California State University, Sacramento
Computer Science Department

CSC 131 : Computer Software Engineering

Fall 2022

Lecture # 1

Introduction:

Software Engineering Concepts, Process and Models

SACRAMENTO STATE
Redefine the Possible

What is Software?

A **set of instructions** that are designed to accomplish a desired task or function.

Software Includes:

- ✓Programs
- ✓Documents
- ✓Data…

Characteristics of Software

- A software is **engineered or developed**.
- A software does not wear out, but it **deteriorates**.
- A software is still mostly **custom built**.

Wear vs. Deterioration

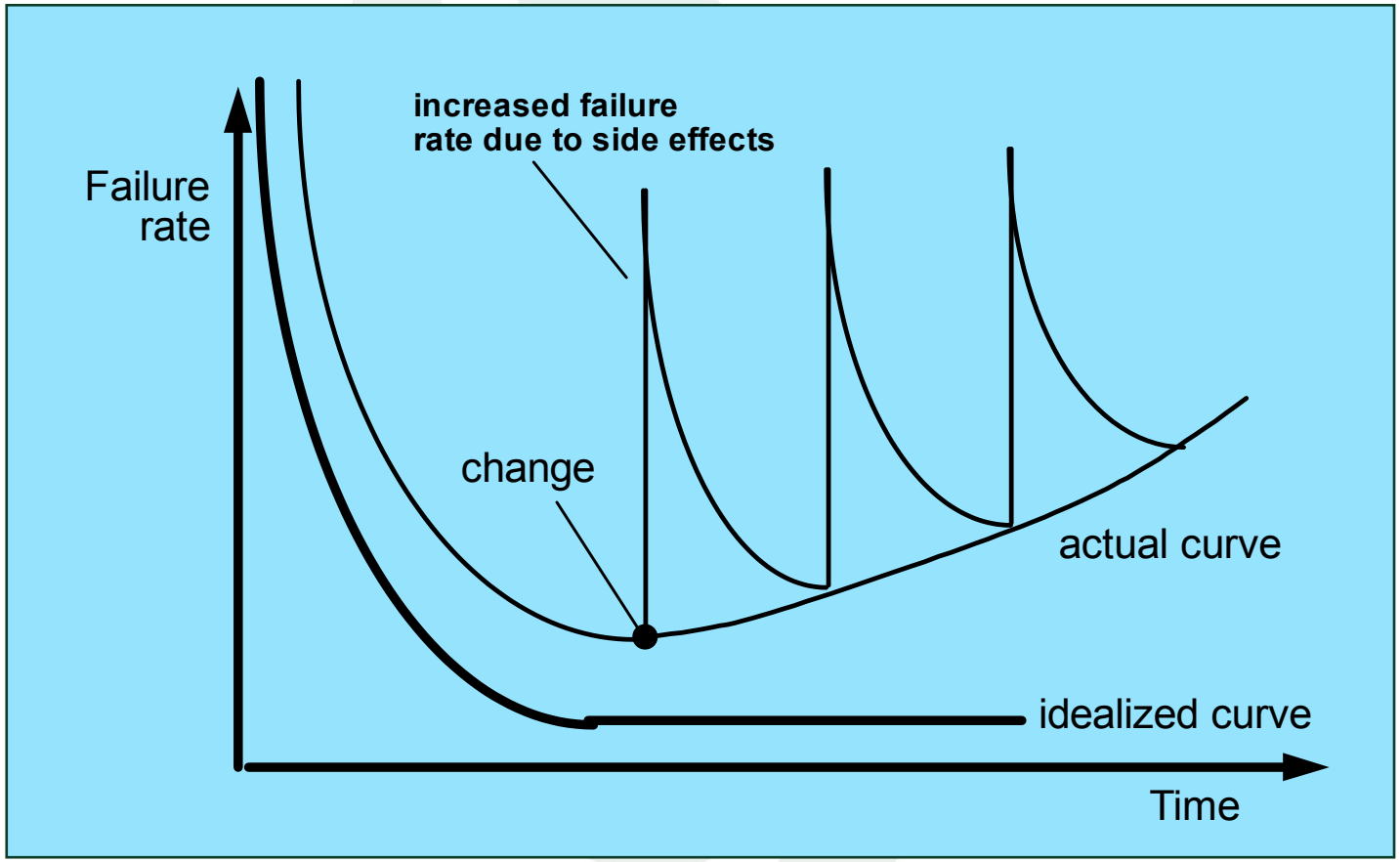


Fig 1

The Cost of Change

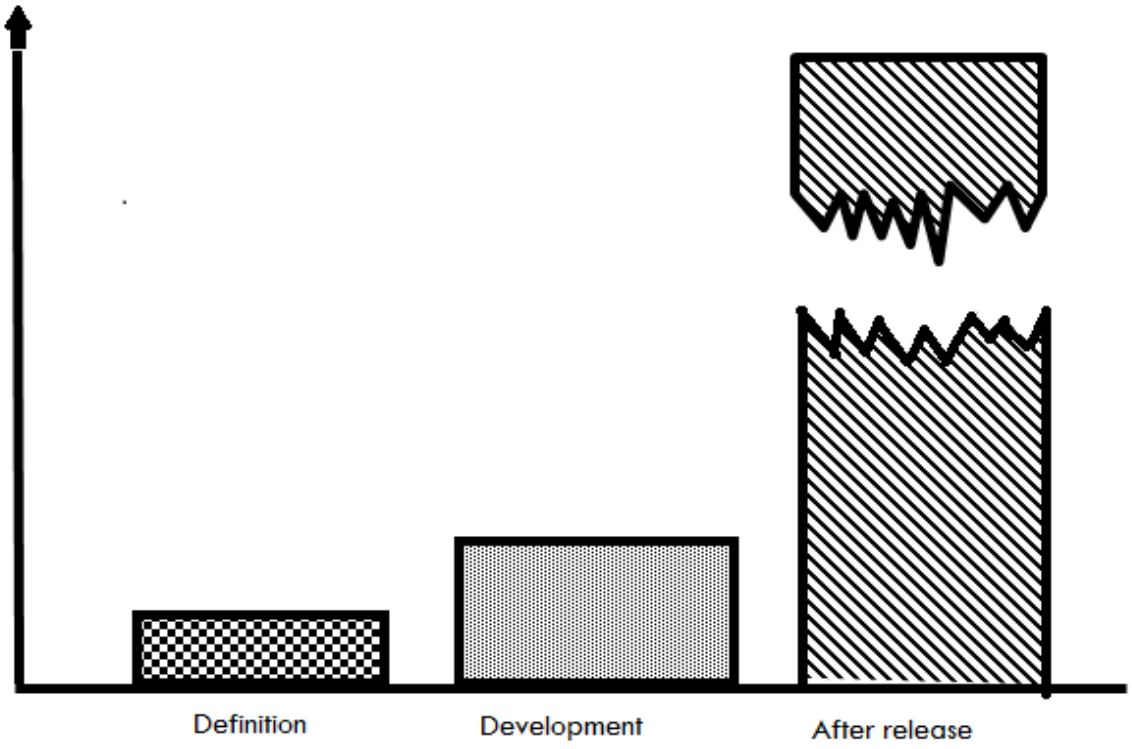


Fig 2

Software Applications

- System software
- Real-time software
- Business software
- Engineering/scientific software
- Embedded software
- PC software
- AI software
- WebApps (Web applications)

Software Engineering Classic Definition (1969)

- The establishment and use of sound engineering principles in order to obtain economically software that is **reliable** and works **efficiently** on real machines."

