UNIVERSITÀ DEGLI STUDI DI TRIESTE

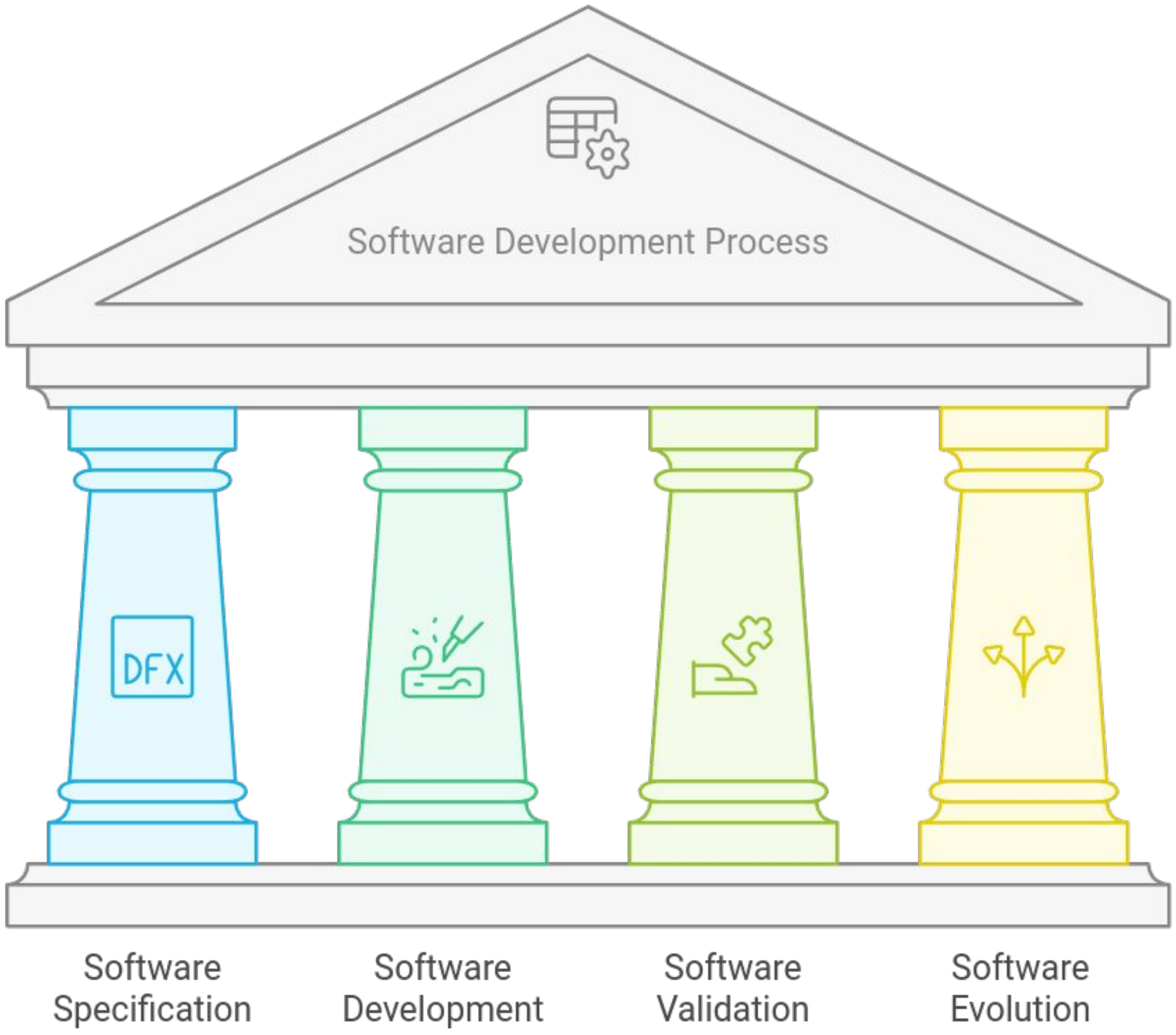
Class 11: Introduction to Software Process Models

Master Course:

Data-driven Systems Engineering (ML Operations)

440MI and 305SM

Data-driven Systems Engineering (ML Operations) 440MI and 305SM



Process activities - specification

The four basic **process activities** of:

- specification;
- development (design and implementation);
- validation;
- evolution;

Process activities

The four basic **process activities** of:

- specification;
- development (design and implementation);
- validation;
- evolution;

Process Models!!!

