

Challenges

- The development of such software poses increasing challenges for software engineering teams, who are themselves distributed, perhaps mobile, have varied skills, and often speak varied languages. The discipline of software engineering must address these challenges through the development and refinement of new techniques, practices, and tools that build upon sound engineering principles. Moreover, the ubiquity of software means that the discipline of software engineering is also extending. A software engineering team must think of software not only as a mathematical description or a product, but also as a service, a commodity, or even as a user experience.

Challenges

- Computer-based systems continue increasingly to pervade every aspect of human activity. As this proceeds, the need to be sure that we can provide safe, efficient, high-quality software for these systems at acceptable costs in time and money, continue to assume increasing importance. The practice of software development is a worldwide enterprise, with nations and companies, both great and small, and both established and emerging, all vying for leading positions. Breakthroughs and improvements in the practice of software development, and the attendant competitive advantages they will convey, will depend pivotally on progress in fundamental software engineering research, and in effective education.

Challenges

- The theme of ICSE 2007 is **Developing Dependable Software**, with which we acknowledge the increasingly crucial role the engineering of software plays in business, healthcare, government and society at-large. The theme also highlights the growing responsibility our profession and its members are expected to assume.

Topics

- Software requirements engineering
- Software architectures and design
- Patterns and frameworks
- Software components and reuse
- Software testing and analysis
- Theory and formal methods
- Computer supported cooperative work
- Human-Computer Interaction
- Software processes and workflows
- Software dependability, safety and reliability
- Reverse engineering and software maintenance
- Software economics and metrics
- AI and Knowledge based software engineering

Topics

- Agile software development
- Empirical software engineering and metrics
- Aspect-orientation and feature interaction
- Distributed/parallel SW systems
- Embedded and real-time software
- Mobile, ubiquitous and pervasive systems
- Software tools and development environments
- SW Configuration management and deployment
- Software policy and ethics
- Software distribution licenses
- Programming languages
- Object-oriented techniques
- End user software engineering
- Internet and information systems development

Let’s start

- Lifecycle of software

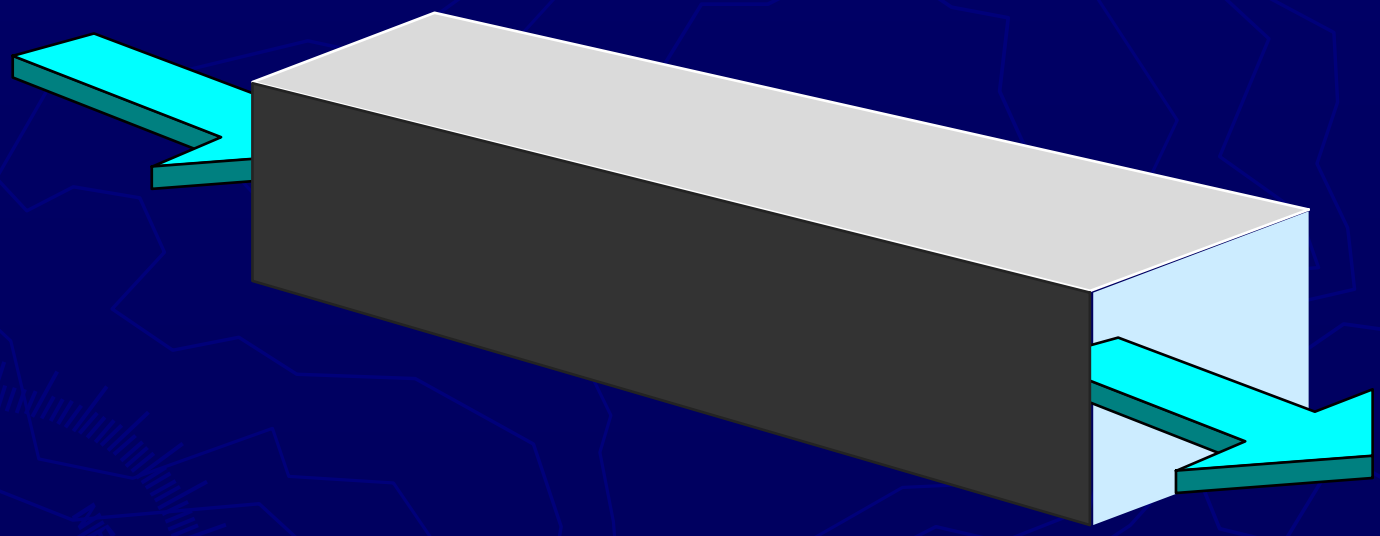
Software lifecycles

- Linear lifecycles
  - The tunnel
  - The Waterfall
  - The V
- Limits of linear lifecycles
- Iterative lifecycles

Goals

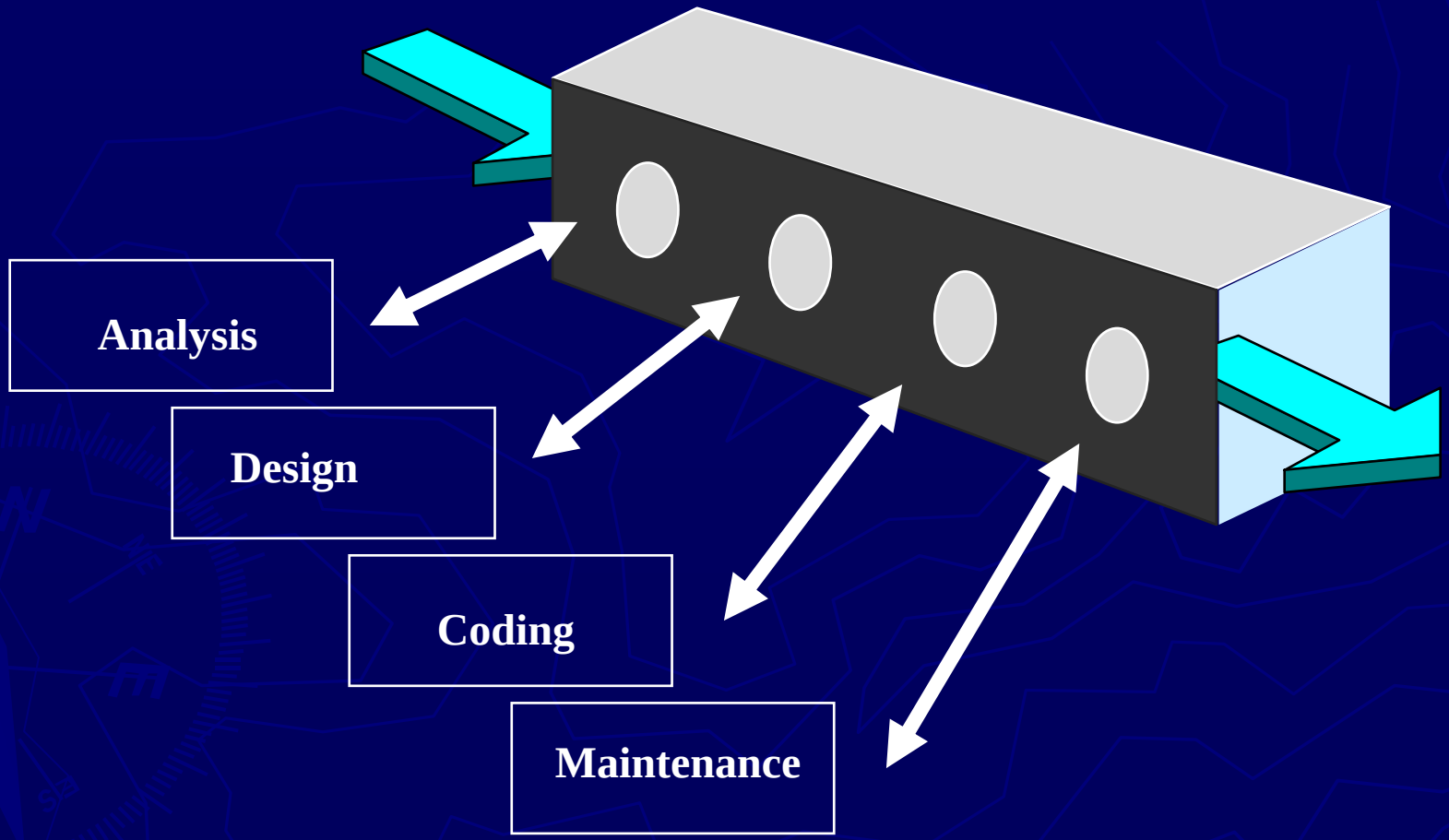
- Handle SW development all the way long
  - Master the risks
  - Master the changes
- Repeatable quality figures
- Breakdown the workload