"Software Engineering:

(1) The application of a **systematic**, **disciplines**, **quantifiable** approach to the development, operation, and maintenance of software; that is the **application of engineering to software**.

(2) The study of approaches as in (1)."

Software Engineering Characteristics

- **Predictable Results**
  - ✓ Product will be produced.
  - ✓ Production/release time is known in advance
  - ✓ If results were predictable, we would not see V1.0.1, V 1.0.2, etc. (Are these bug fix releases..?)

- **Measurable Progress**
  - ✓ Product goes through well defined phases
  - ✓ Engineers can reliably measure progress
  - ✓ Metrics can be defined to provide accurate indication of progress and cost.

Software Engineering Characteristics

- **Ability to React/Adapt to Change**
  - ✓ The product is well planned and documented
  - ✓ Changes in requirements is well anticipated
  - ✓ Impact of change more readily assessed.

- **Other Characteristics of an Engineering Process**
  - ✓ Well defined and repeatable process
  - ✓ Standardized guidelines and procedures
  - ✓ Documentation

The Process

Software Process

- Definition

The activities, techniques, tools, and individuals to produce software/system.

Process Principles

- Prescribes all major activities
- Uses resources, within a set of constraints, to produce intermediate and final products.
- May be composed of sub-processes.
- Each activity has entry and exit criteria.
- Activities are organized in **a sequence**.
- Has a set of **guiding principles** to explain goals.
- Constraints may apply to activity, resource or product.

Capability Maturity Models CMM & CMMI

Capability Maturity Model (CMM) & Capability Maturity Model Integration (CMMI)

- ✓ Developed by **SEI** – (Software Engineering Institute at Carnegie Mellon)
- ✓ The CMMI project is sponsored by the U.S. Department of Defense (**DoD**) and the National Defense Industrial Association (**NDIA**).

CMM

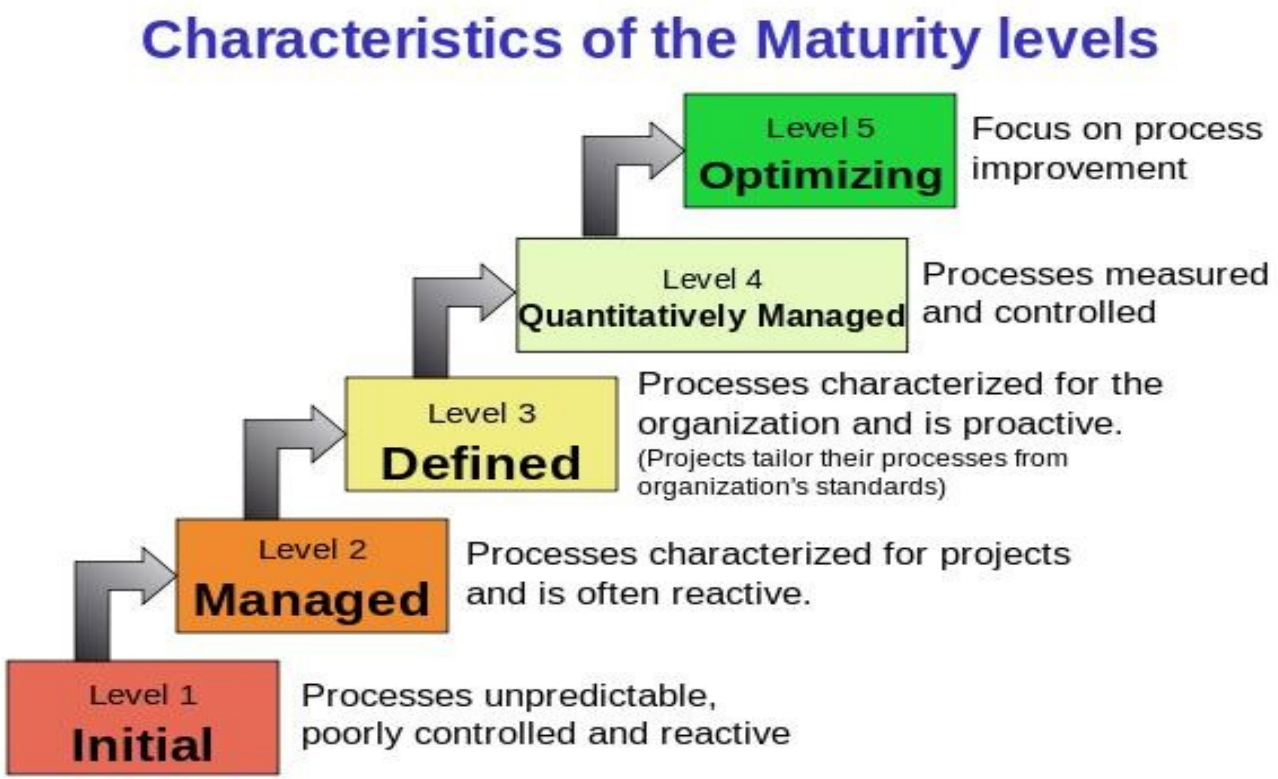

Characteristics of the Maturity levels
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

Fig 3

CMM Five Maturity Levels

- Optimized
- Managed
- Defined
- Repeatable
- Initial

CMM Five Maturity Levels

- **Initial level**
  - Processes are disorganized, even chaotic. Success is likely to depend on individual efforts, and is not considered to be repeatable, because processes would not be sufficiently defined and documented to allow them to be replicated.

- **Repeatable level**
  - Basic project management techniques are established, and successes could be repeated, because the requisite processes would have been made established, defined, and documented.

CMM Five Maturity Levels

- **Defined level,**
  - An organization has developed its own standard software process through greater attention to documentation, standardization, and integration.
- **Managed level,**
  - An organization monitors and controls its own processes through data collection and analysis.
- **Optimizing level,**
  - Processes are constantly being improved through monitoring feedback from current processes and introducing innovative processes to better serve the organization's particular needs.