- Software process descriptions;
- Plan-driven, Incremental and Hybrid processes
- Plan-drive: Waterfall
- Incremental: Agile
- Downtime
- Agile
  - History
  - Principles
  - Manifesto
  - Drawbacks

Process Models!!!

- When we describe and discuss software project **processes**, we usually talk about the **activities** in these processes such as:
  - * specifying a data model
  - * designing a user interface
  - * service architecture

- **Process descriptions** may also include:
  - * **Products**, which are the outcomes of a process activity;
  - * **Roles**, which reflect the responsibilities of the people involved in the process;
  - *** Pre- and post-conditions**, which are statements that are true before and after a process activity has been enacted or a product produced.

Plan-driven and Incremental processes

- **Plan-driven** processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In **incremental** processes, planning is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Plan-driven processes

Key Points:

- Suitable for **large-scale systems** requiring strong coordination and documentation.
- Emphasizes **predictability, control, and formal planning**.
- Ideal for **stable environments** with clearly defined requirements.
- Requires experienced personnel mainly at **the initial planning and design stages**.
- Personnel thrive in **structured and organized workflows**, guided by well-defined processes.

Example:

Safety-critical systems such as aerospace, healthcare, or financial applications.

Iterative processes

Key Points:

- Suitable for **small teams and products** due to its limited scalability.
- Encourages **iterative development**, rapid experimentation, and **continuous feedback**.
- Works best in **dynamic environments** where requirements change frequently.
- Requires **experienced practitioners** throughout the process.
- Team members thrive in **flexible and adaptive settings**, valuing autonomy and creativity.

Example:

Startups developing new apps or features that evolve with user feedback.

