Software Engineering: Principles and Practices (Third Edition)

Chapter 4 Software Process Models

QUOTE:

“A software process describes what to communicate, when to communicate, how to communicate, and with whom to communicate (most process descriptions will not tell you why; this is normally left for someone to write a book about!).”

Stephen Palmer and John Felsing (Software Engineers and Authors)

Table 3-3. Relationships between Polya activities, software activities, and software tasks

Polya Activity	**Software Activity**	**Software Task**
Understanding the Problem	Analysis	Problem identification
		Problem analysis
		Scope definition
		Requirements identification
		Decision analysis
Devising a Plan	Design	Network design
		Database design
		Process design
		Input design
		User-interface design
		Output design
Carrying Out the Plan	Implementation	Construction
		Testing
		Data migration
		System installation
		System changeover
Looking Back	Maintenance	Corrective maintenance
		Perfective maintenance
		Adaptive maintenance
		Preventive maintenance
		Refactorative maintenance

What is a software process model?

- It is a framework that is used to guide the work required to initiate a software development project and to bring it to a successful completion.
- It is a generalized pattern that represents the overall plan for executing a software project from beginning to end.
- It articulates the
  - Activities and tasks that will be performed
  - Sequence and timing of those activities and tasks
  - Criteria that will be used for transitioning from one activity or task to another
  - Deliverables that will be produced at the end of the activities and tasks

What is a software process model?

- The software process model chosen for a given software project is directly influenced by the software process philosophy of the software development organization and/or the nature of the software project itself.
- Software process philosophies
  - Plan-driven
  - Agile
  - Component-based

Plan-Driven Software Process Philosophy

- The plan-driven software process philosophy embraces the idea of developing software systems according to a plan.
- This philosophy is process centric in that it relies heavily on following a repeatable, yet constantly improving, process.

Plan-Driven Principles

- Static Software Process
- Project Milestones
- Stable Requirements (relatively)
- Written Documentation
- Conventional Human Resource Allocation
- Custom Development

Agile Software Process Philosophy

- The agile software process philosophy embraces the idea of developing software systems in a flexible and adaptable way. This philosophy is product centric in that it relies heavily on delivering the right product in the right way at the right time.
- According to the *Manifesto for Agile Software Development*, the agile philosophy values individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [32]

Agile Principles

- Customer Satisfaction
- Changing Requirements
- Frequent Delivery
- Progress
- Sustainable Development
- Client Cooperation
- Face-to-Face Communication
- Trust
- Good Design
- Minimal Work
- Self-Organizing Teams
- Adaptation

Component-Based Software Process Philosophy

- The component-based software process philosophy embraces the idea of developing software systems through the assembly and integration of existing software components.
- This philosophy is component centric in that it relies heavily on the effective union of pre-made software components.