CMMI Maturity Levels

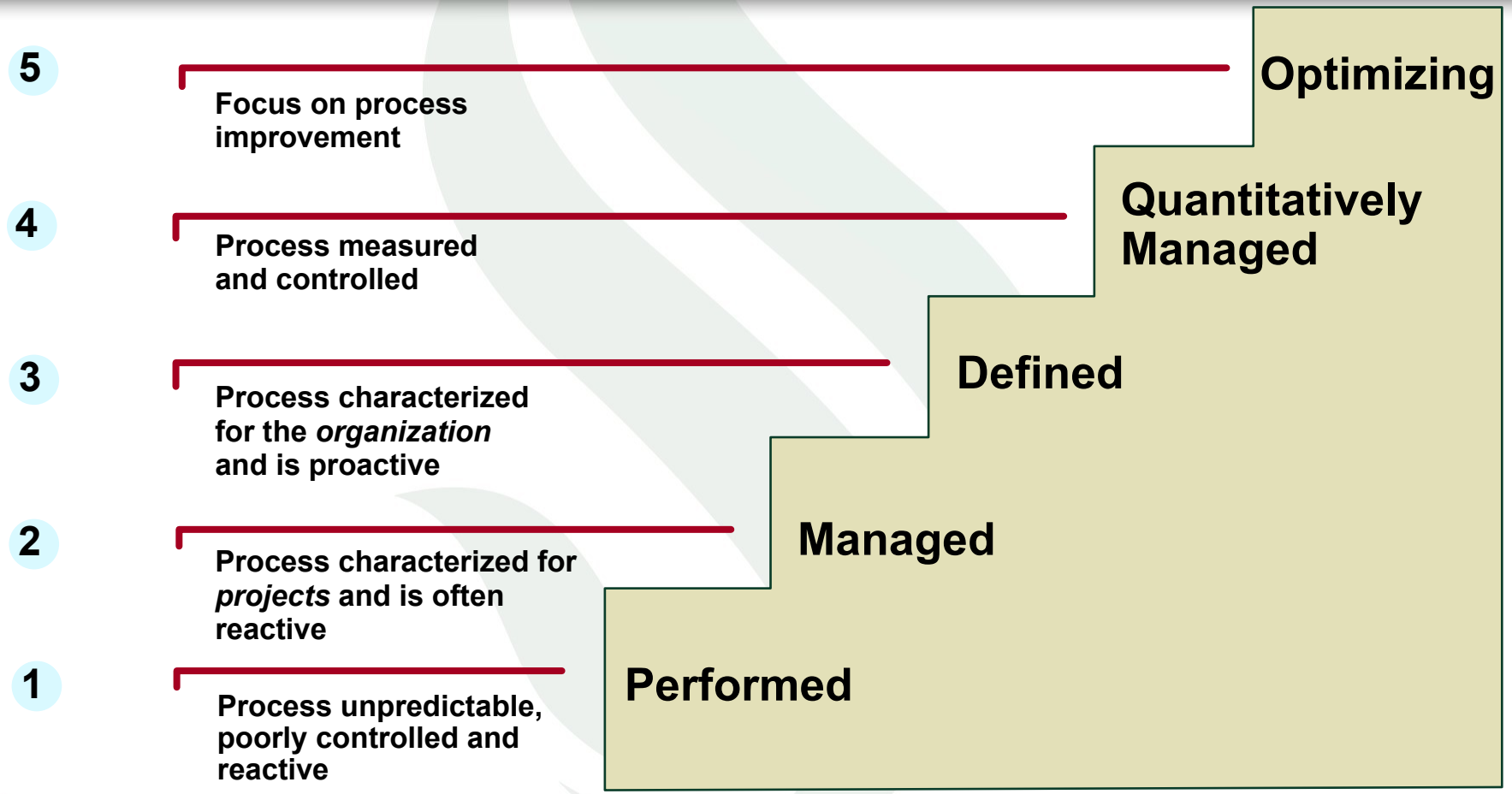


Fig 4

Software Engineering Process Models

- ✓ **Strategy** need to be developed to solve an actual problem.
- ✓ This strategy encompasses: **Process, Methods, and tools**.
- ✓ This strategy is known as a **Process Model**

- The process model is chosen based on the following:
  - **Nature** of the project
  - **Methods** and tools to be used
  - **Deliverables** that are required

Process as Problem Solving

Software development can be characterized as a **problem-solving process**

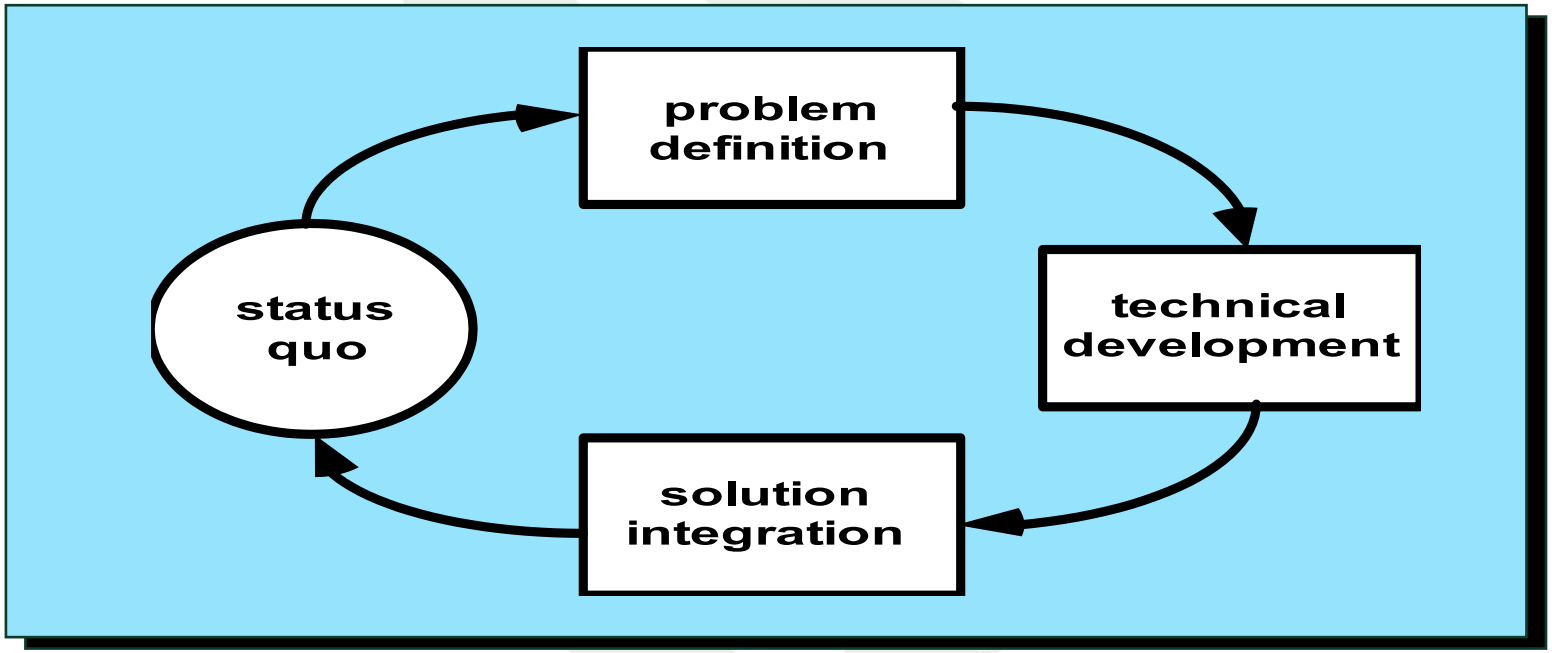


Fig 5

Variations in the Process

- ✓ Regardless of exact procedure, all broadly follow the phases of Software Developr
  - **Requirements** Engineering p
  - **Design** phase
  - **Implementation** phase
  - **Testing** phase
  - **Deployment** phase
  - **Maintenance** phase

- ✓ Some use different terms – some combine phases

Software Life-Cycle Models

- Definition

  The series of steps through which the product progresses.

- The model specifies
  - ✓ The various **phases** of the process
    - e.g., requirements, specification, design…
  - ✓ The **order** in which they are carried out

Software Lifecycle Models

- **Build-and-fix model**

  Develop systems
  - Without specs or design
  - Modify until customer is satisfied

- **Why doesn't build-and-fix scale?**
  - Changes during maintenance
  - Most expensive!