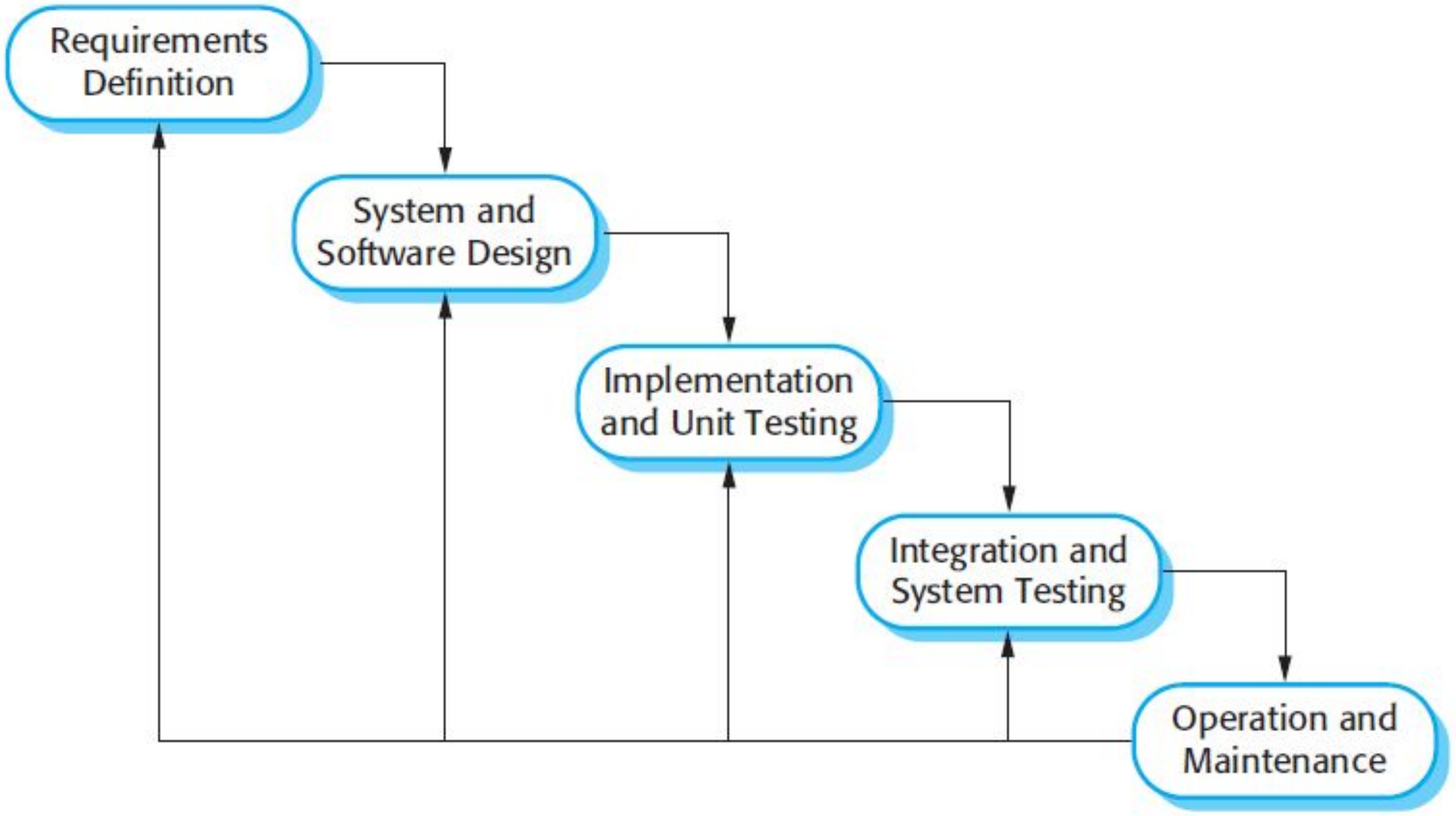
Hybrid Approaches

- Early stages: **Agile exploration** → rapid prototyping and validation.
- Later stages: **Plan-driven discipline** → scaling, integration, and maintenance.