The tunnel model

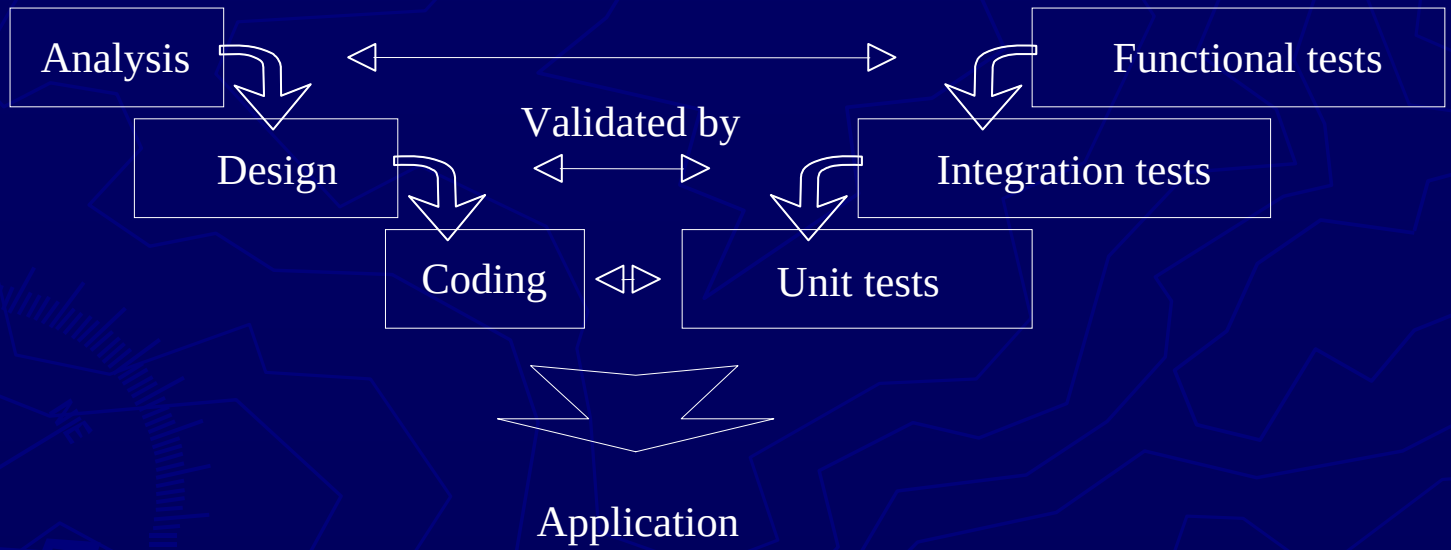


t_0 → ?

The waterfall model



The V model

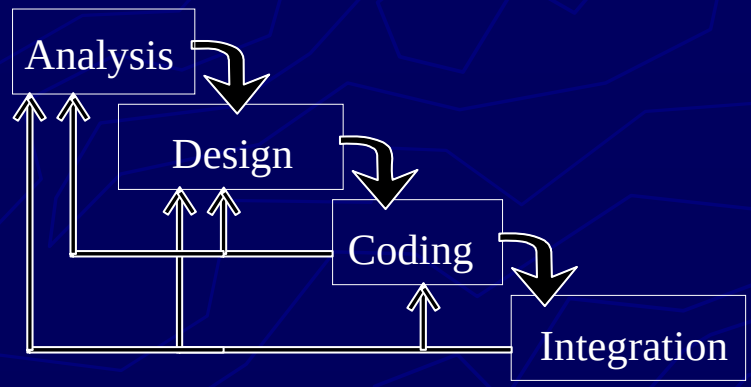

Analysis
Functional tests
Validated by
Design
Integration tests
Coding
Unit tests
Application

Features of the waterfall

- Linear, flowing down
- Limited feedback
- Documentation based
- Validated by reviews
- From requirements to implementation
- Good when requirements are well-known

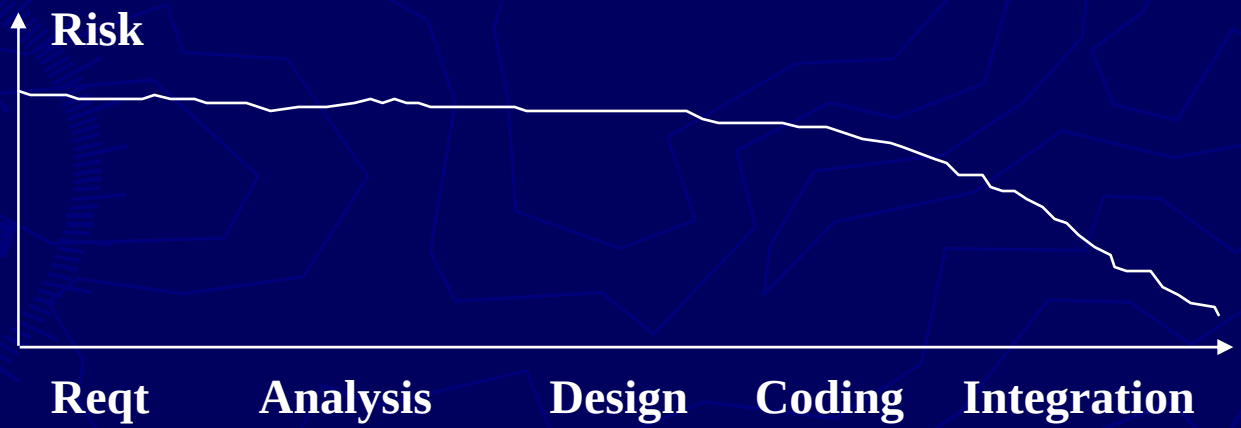
Limits of the waterfall

- Lack of knowledge of the reqts by the customer
- Misunderstanding of the reqts by the supplier
- Resulting Instability
- Technical choices
- Personnel turnover



