Process Agility and Software Usability: Toward Lightweight Usage-Centered Design

by Larry L. Constantine

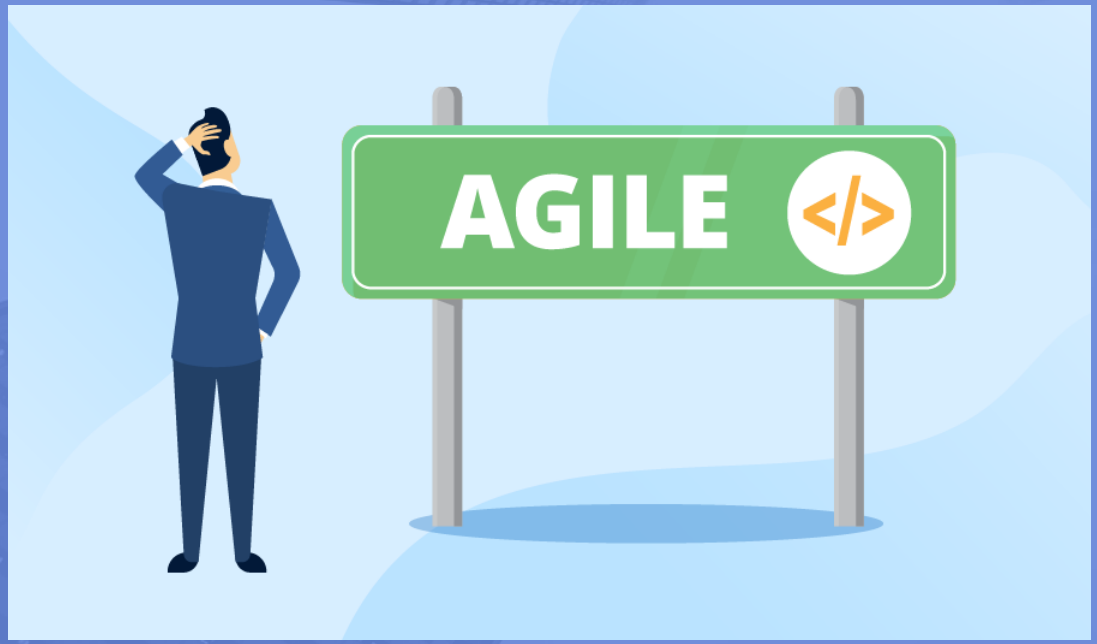
HELLO!
I AM ALESSANDRO SCARLATO

But first a bit of Accessibility!

QR CODE

You can download this slides
by scanning this code

What is Agile Development?



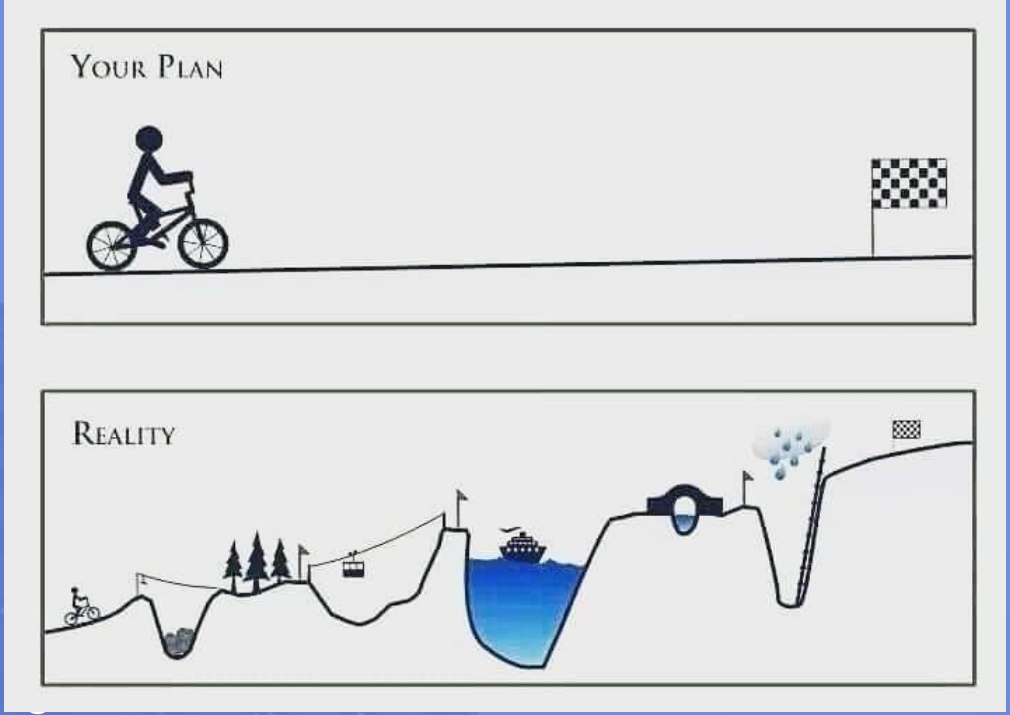
Agile Development Methodology" is an umbrella term for a number of different iterative and incremental software development methodologies. Compared to traditional waterfall models, they are all lightweight and adaptable.