- Combining both allows organizations to achieve:
  - **Agility for innovation**, and
  - **Structure for reliability**.

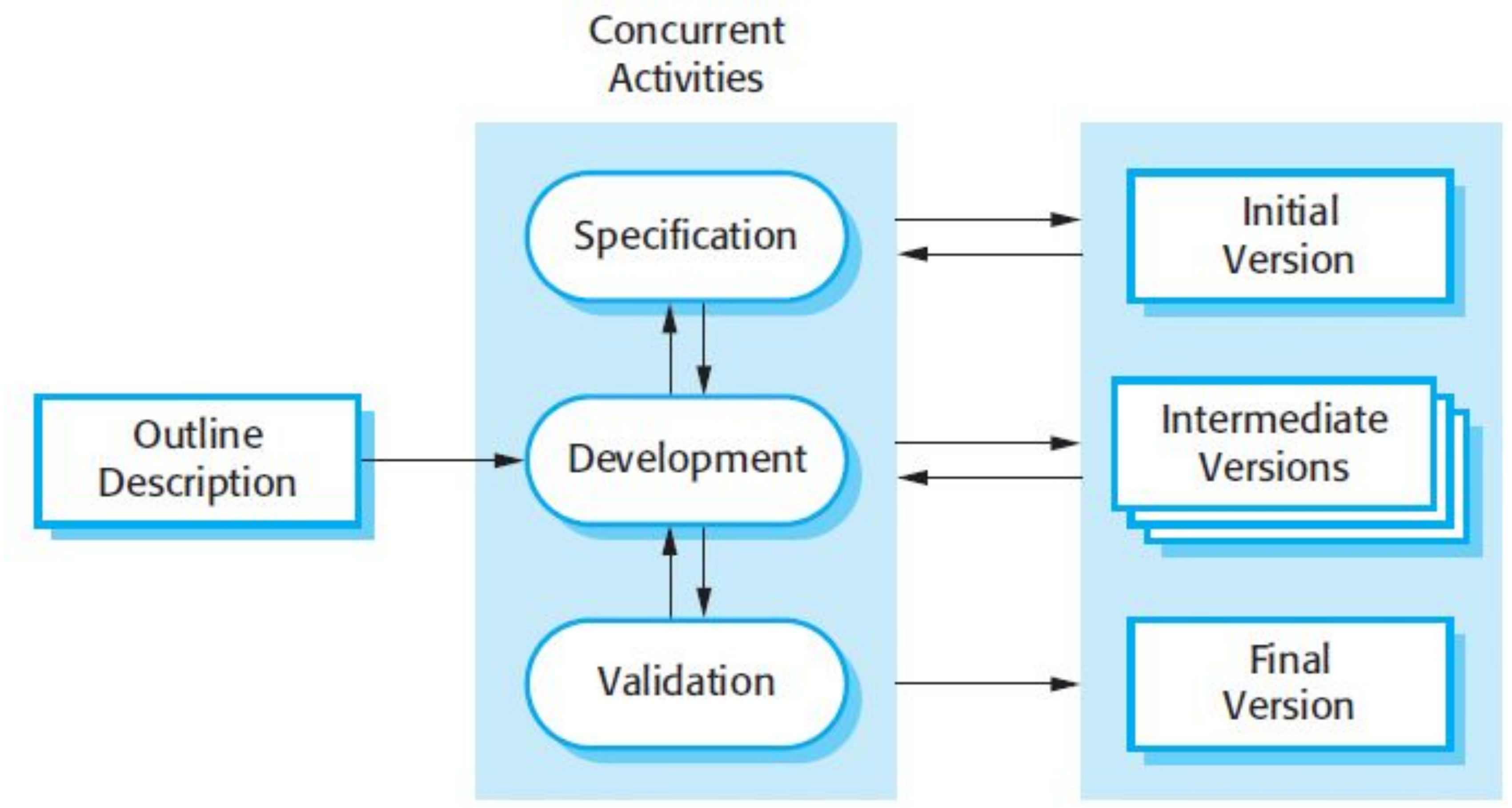
Conclusion:

The best development approach depends on context: team size, system criticality, and environmental stability.

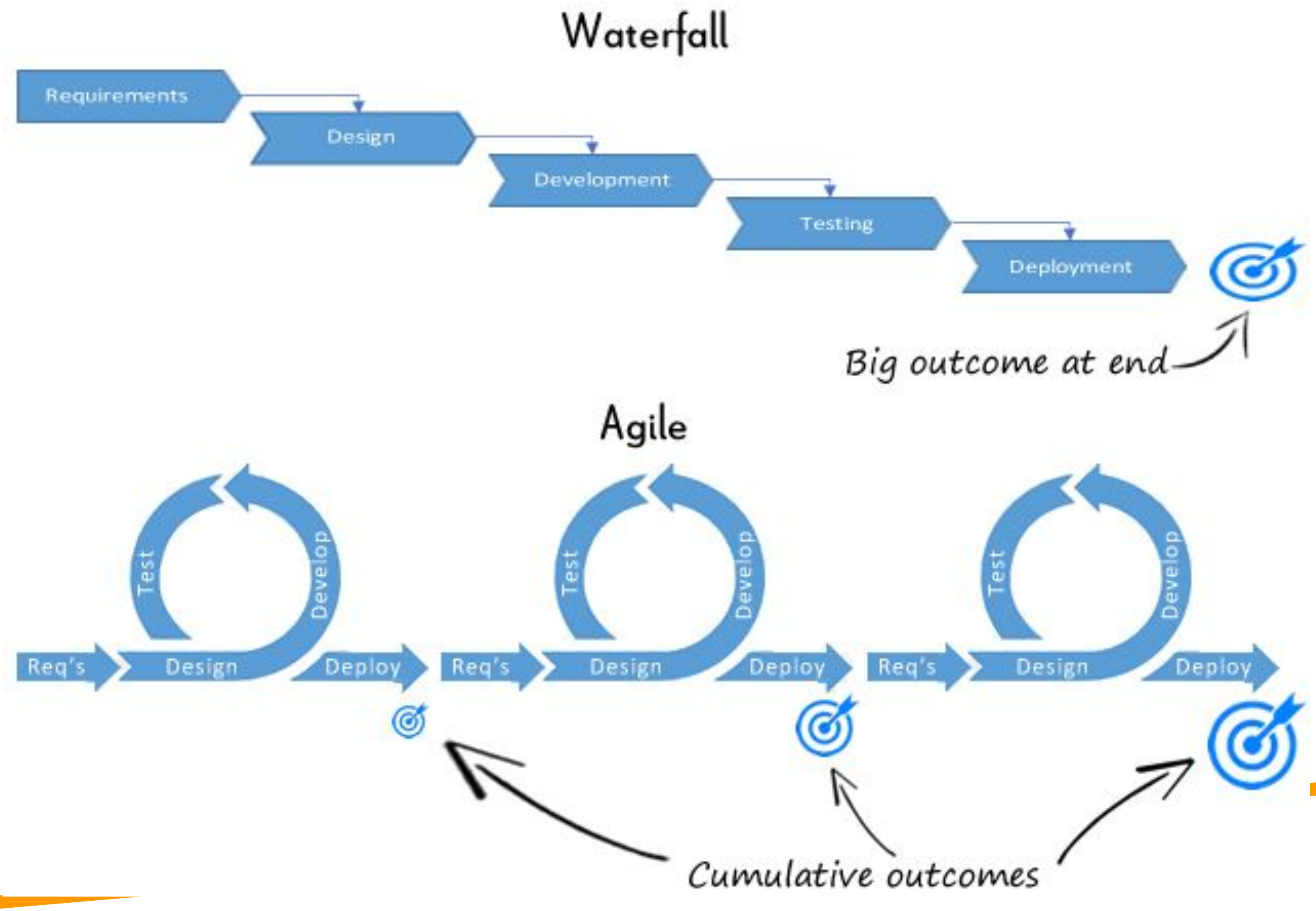
Plan-driven: Waterfall


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Incremental: Agile



Waterfall vs Agile



Downtime

Refers to the **period when the system or part of it is unavailable** to users, usually because of updates, integration, maintenance, or deployment activities.

In the **Incremental Development Model**, **downtime is minimized** because the system is **built, tested, and released in small, functional increments** rather than as a whole at once.

Suppose an e-commerce platform is being developed incrementally:

- In Increment 1, the login and registration system is deployed.
- In Increment 2, the shopping cart is added
- In Increment 3, the payment system is integrated.

During each increment:

- Only the **new component** (e.g., payment module) might cause short downtime when deployed.
- The rest of the system (login, cart, etc.) **remains operational**.

Agile - History

- **Dissatisfaction** with the **overheads** involved in **software design methods** of the 1980s and 1990s led to the creation of **agile methods**. These methods:
  - Focus on the **code** rather than the **design**;
  - Are based on an **iterative approach** to software development;
  - Are intended to **deliver** working software **quickly** and evolve this quickly to meet **changing requirements**;

- **The aim of agile methods is to reduce overheads in the software process** (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Agile - Definition

Agile - Definition

Rapid development and **delivery** is now often the most important requirement for software systems:

Businesses operate in a **fast-changing requirement** and it is practically **impossible** to produce a set of **stable** software **requirements**

Software has to evolve quickly to reflect changing business **needs**.

Rapid software development

1) **Specification**, **design** and **implementation** are interleaved;

2) **System is developed as a series of versions** with stakeholders involved in version evaluation

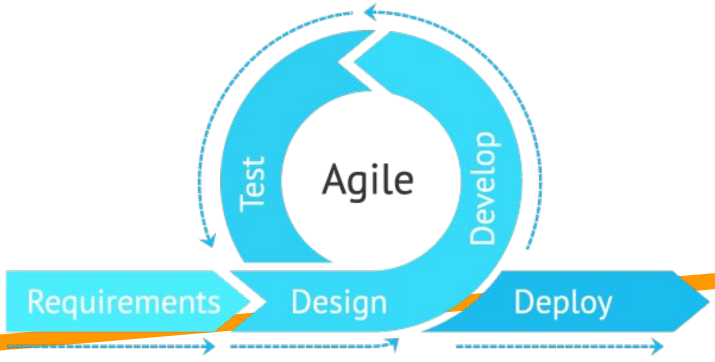
3) **User interfaces** are often developed using an **IDE and graphical toolset**.

Agile - Definition

Agile - Manifesto

- 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.