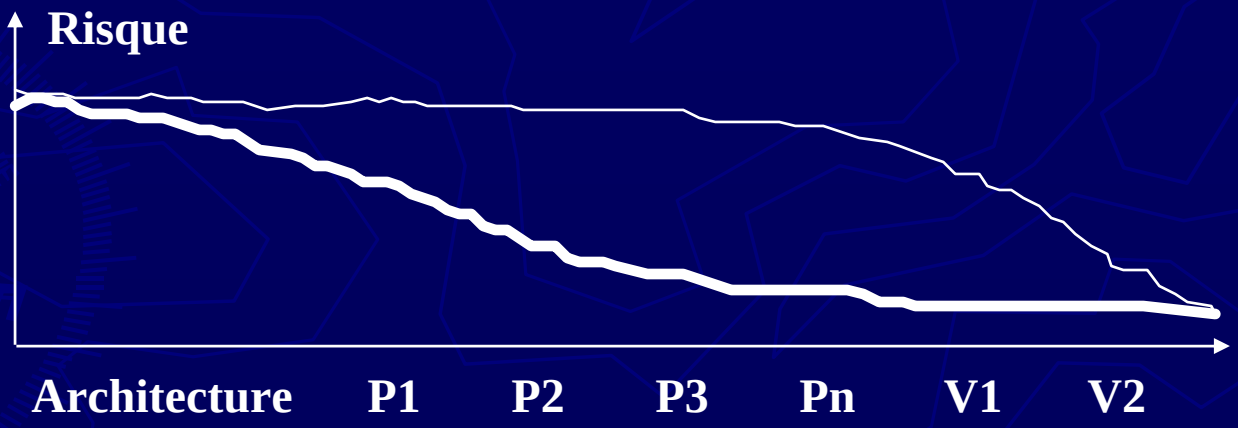
Risks and waterfall

- Late problem identification
- Late proof of satisfaction
- Reviews of documentation, not real product



Risk reduction

- Faster decrease of risks due to prototyping

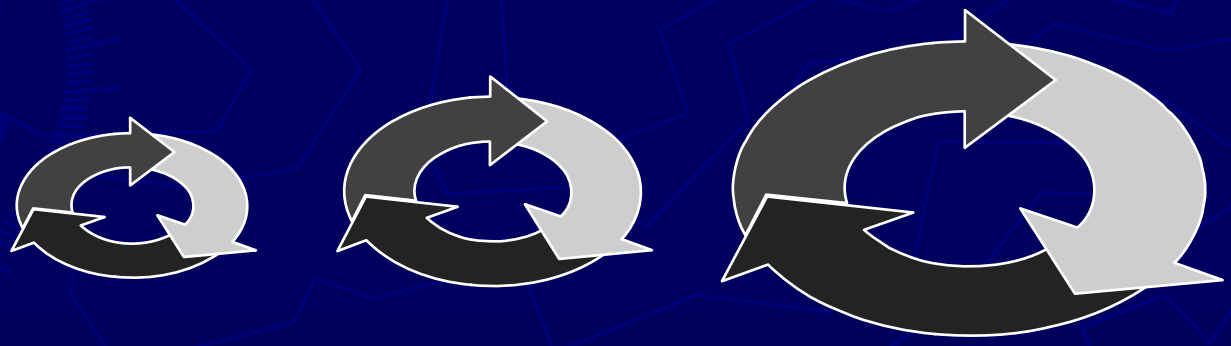


Enhancement

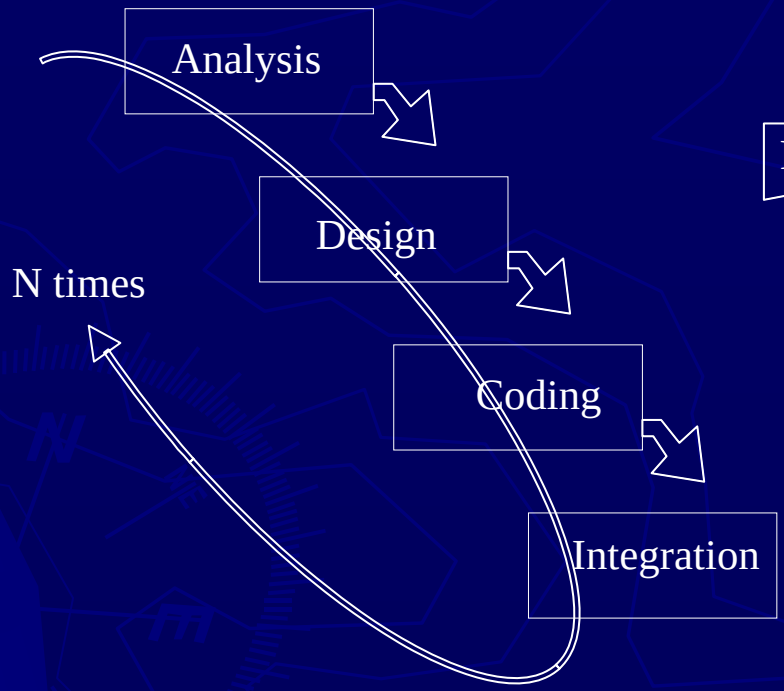
- Distinction between phases and activities
- Incremental system building
- Early proof of feasibility
- Prototyping as support of system construction
- Concrete (executable) software evaluation

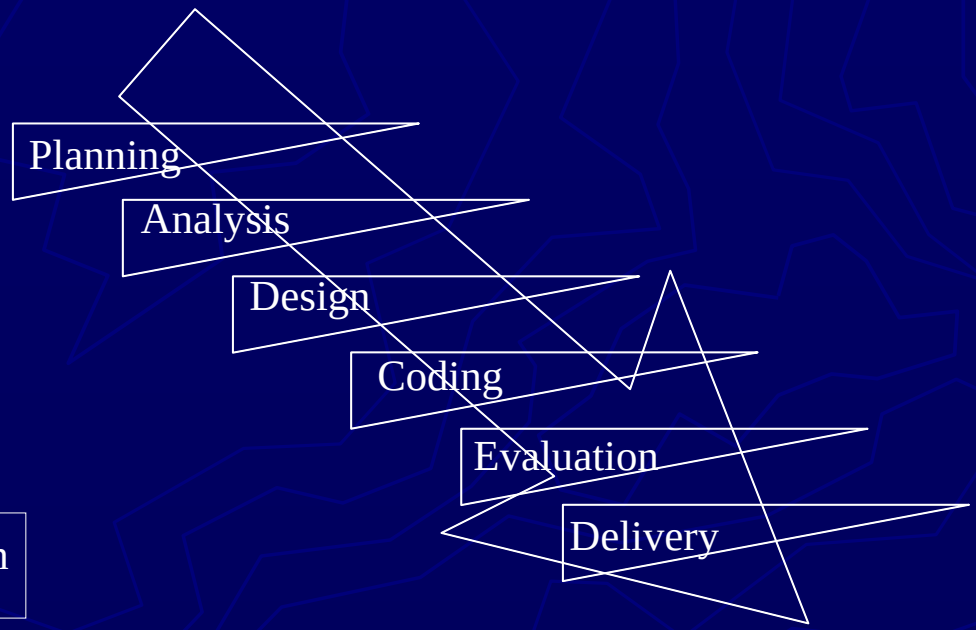
Iterative and incremental lifecycle

- Iterative : the process is applied several times
- Incremental : an iteration raises the knowledge
- A *better* waterfall