Successful agile teams can produce higher-quality software better meeting user needs quicker and at a lower cost.

BASICS OF AGILE

- Satisfy customer through early and continuous delivery of features
- Embrace change, even late-changing requirements
- Gives your customer a competitive advantage
- Co-location of roles, daily close collaboration
- Simplicity – maximize amount of work NOT done

What is Agile Development?



Remember agility is not just being fast!

Agility isn't about speed. Agile is about navigating ambiguity, adapting to change & applying new learnings.

The ability to create and respond to change in order to succeed in an environment.

What is Agile Development?

Agile methods complete small portions of the deliverables in each delivery cycle (iteration or sprints).

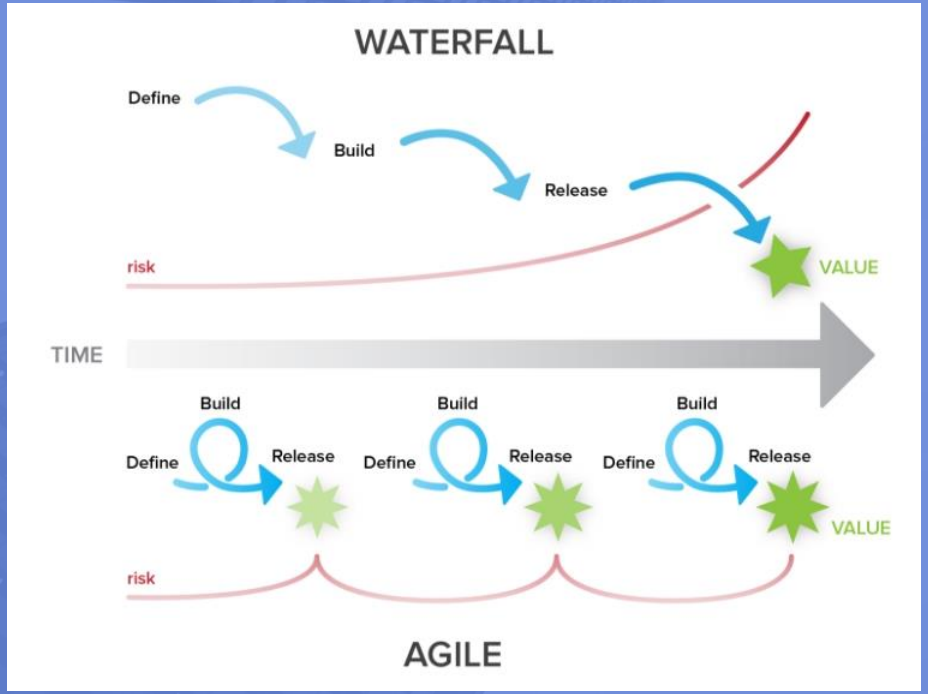
Iterative methods evolve the entire set of deliverables over time, completing them near the end of the project.

Iterations, or sprints, are short time frames (timeboxes) that typically last from one to four weeks. All cycles during an iteration: planning, analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders.

THE RULES OF AGILITY ARE VERY SIMPLE:

- Work in short release cycles.
- Do only what is needed without embellishment.
- Don’t waste time in analysis or design, just start cutting code.
- Describe the problem simply in terms of small, distinct pieces, then implement these pieces in successive iterations.
- Develop a reliable system by building and testing in increments with immediate feedback.
- Start with something small and simple that works, then elaborate on successive iterations.
- Maintain tight communication with clients and among programmers.
- Test every piece in itself and regression test continuously.