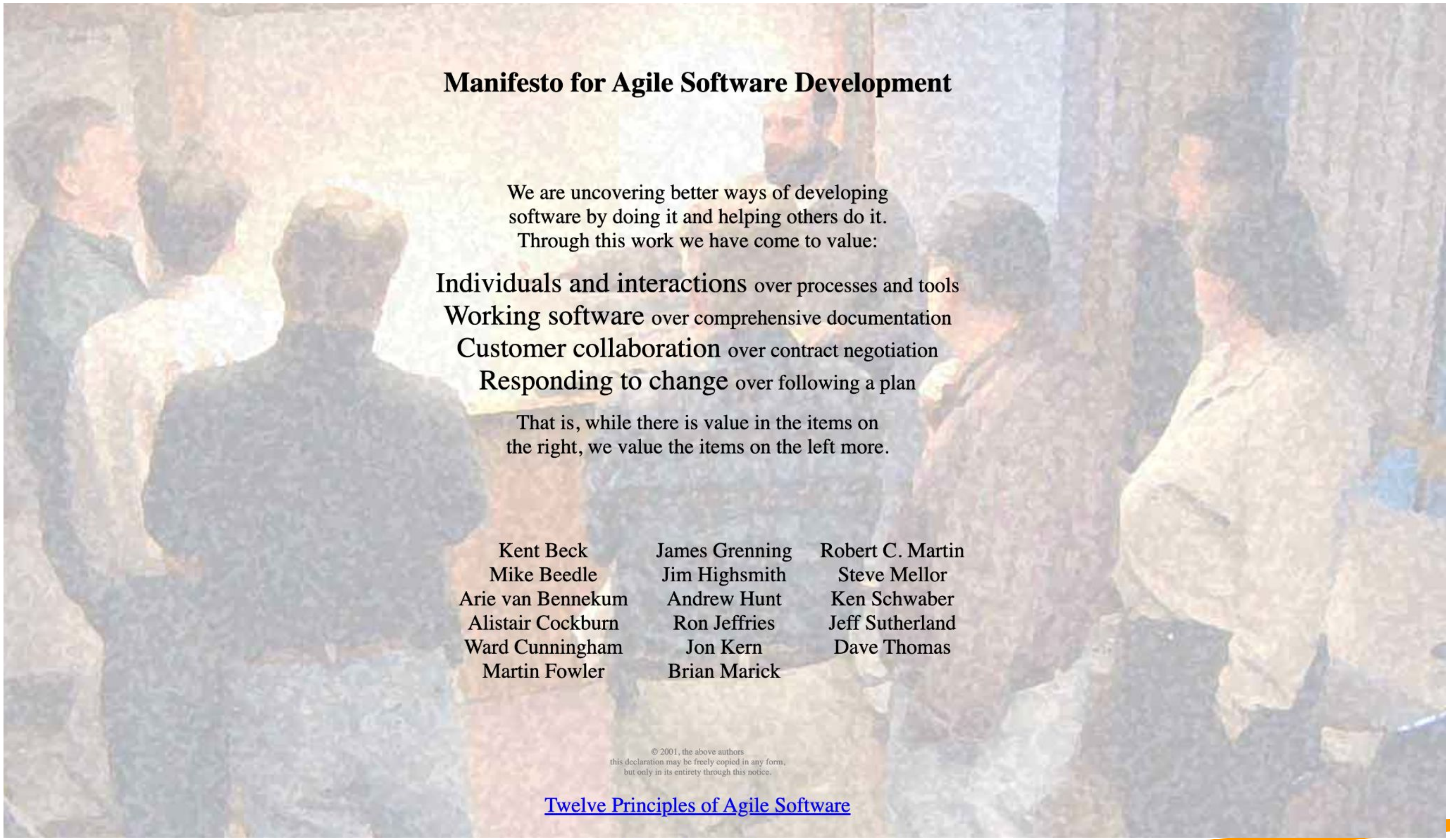


Agile - Principles

Principle	**Description**
Customer involvement	**Customers should be closely involved** throughout the development process. Their role is provide and **prioritize new system requirements** and to **evaluate the iterations** of the system.
Incremental delivery	The software is developed in increments with the **customer specifying** the requirements to be **included in each increment.**
People not process	**The skills of the development team should be recognized and exploited**. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	**Expect the system requirements to** change and so design the system to **accommodate** these changes.
Maintain simplicity	**Focus on simplicity** in both the **software** being developed and in the **development process.** Wherever possible, actively work to eliminate complexity from the system.

Agile - Problems

- It can be difficult to **keep the interest of customers** who are involved in the process.
- **Team members may be unsuited** to the intense involvement that characterises **agile methods.**
- **Prioritising changes can be difficult** where there are multiple stakeholders.
- **Maintaining simplicity requires extra work.**
- **Contracts may be a problem** as with other approaches to iterative development.



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	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Twelve Principles of Agile Software