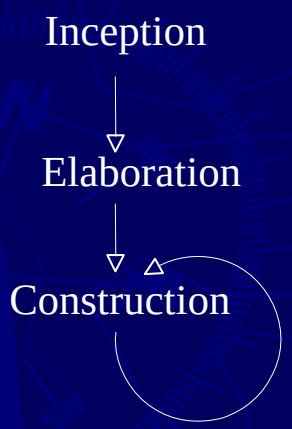
A Mini-Waterfall

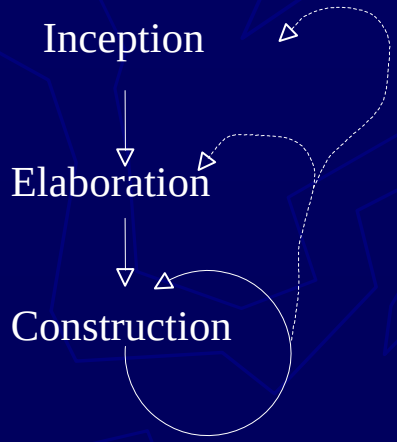


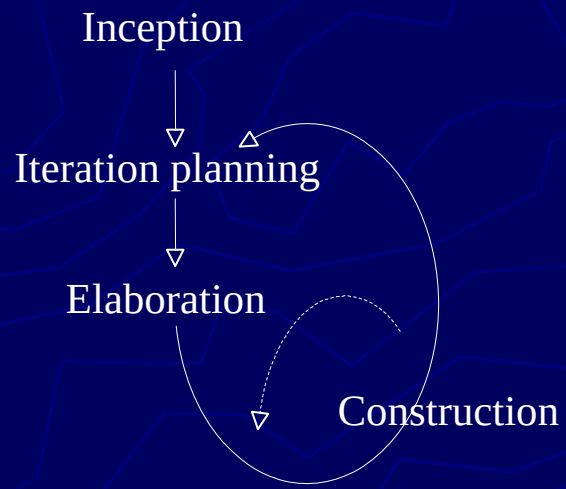


Iterative lifecycle variants

- According to project size, domain complexity and architectural choices
- The b-shaped is the most frequent

