Inf2-SEPP:
Lecture 12 Part 2: Design in Plan-Driven vs Agile Software Development Processes

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh
The extreme opposed viewpoints to design:

- From the classic Waterfall plan-driven process: Big design Up Front (BDUF)
- From the Extreme Programming (XP) agile process: 'You Aren't Gonna Need It' (YAGNI), 'Do the simplest thing that can possibly work' (DTSTTCPW), emerging design

More on design in different processes:

- In plan-driven processes
- In agile processes

What do companies do in reality?
Extreme opposed viewpoints to design: 1. From the classic Waterfall plan-driven process

Waterfall prevalent software development process for decades

*Big design Up Front (BDUF):*

- Derogatory term used by agile community referring to design in the classic plan-driven Waterfall Model
- Approach in which the system’s design is completed and perfected before starting the implementation
- Time, effort and money are invested into doing design properly
- Thorough documentation is kept on the design
BDUF: Advantages and disadvantages

Advantages:

▶ Good for systems with stable requirements
▶ Economical and efficient if changes can be predicted as everything is planned ahead of time
▶ Can simplify development, save rework, help understand design
▶ Easy to cost and schedule design

Disadvantages:

▶ In many contexts error prone as one cannot foresee all changes
▶ Can be wasteful if things do not go to plan
▶ To reduce risk, adding potentially useful functionality (’gold plating’) the design, which can turn out to be wasted effort
Extreme Programming (XP) one of the most influential agile processes, and the most specific regarding appropriate software engineering practices.

XP maxims regarding design:

- **You aren’t gonna need it (YAGNI):**
  - Not overengineering a design just because you think you may need some things later (i.e. ’gold plating’)
  - Focusing on requirements for each iteration

- **Related: Do the simplest thing that could possibly work (DTSTTCPW):**
  - Picking to do something that can be done quickly (right now)
  - Picking a minimal solution for solving the direct problem
  - Moving on to other important things to do as soon as possible
Extreme opposed viewpoints to design: 2. XP design maxims and practices

YAGNI and DTSTTCPW foundation of the practice of simple design.

YAGNI also considered to to be related to the XP practice of emergent design (evolutionary design):

- Minimal or no design up front (NDUF)
- Growing a design as your understanding of the problem (and its solution) evolves.
- Not creating lengthy documentation on the design
YAGNI and DTSTTCPW: Advantages and disadvantages

Advantages:

▶ Less wasteful in terms of time, money, effort (we may not get it right)
▶ Design is easier to understand (controversial)
▶ More targeted on needs
▶ Support agile overall, making short iterations possible
▶ Supported by agile: short iterations + feedback reactive to change so no need for ’gold plating’; can focus on the ’now’

Disadvantages:

▶ Not building flexible components and frameworks until they are needed questionable decision
▶ Another practice of XP, refactoring, seen by some as breaking YAGNI
Emergent design: Advantages and disadvantages

Advantages:

▶ Takes advantage of new learnings as they emerge
▶ Reactive to change
▶ Encourages collaboration within the team
▶ Uncertainty about the effectiveness of design removed
▶ Saves time by avoiding documentation that may not be useful

Disadvantages:

▶ Can make it difficult to see big picture of design, and may lead to mediocre, inconsistent design
▶ Can lead design to break (refactoring essential)
▶ Design difficult to cost
More on design in plan-driven processes

In plan-driven software development processes:

- Design (as requirements and other activities) is a separate stage in the software development
- Architectural and detailed design carried out thoroughly
- Heavyweight design documentation produced
- Formally using modelling and notation (e.g. UML class, sequence, communication diagrams) often associated with plan-driven development
- Outputs from design used to plan implementation
More on design in agile processes

In agile software development processes:

- Design and implementation the focus (‘working software’)
- *Agile not doing architectural design is a myth*; Overall system architecture seen as important in early stage of development.
- Design interleaved with requirements and implementation in each iteration; focusing on most important unfinished features
- No formal, detailed, design documents are produced (seen as waste of time): informal documents or design documentation is automatically generated by programming environment
- Outputs of design may not be specification documents, but reflected later in the code
- Models (e.g. UML class, sequence, communication diagrams) may be informally used to facilitate team communication
What do companies do in reality?

Reminder: companies in reality use a mix and match of software development processes

Many critics see neither BDUF nor the XP maxims/practices as ideal in all situations. They are usually best for software which fit the processes which produced them.

Some more mixed approaches to design were also proposed:

- "Just enough" up front design
- Adaptable design up front

If interested, see recommended resources
Resources (check links)

► Essential:
  ► ’Big Design Up Front Versus Emergent Design’, Anthony Langsworth
  ► First part (until separating line) of c2 wiki ’Do the Simplest Thing That Could Possibly Work’
  ► c2 wiki ’You Arent Gonna Need It’ until ’humor’
  ► ’Is design dead?’ by Martin Fowler, until ’Patterns and XP’
Recommended:
- 'Just Enough’ Up Front Design by Simon Brown
- 'Adaptable Design Up Front’ by Hayim Makabee