Software Lifecycle Models

Waterfall model : Also known as **Linear Sequential Model**

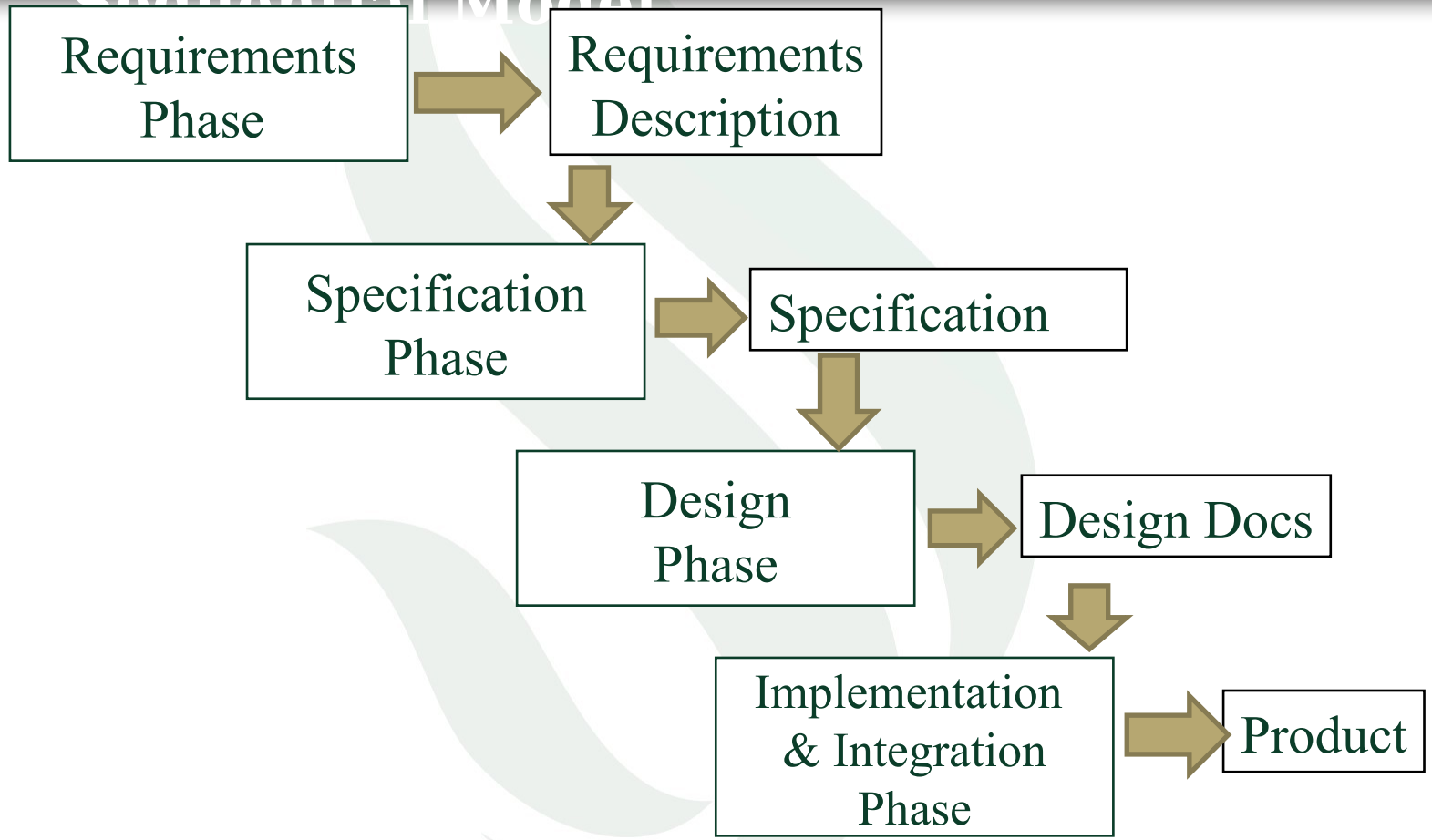


Fig 6

Drawbacks of Waterfall Model

- **Document-driven model**
  - Customers cannot understand them …
    - ✓Imagine an architect just showing you a textual spec!
  - First time client sees a working product after it has been coded. Problems here?
    - ✓Leads to products that don’t meet customers needs

Drawbacks of Waterfall Model

- **Assumes feasibility before implementation**
  - ✓ Re-design is problematic
- **Works best when you know what you're doing**
  - ✓ When requirements are stable & problem is well-known

V Model

- In software development, the V-model represents a development process that may be considered an **extension of the waterfall model**.

- Instead of moving down in a linear way, the process steps are **bent upwards after the coding phase**, to form the typical V shape.

V Model

- The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.

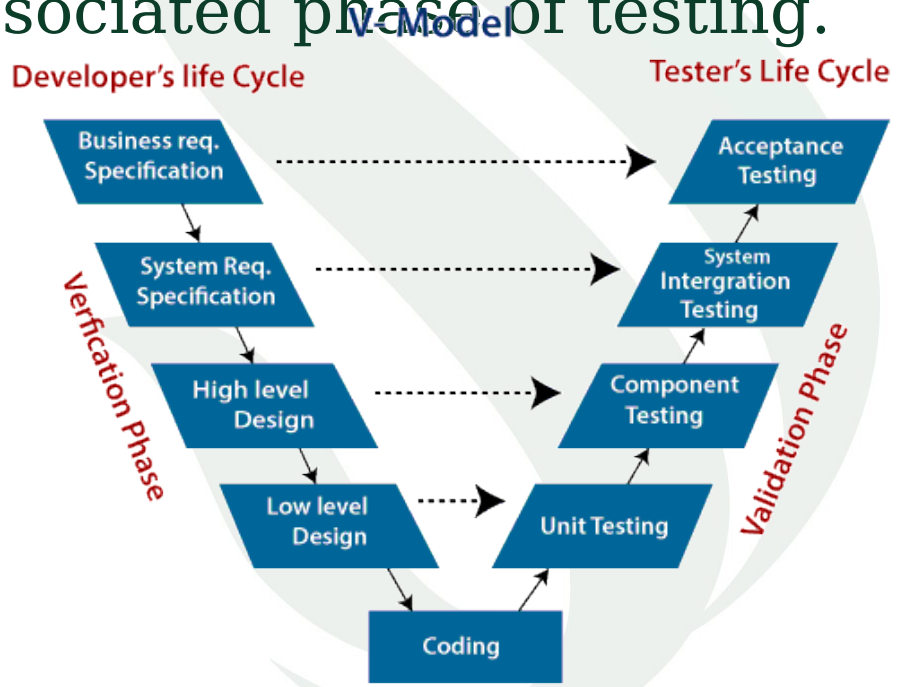


Fig 7

Prototype Model

- In software development, the prototype model represents a creation of a working prototype of a product before carrying out the development of actual software.

- In this model, a **working replicate of the end product** developed in the first place, tested and changed according to customer feedback.