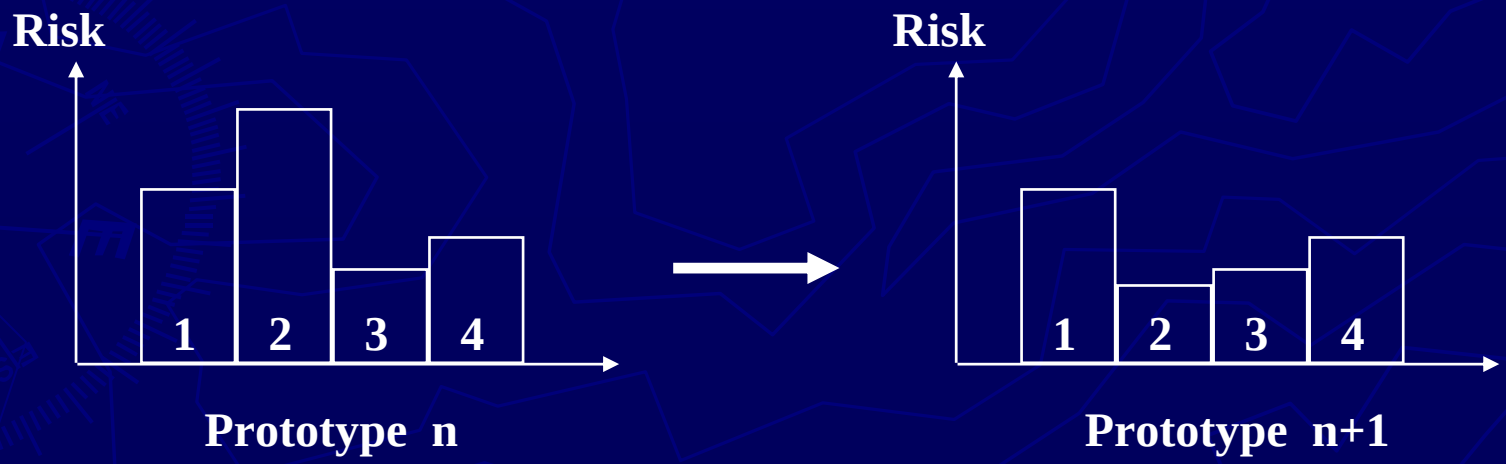




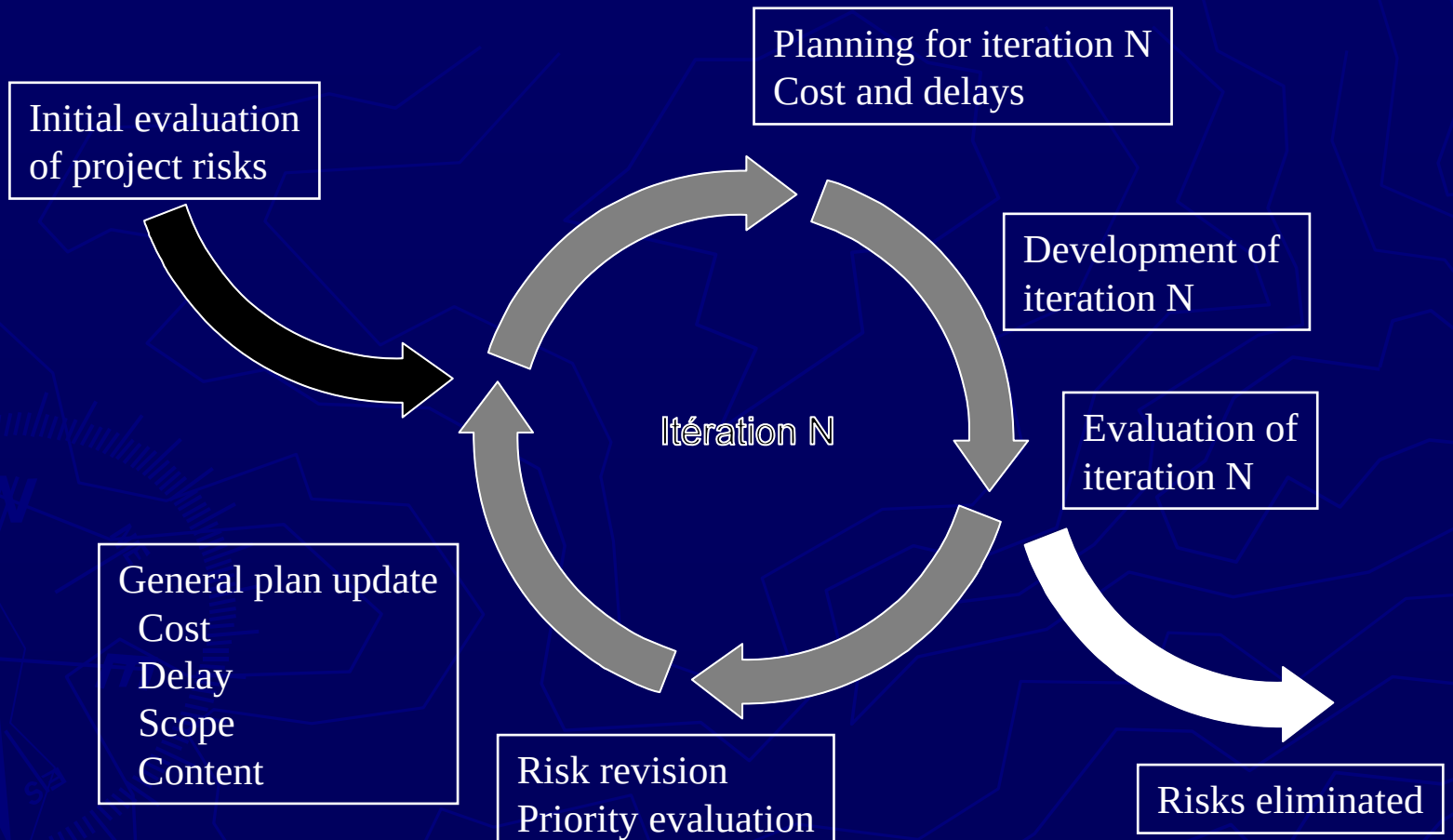


Risk management

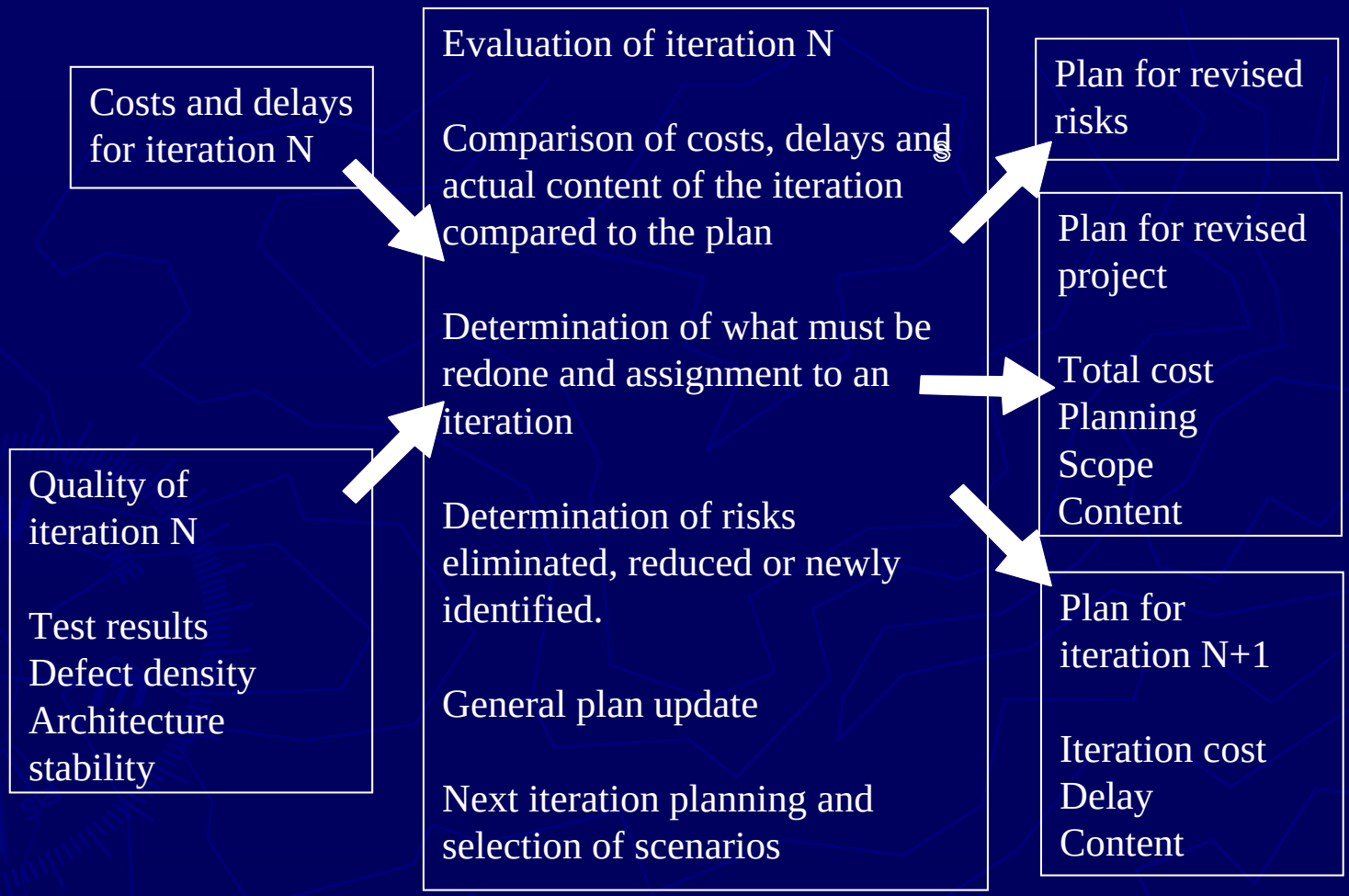
- Each prototype eliminates part of the overall risk
- A prototype is an executable program
- A prototype can be evaluated