AGILE VS WATERFALL



The waterfall model: Do each lifecycle activity in sequence for whole system and then move on to next lifecycle activity. Waterfall model can't manage change (no feedback until it's too late).

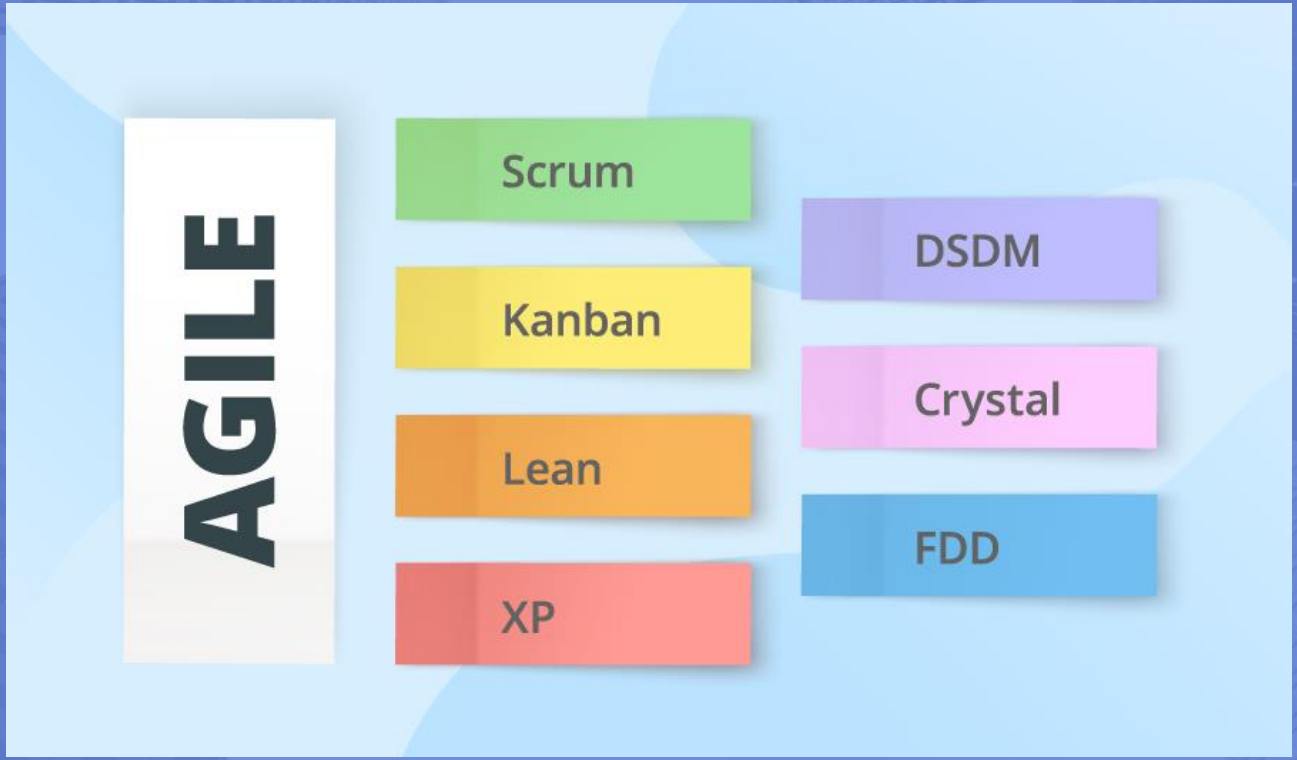
Agile process: Do all the lifecycle activities for one feature of system and then move on to next Feature. In the case of Agile, the customer has space for changes