Prototype Model

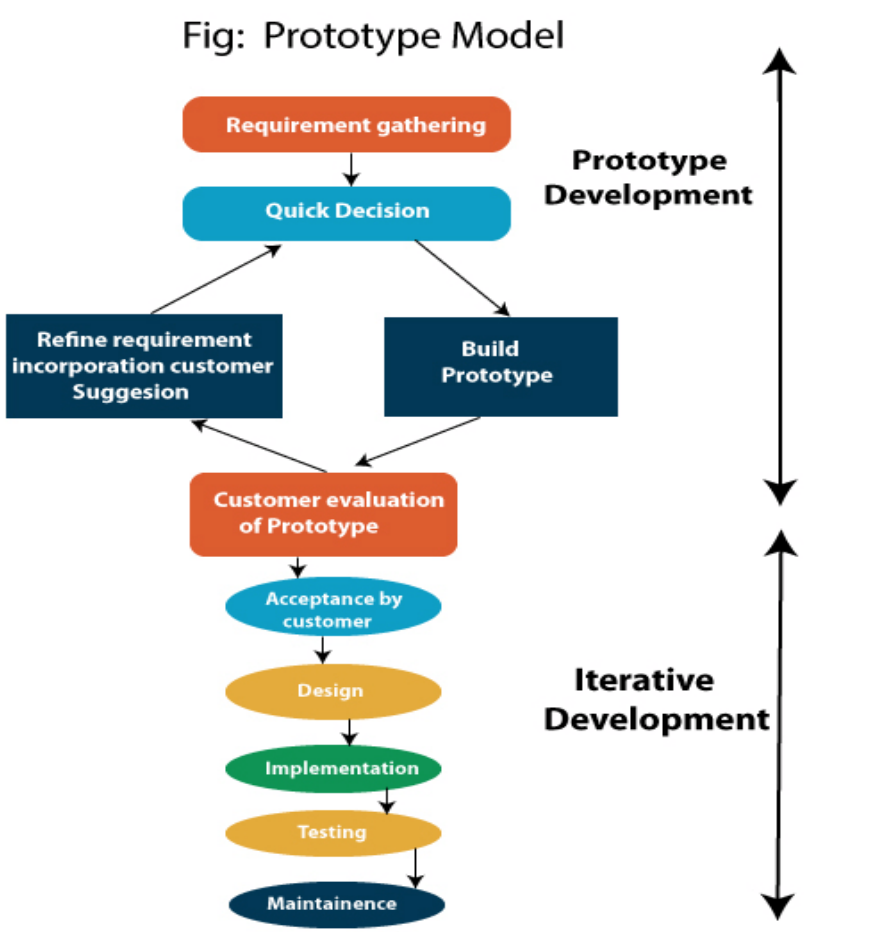


Fig. 8

Agile Model
Where it began...

Fig. 9

In 2001, seventeen software developers met at a resort in Snowbird, Utah to discuss these lightweight development methods:

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

Together they published the Manifesto for Agile Software Development

Manifesto for Agile Software Development

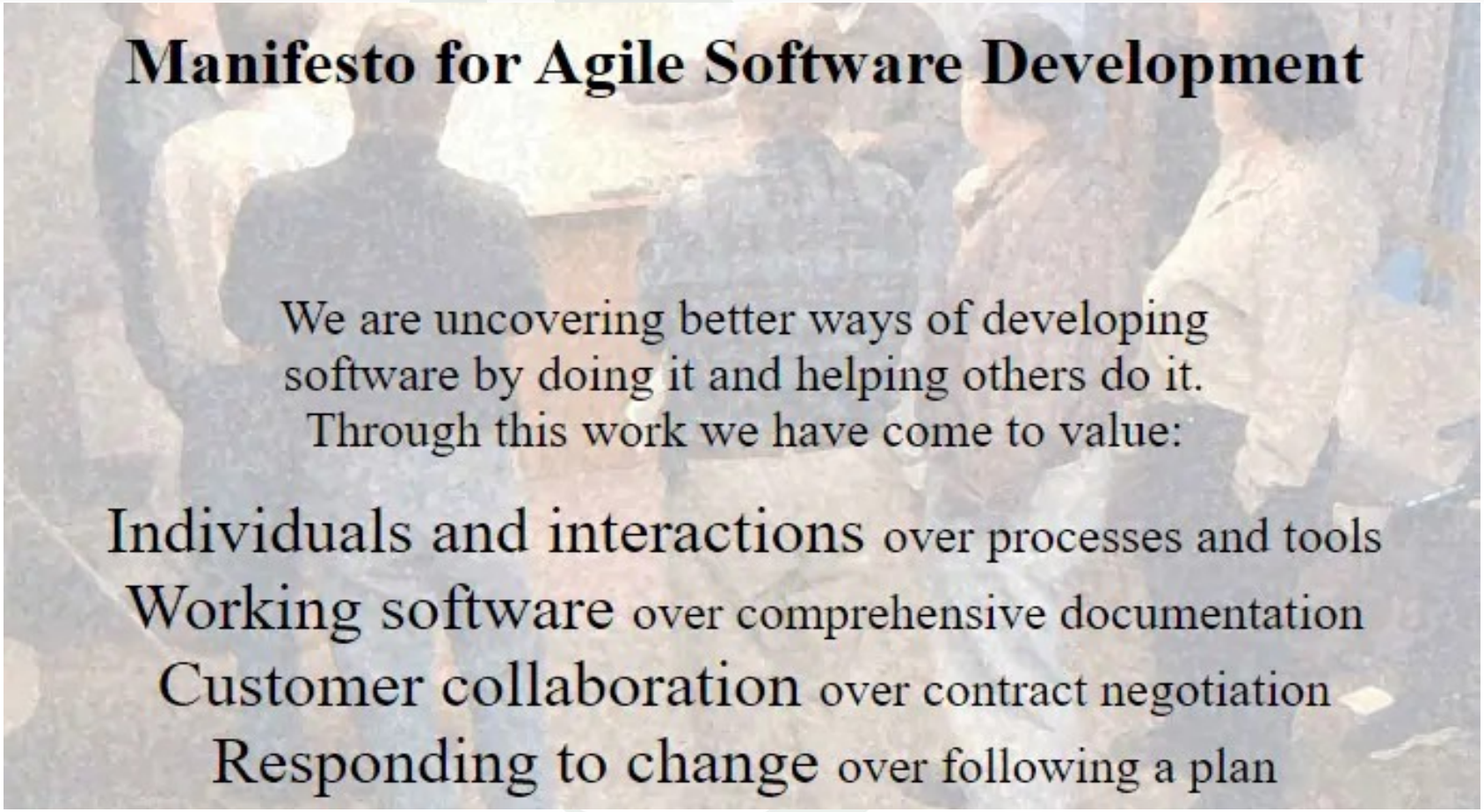


Fig. 10

AGILE MANIFESTO 2001

17 software professionals who were fed up with traditional models.

Fig. 11

AgileAlliance.org

Agile Principles

“Guiding practices that support teams in implementing and executing with agility.”

The equivalent of “generic activities” that all agile frameworks should

AgileAlliance.org

Fig. 12

GENERAL AGILE SDLC MODEL

The Generic Process for "doing" Agile.