Risk driven iteration



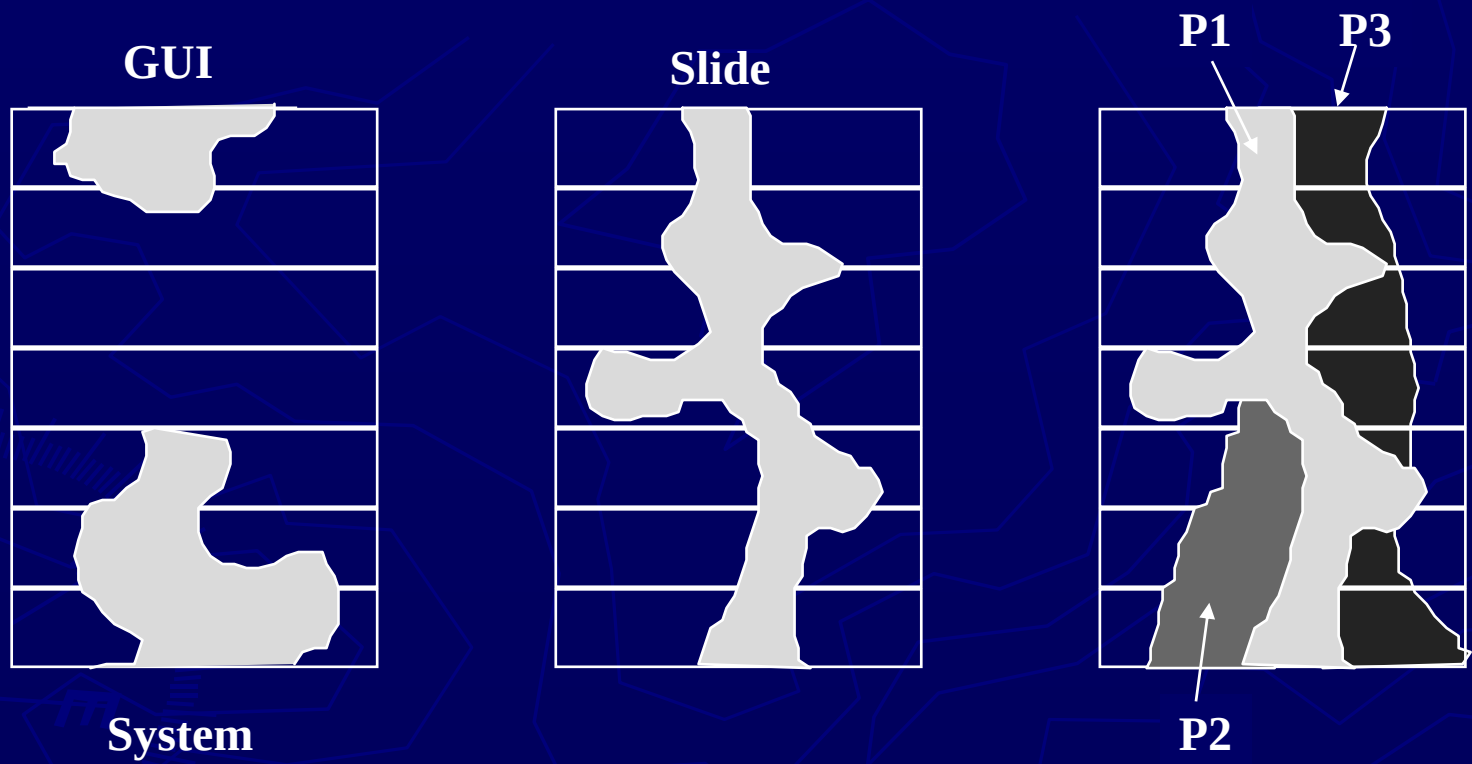
Evaluation of an iteration



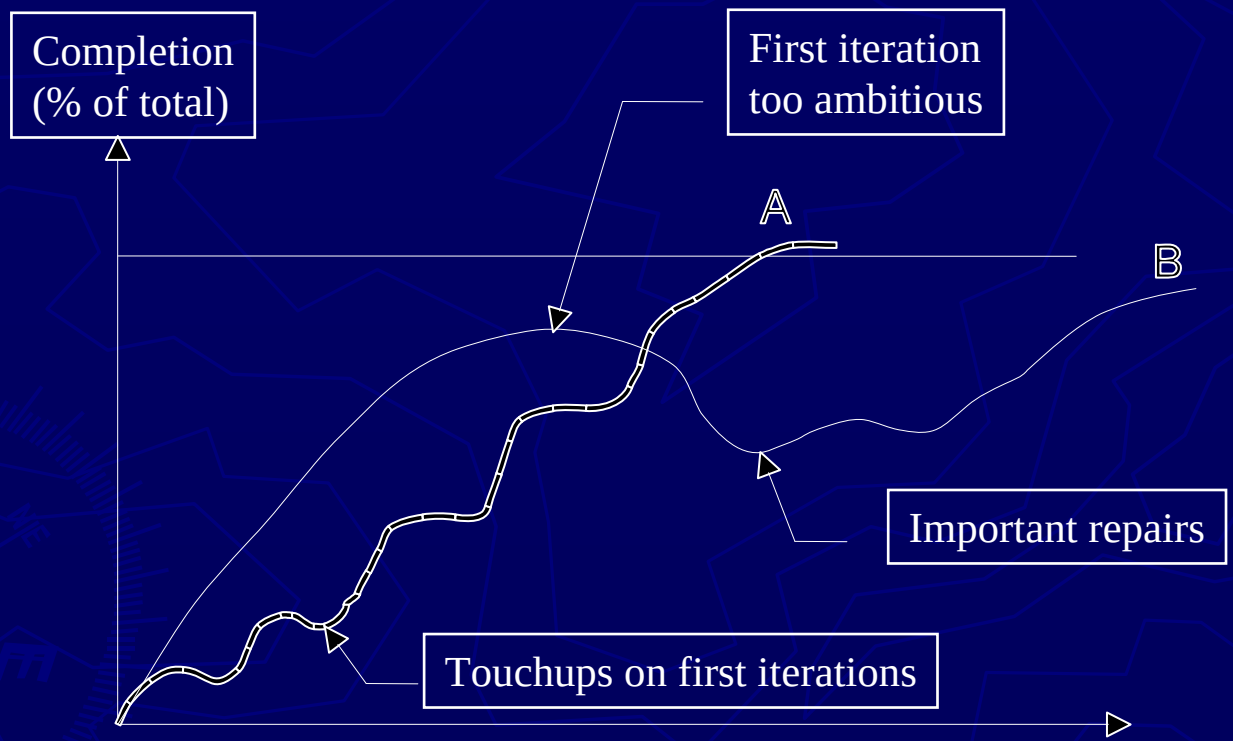
Recurring risks

- Integration too complex
- Poorly adapted development environment
- Users not favorable
- Technology too complex
- Manual activities too heavy
- Inadequate reusable components
- Excessive bureaucracy