AGILE MAINFESTO

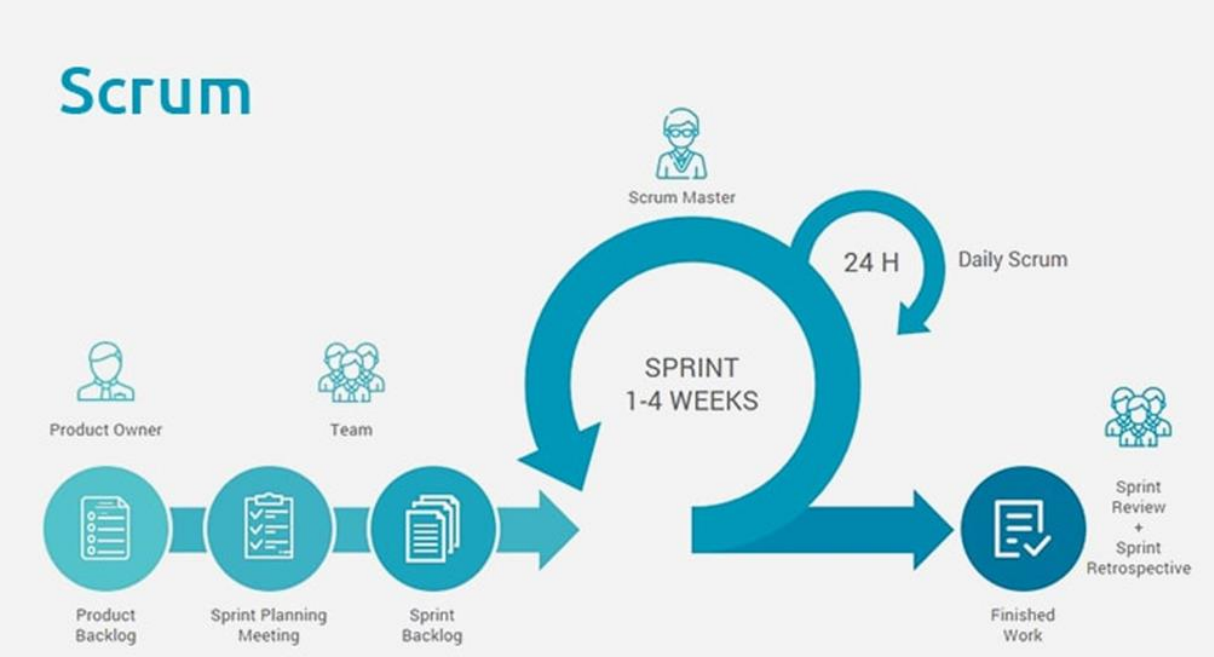
In 2001 , 17 software developers met at a resort in Snowbird, Utah to discuss lightweight development methods. Together they published the Agile Manifesto.

The Manifesto for Agile Software Development is based on this twelve principles.

AGILE METHODOLOGIES



SCRUM

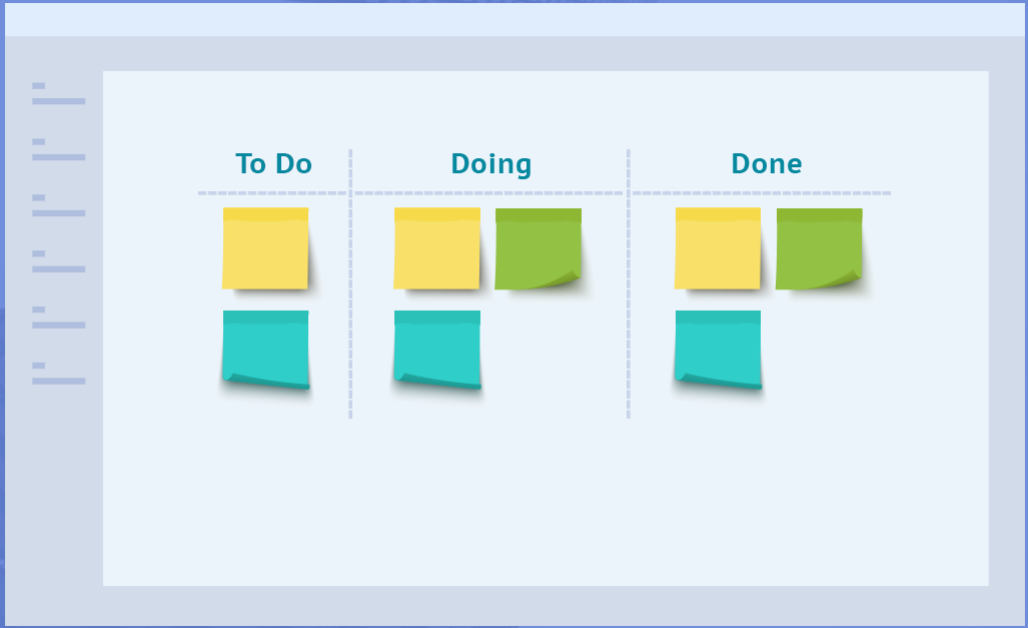


Scrum is an approach where the development process is split into short (1-4 weeks) incremental interactions (called sprints), project progress is constantly tracked, and planned meetings with the customers and developers are regularly conducted.