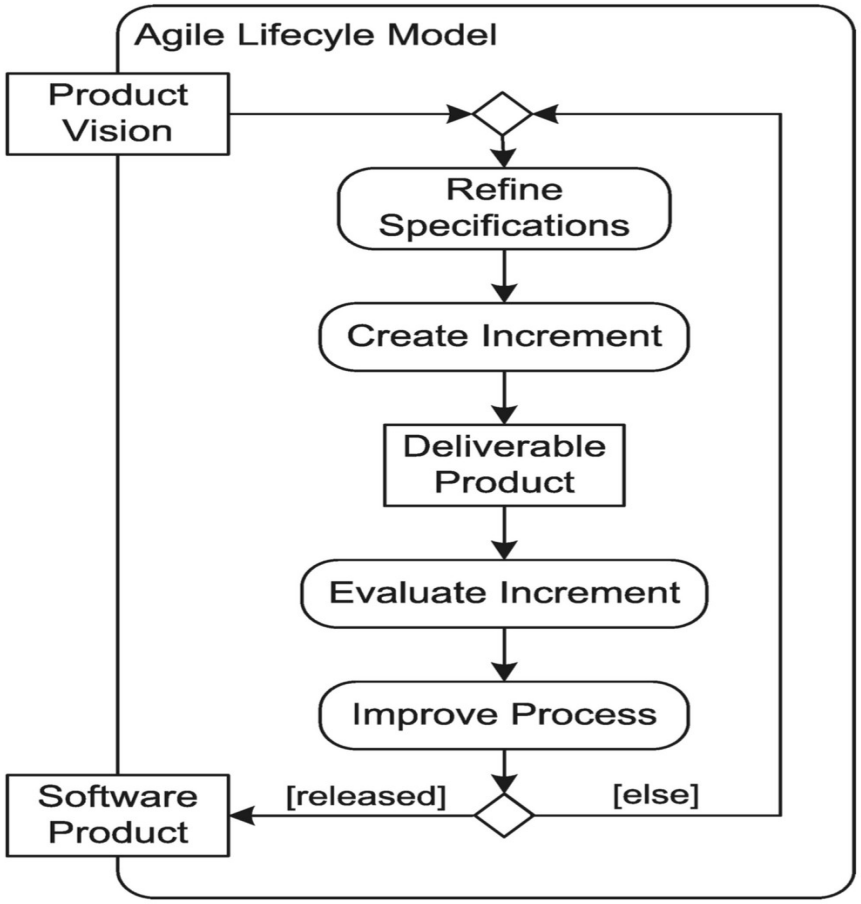


Fig. 13

Agile Advantages

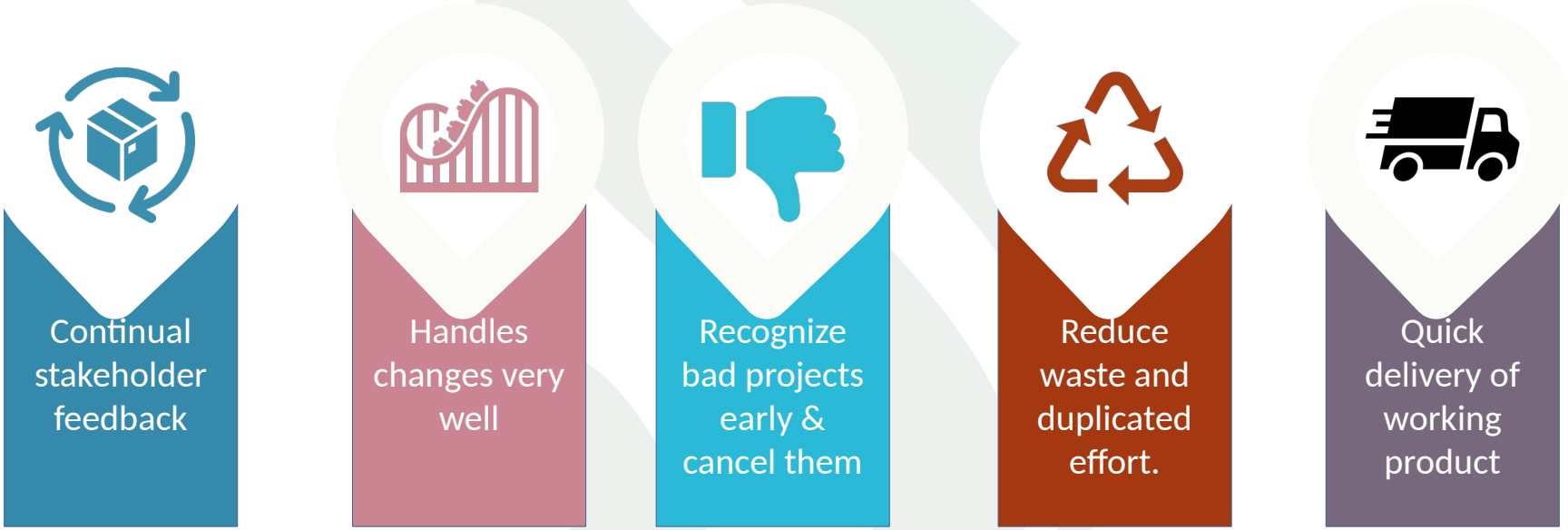


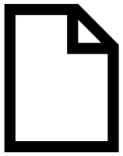
Fig. 14

Agile Concerns/Issues

Hard to predict outcomes.

"All or nothing" – incremental design falls short.

Very light on documentation



Stakeholder MUST commit ongoing time & effort

Need special techniques to scale for large projects.

Fig. 15

Agile Frameworks / Methodologies

Each framework provides a set of guidelines that, when followed, help teams achieve agility