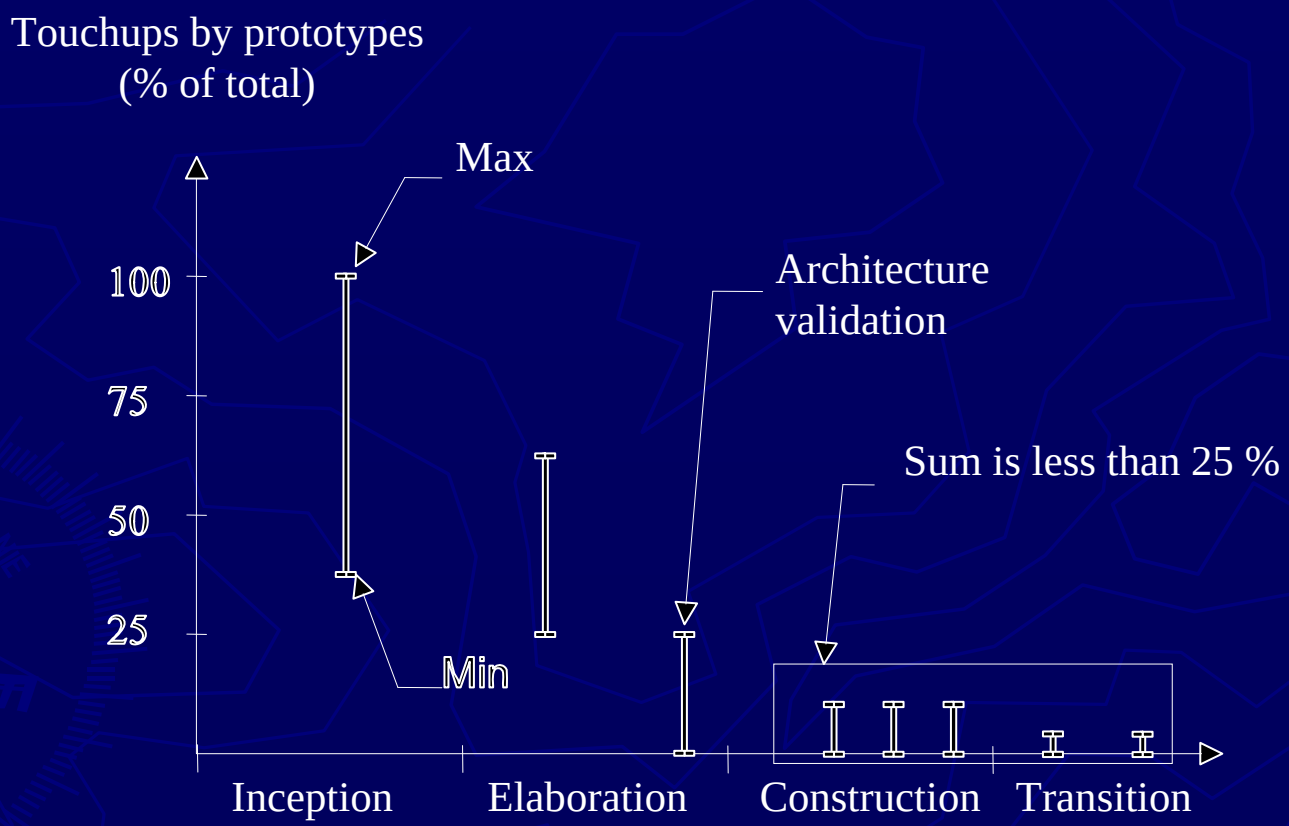
Structure of prototypes



Iteration planning



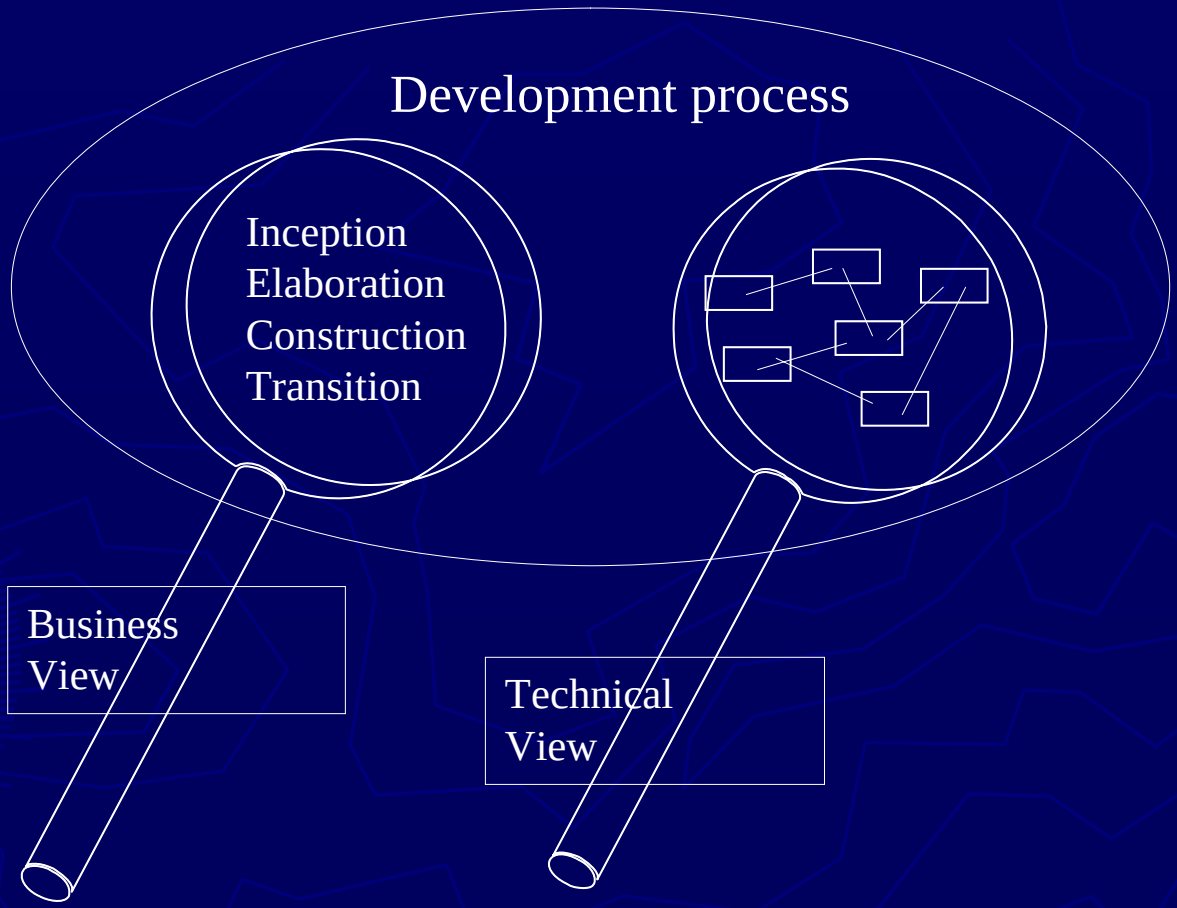
Touchup evolution



Implementing iterative lifecycle

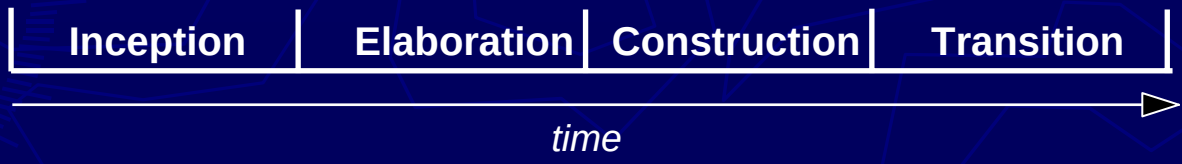
- Requires complete adherence of involved parties
- Business view
  - financial, strategic, commercial humans aspects
- Technical view
  - Engineering, quality and modeling method

Two complementary views



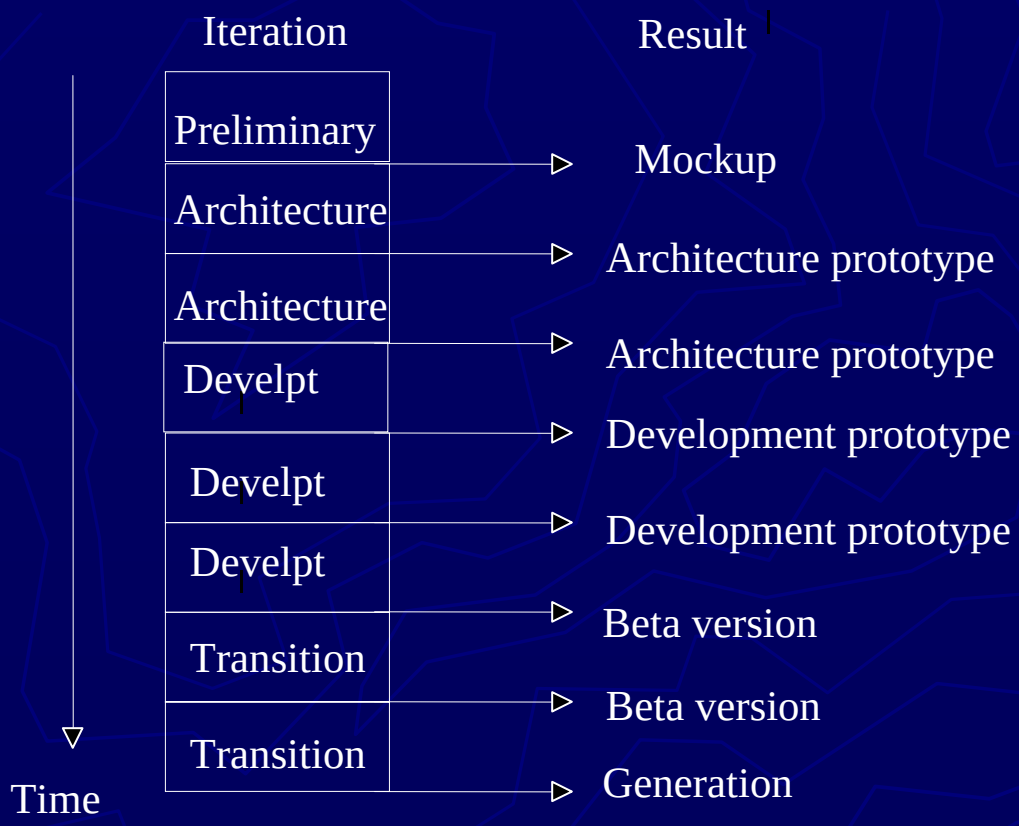
Business view

- Split into phases
  - Inception (feasibility study)
  - Elaboration (architecture, planning)
  - Construction
  - Transition