LEAN SOFTWARE DEVELOPMENT

Lean is an approach focused on minimizing waste in the software development (excessive functionality, downtime in the development process, unclear requirements), promoting efficient use of team resources, and concurrent teamwork.

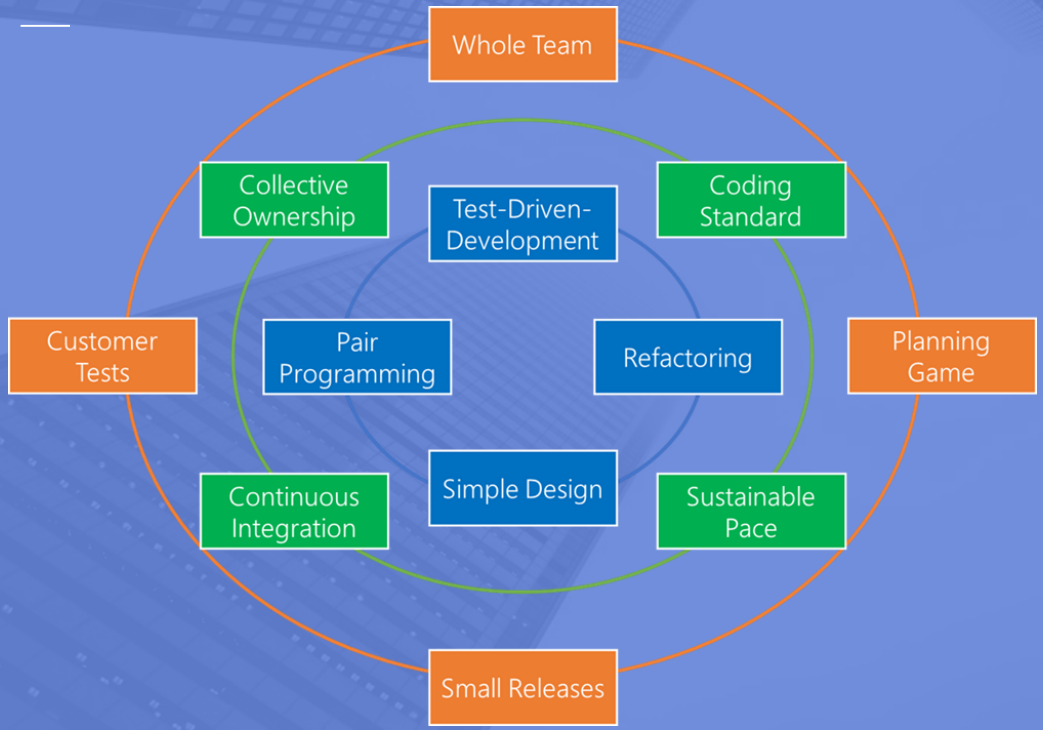
KANBAN



Kanban is an approach based on a visual representation of the work in progress and limiting tasks taken in an iteration to meet the iteration goals and avoid the development team overburdening.

Kanban is a visual way to manage tasks and workflows, which utilizes a kanban board with columns and cards. The cards represent tasks, and the columns organize those tasks by their progress or current stage in development.