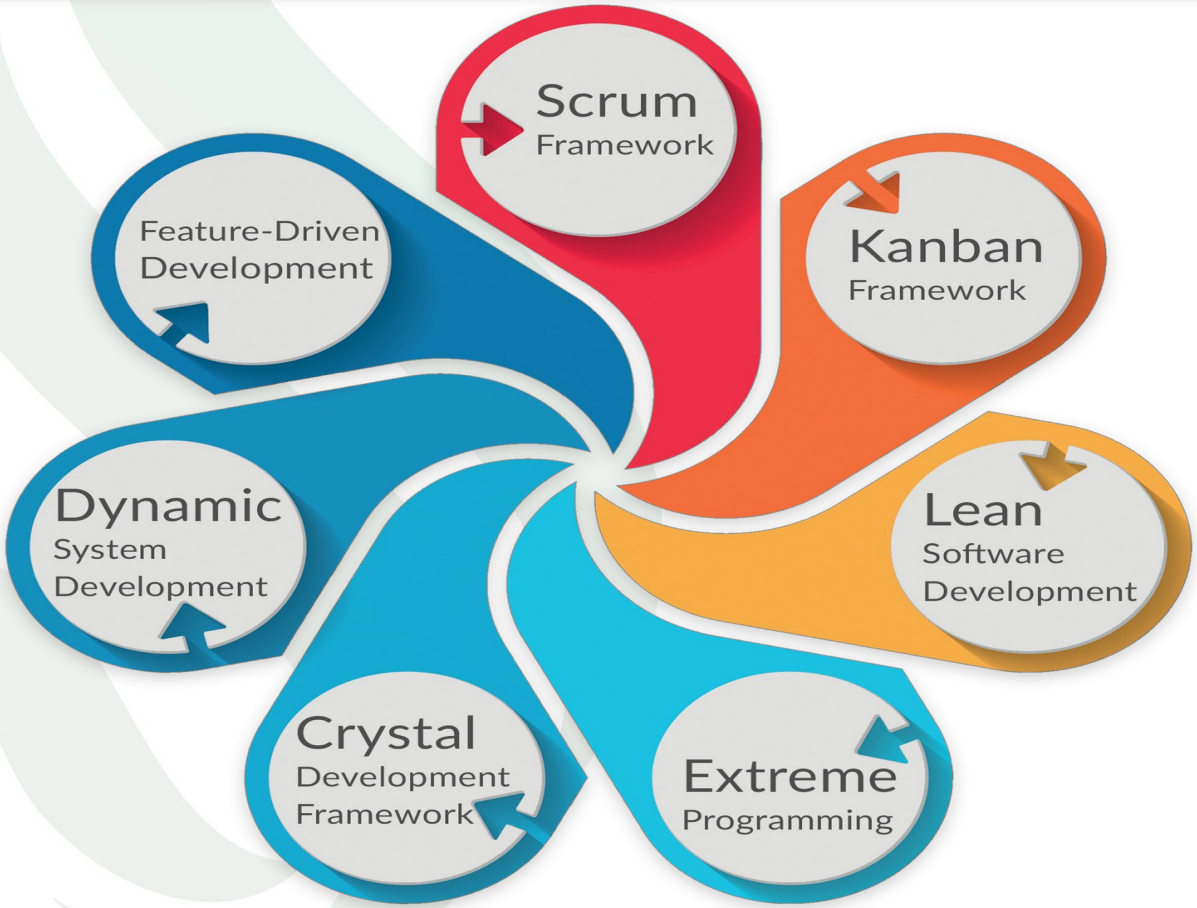


Fig. 16

SCRUM

THE MOST POPULAR AGILE FRAMEWORK

Scrum Framework

(Sutherland and Schwaber; 1995)

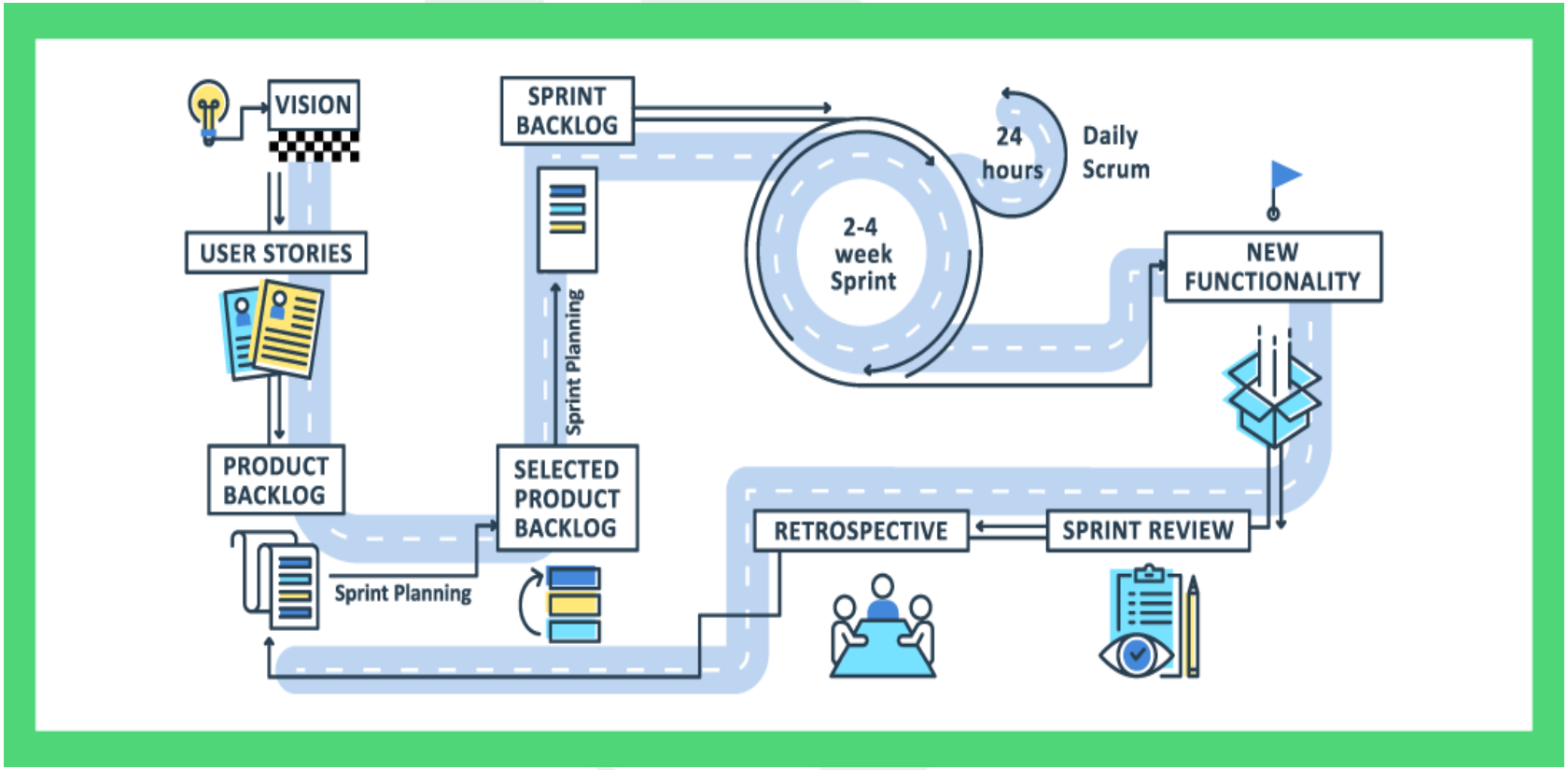


Fig. 17

Scrum Roles

Fig. 18

Product Owner

Responsible for achieving maximum business value. Face of the customer. Prioritizes stories.

ScrumMaster

Helps the Team be successful; Secures needed resources. Works with Product Owner on backlog.

Team

Multi-functional (design, code, test). Estimates work time and load.

Product Owner

Always 2 steps ahead and ready with more work for the team.