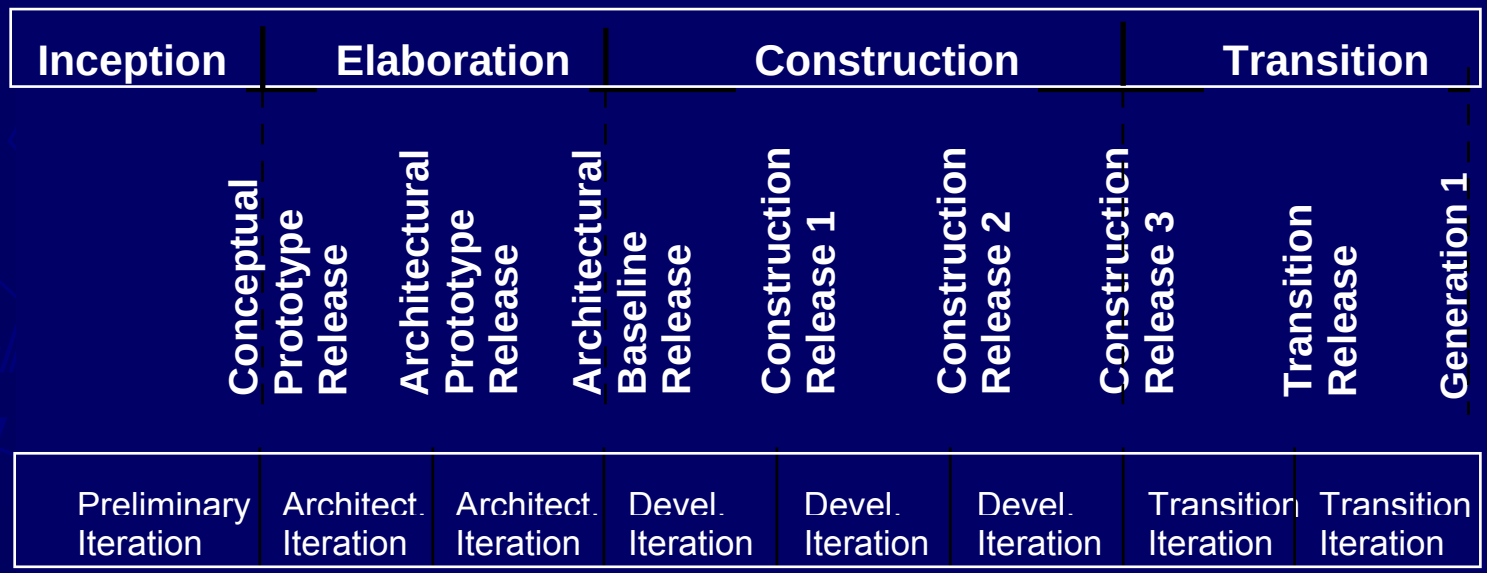


Technical view

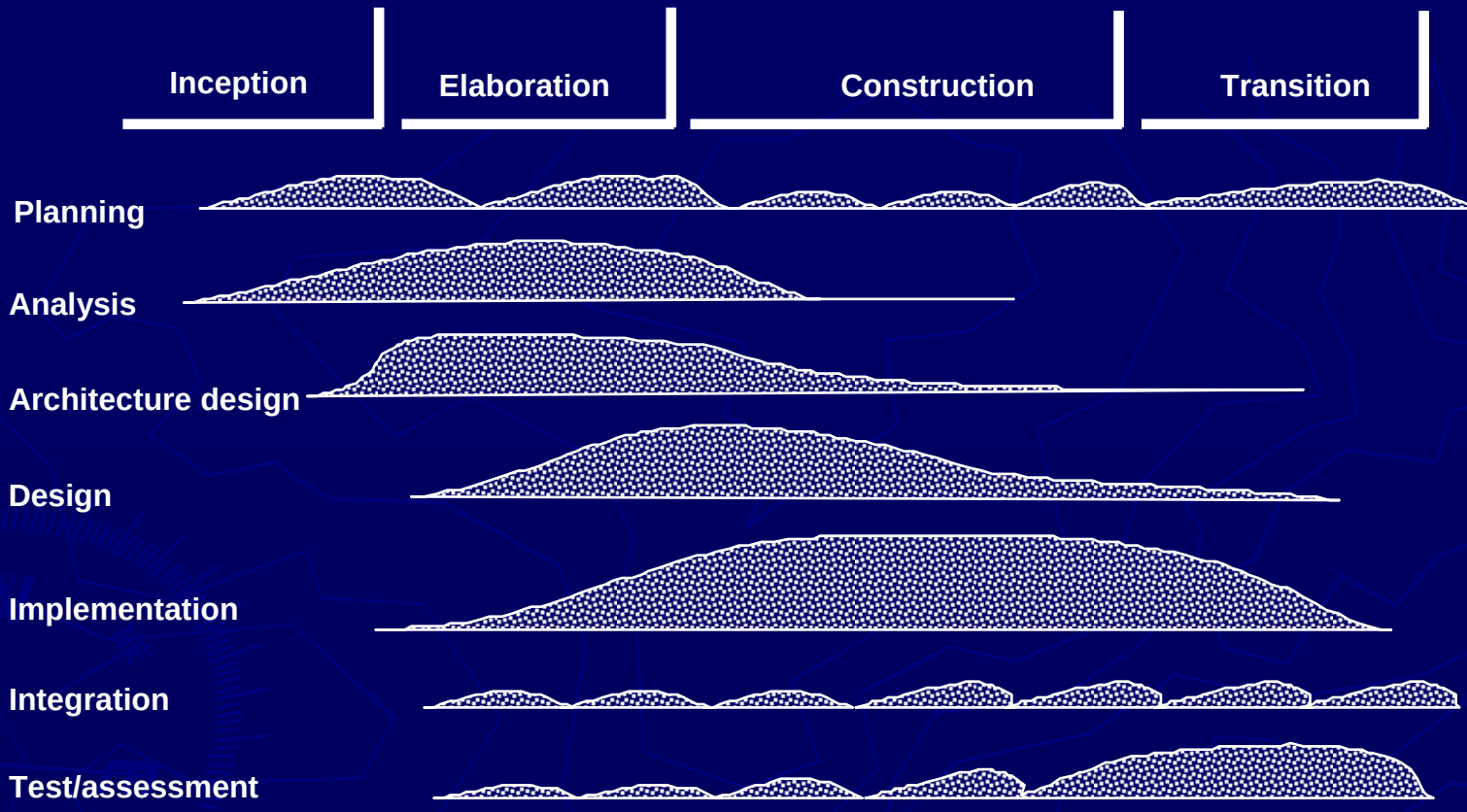
- Iteration driven



Synchronization of the two views

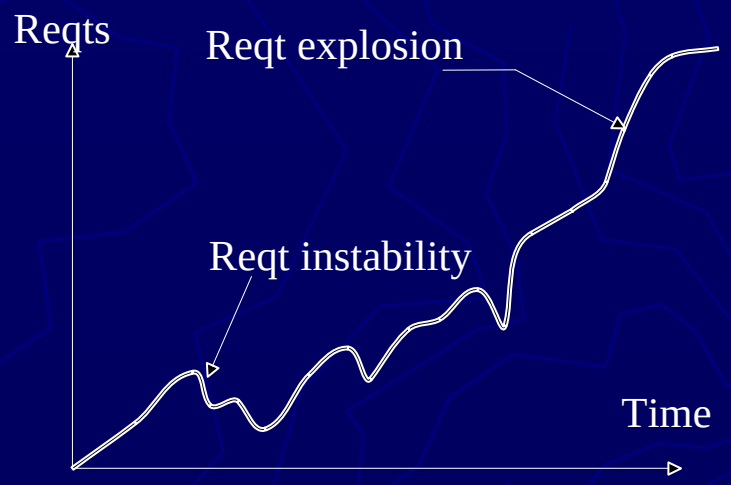


Activities and Phases



Misconceptions about the iterative lifecycle

- Encourages tinkering
- Causes problems
- Endless restarting
- Excuse to not plan
- Only applies to developers
- Encourages endless addition of new requirements



Conclusion

- Iterative lifecycle
  - simply copes with reality
  - allows evolution
  - based on prototyping
  - requires planing and management
  - require supporting environment
  - very compatible with OO