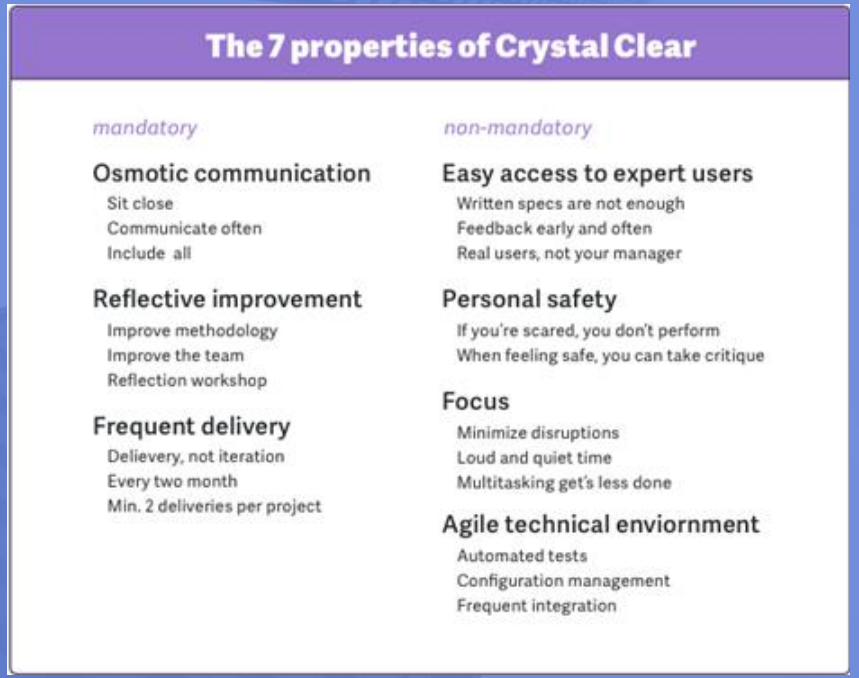
EXTREME PROGRAMMING



XP is focused on speed and continuous delivery. It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks.
Taken to the extreme, code can be reviewed continuously through the practice of pair programming , with one programmer writing the code and the other one reviewing the line of code written.

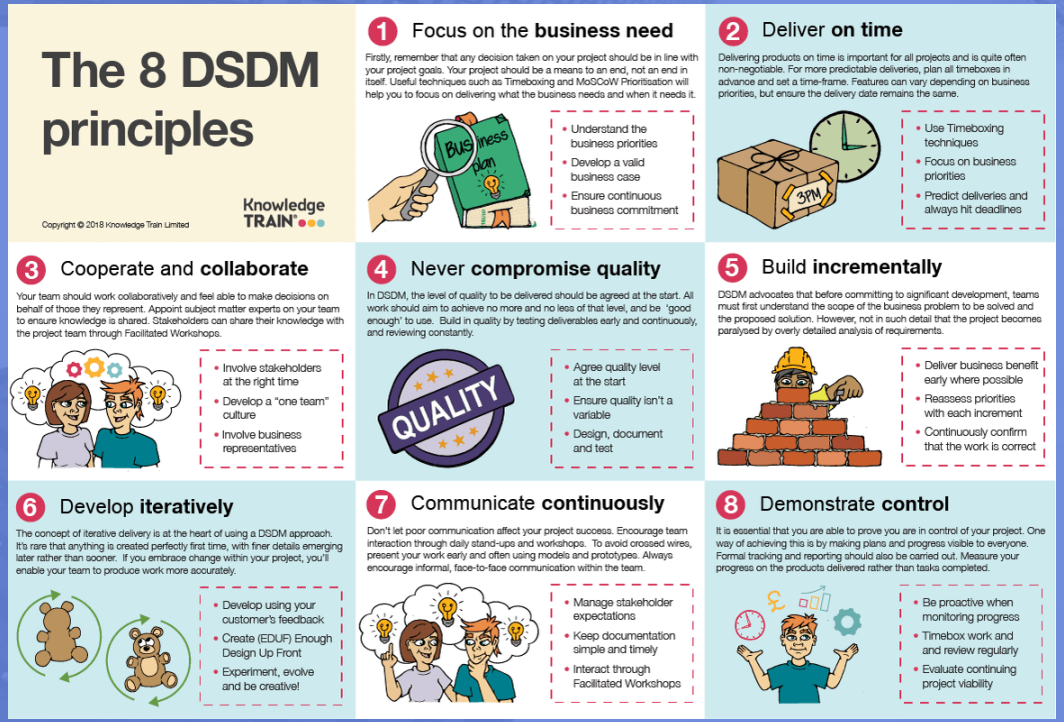
it has the twelve supporting practices shown in the picture

CRYSTAL



Crystal is a family of sub-methodologies that scales projects based on their size and criticality, applying different sub-methodologies respectively. Each has unique characteristics driven by several factors, such as team size, system criticality, and project priorities