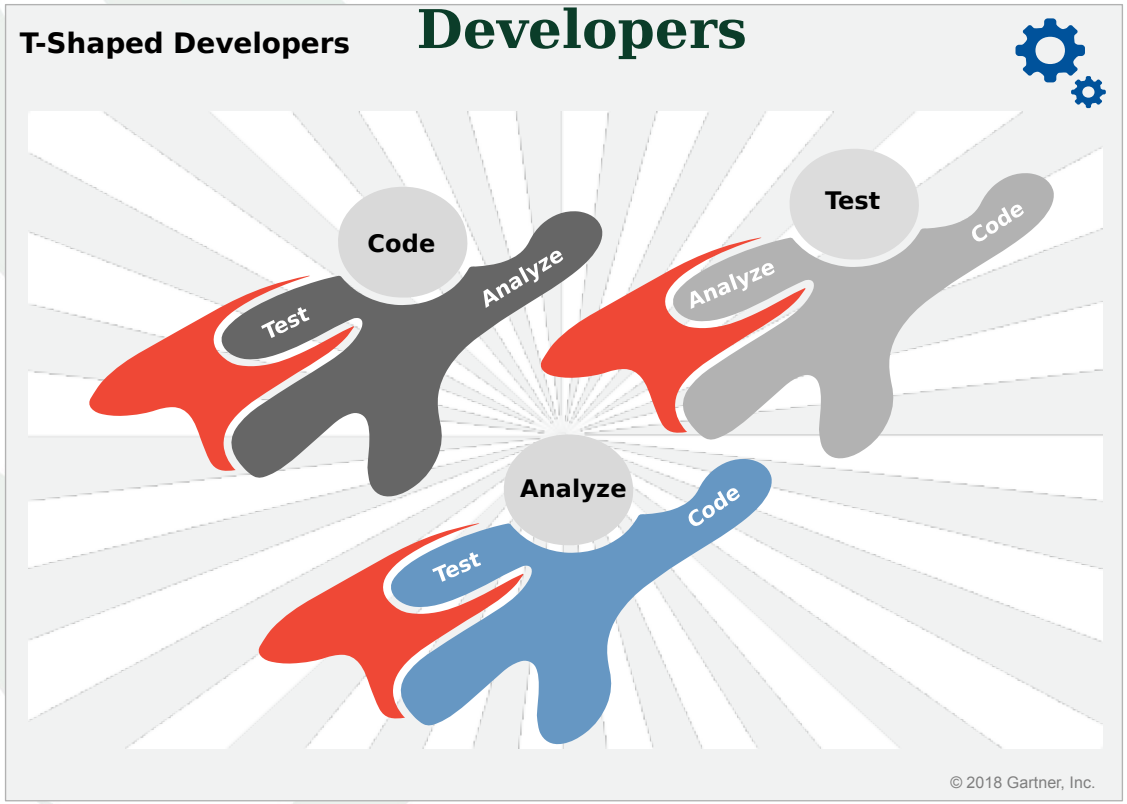
- Works with stakeholders and the Development Team to create an **ordered list of work** called a product **backlog**.
- Continuously **refines** and reorders the product backlog to drive product success.
- Discusses the sprint objective and works with the Scrum team during sprint planning to craft a sprint goal.
- Ensures that two to three sprints worth of work are refined in the product backlog to a level in which they can be developed (w/ help of Scrum Master.)

Gartner 3906724

Development Team

Developers may have an area of expertise but should be able multiple things.

The Goal Is an Agile Team of T-Shaped Developers



SACRAMENTO STATE
Redefine the Possible

Fig. 19

Scrum Master Do's

Make Scrum Master your primary role

DO ✓

1. Guide the team
2. Facilitate Scrum events
3. Be a manager of impediments
4. Protect the team (from complacency and dysfunction)
5. Encourage learning
5. Hold the team accountable
6. Support the product owner
7. Enable and empower the team
8. Promote self-organization
9. Become an agent of change

Source: Gartner
ID: 365100

Scrum Mas

Avoid additional roles that include management, project coordination or product ownership.
If holding multiple roles, consider occupying the remainder of your time as a developer.

DON'T

1. Direct the team
2. Own Scrum events
3. Be a manager of people
4. Act as a team interface
5. Be the source of all knowledge

6. Pressure the team into action
7. Prioritize or assign work
8. Become a team administrator
9. Solve team problems
10. Become the Scrum police

Source: Gartner
ID: 365100

Agile Model : Scrum

- ✓ Scrum - distinguishing features
- ✓ Development work is partitioned into “packets”
- ✓ Testing and documentation are on-going as the product is constructed
- ✓ Work occurs in “sprints” and is derived from a “backlog”
- ✓ Changes are not introduced in sprints but in backlog
- ✓ Meetings are very short (15 minutes daily)

Agile Model : Scrum

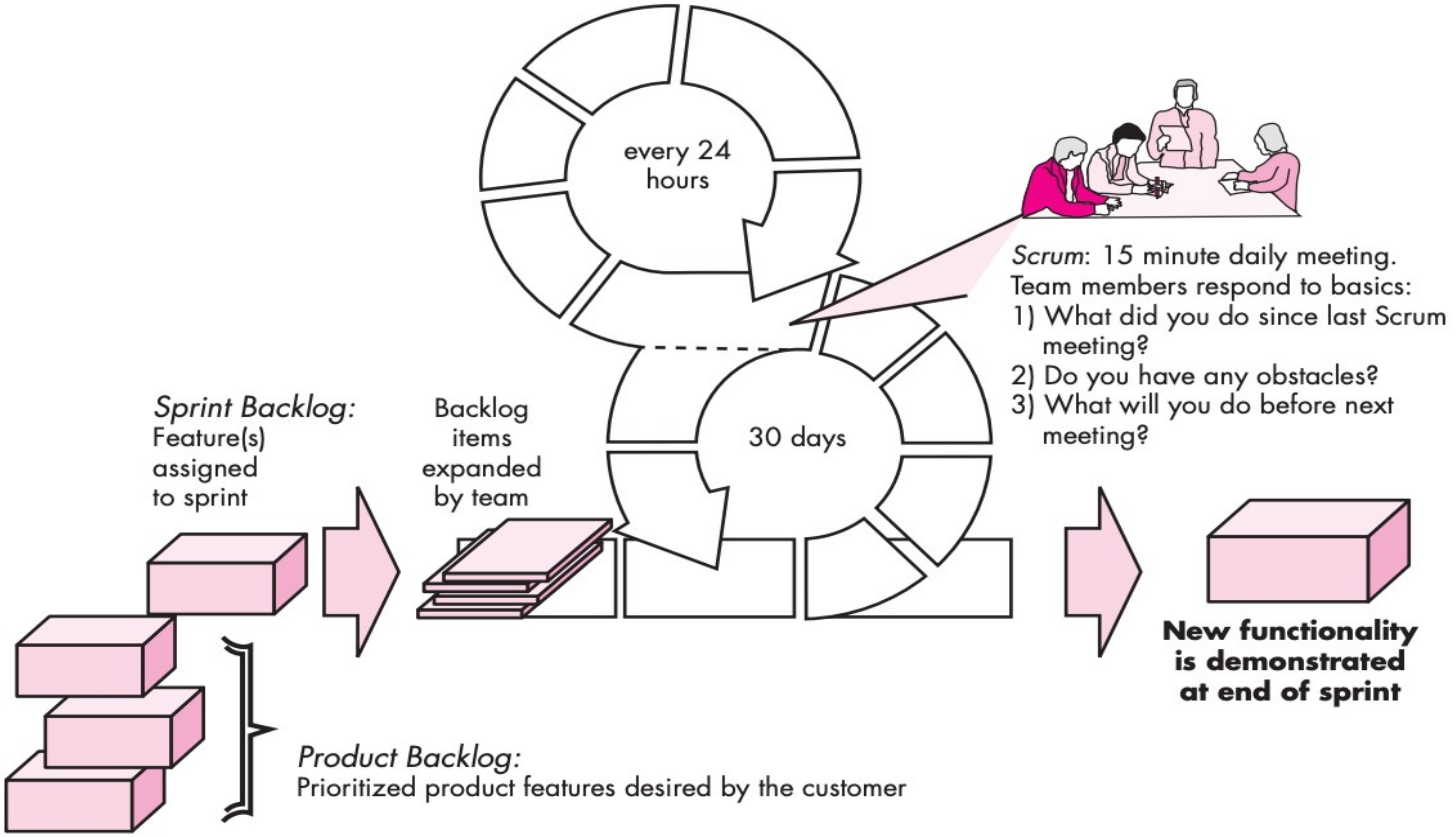


Fig. 20

Questions…?

What is next..?
Software Engineering Project Discussion

Next Topic:
System Engineering & Requirements Engineering