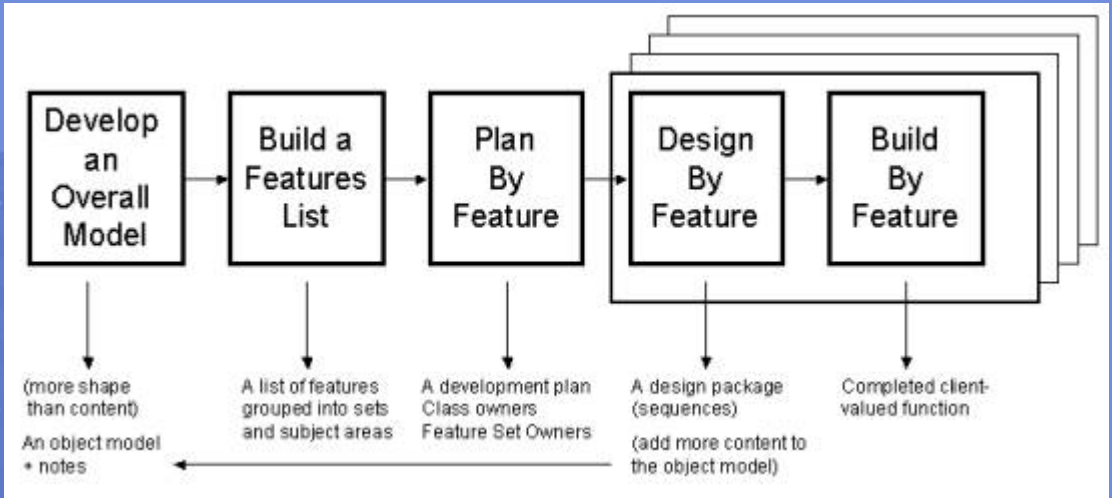
DYNAMIC SYSTEMS DEVELOPMENT METHOD (DSDM)



DSDM is based on eight key principles that direct the team and create a mindset to deliver on time and within budget. These agile principles primarily revolve around business needs/value, active user involvement, empowered teams, frequent delivery, integrated testing, and stakeholder collaboration.

FEATURE DRIVEN DEVELOPMENT

Feature Driven Development consists of five basic activities:



FDD (Feature-Driven Development) is an approach where the overall software model is created first, followed by preparing a list of features required, planning, designing, and building by feature.

AGILE USAGE-CENTERED DESIGN

Usability and the user experience are emerging as critical determinants of success.

If customers cannot find what they are looking for, they cannot buy it.

Poorly designed interfaces increase user errors, which can be costly. Mistakes made entering credit card billing information, for example, can lead to lost sales. Billions of dollars in lost Web sales can be traced to usability problems.

In addition to usability, aesthetic aspects of design may also figure in the user experience.

AGILE USAGE-CENTERED DESIGN

User interface design and usability are acknowledged as weak points in both the “heavyweight” processes, such as the so called Unified Process, and the lightweight competitors, such as Extreme Programming.

XP and the other light methods are light on the user side of software. They seem to be at their best in applications that are not GUI-intensive.

XP do explicitly provide for user or client participation in pinning down requirements and setting scope through jointly developed scenarios, known as user stories. But user stories are typically written by customers or customer representatives, not necessarily by genuine direct end-users. Confusion between "end users" and "customers"

CONCLUSIONS BENEFITS 

- **Agile is a lightweight methodology:** It has only a few rules and practices, or only ones that are easy to follow. In contrast, a complex method with many rules is considered a heavyweight methodology.
- **The agile processes concentrate on code:** Design is on-the-fly and as needed. No need to prepare thorough documentation. This is because the project scope may be changed.
- **Early results:** With short release cycles the developers get down to the project as soon as they become aware of its basic features.
- **More reliable code:** Wich means more up-time, cheaper development, and less support and maintenance.

CONCLUSIONS PITFALLS

- **Poor Resource Planning:** Because teams won't know what their end result will look like from day one, it's challenging to predict efforts like cost, time and resources required at the beginning of a project.
- **Limited Documentation:** Usually not at the beginning. As a result, it becomes less detailed.
- **Fragmented Output:** When a teams work on each component in different cycles, the complete output often becomes very fragmented rather than one cohesive unit.
- **No Finite End:** There is never a clear vision of what the "final product" looks like.
- **Difficult Measurement:** The"see-as-you-go" nature makes measuring progress difficult.

THANKS

Any questions?