Component-Based Principles

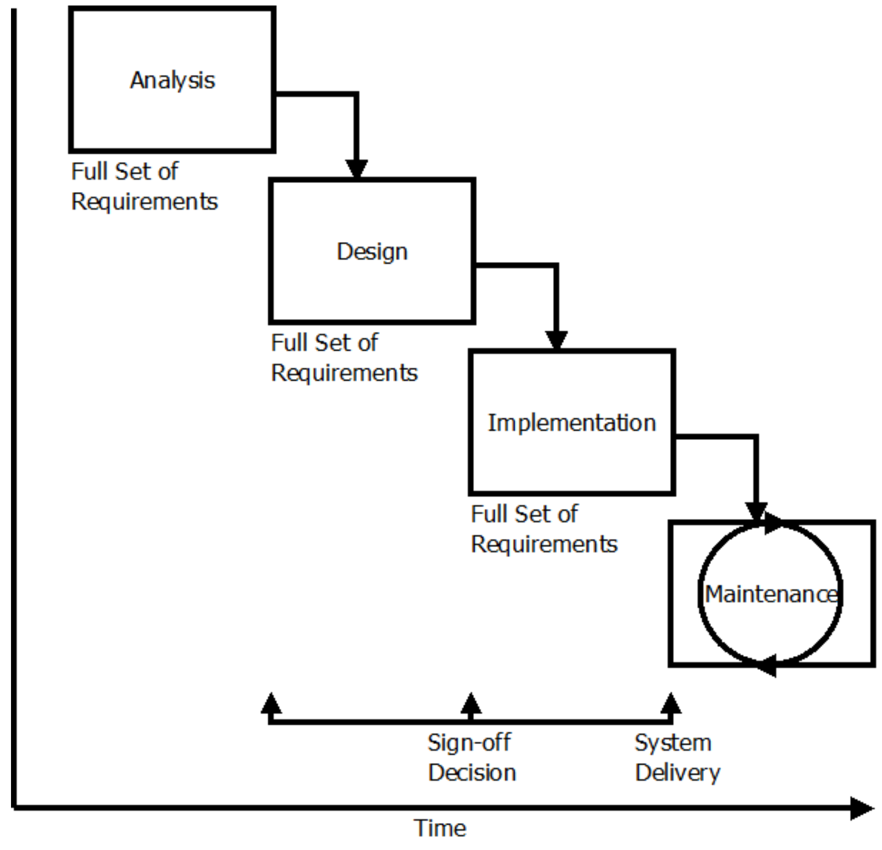
- Reuse
- Communication
- Substitution

Specific Process Models

- Sequential
- Iterative

Sequential Process Models

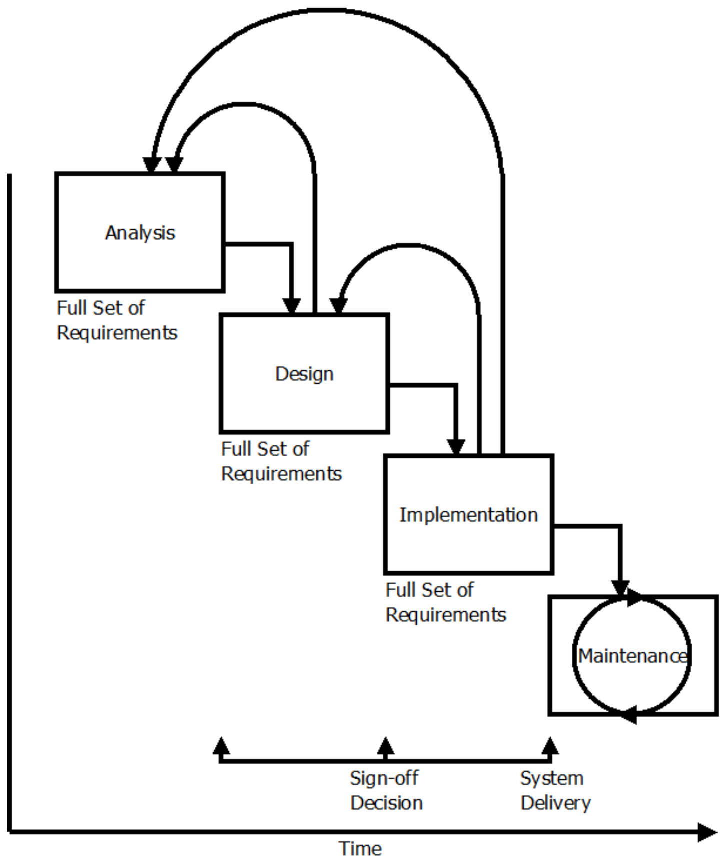
Figure 4-1. Waterfall process model



Waterfall process model

- Advantages
  - This model is relatively easy to understand and implement
- Disadvantages
  - Since analysis and design are done early in the SDLC, this model virtually guarantees the redesign and redevelopment of functionality later
  - Since the system is delivered at the end of implementation, some serious analysis and design defects may not be uncovered until the system is in use

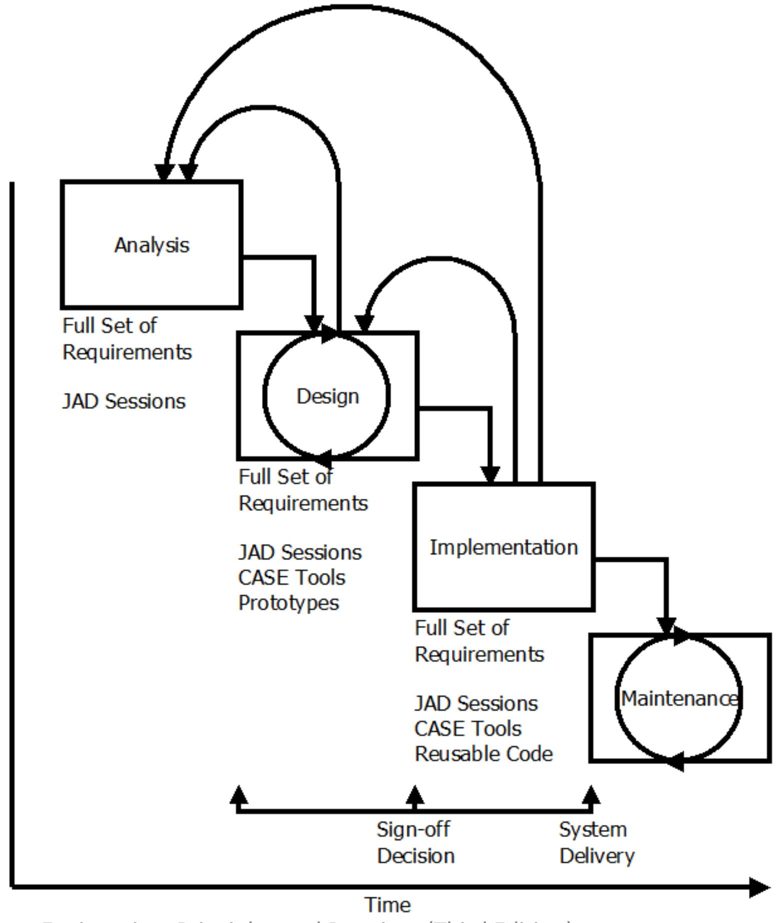
Figure 4-2. Fountain process model



Fountain process model

- Advantages
  - This model is relatively easy to understand and implement
  - Since this model permits the return to previous software activities when necessary, software defects can be fixed earlier in the SDLC
- Disadvantages
  - Since this model permits the return to previous software activities when necessary, it may be difficult to determine the end of project milestones
  - If not managed carefully, this model can devolve into an analyze-some, design-some, implement-some software process similar to the build and fix process model

Figure 4-3. Rapid application development process model



Rapid application development process model

- Advantages
  - Since analysis is done via facilitated JAD sessions, requirements gathering can be done relatively efficiently (i.e., the requirements can be gathered quickly) and relatively effectively (i.e., the correct requirements can be gathered)
  - Since CASE tools are used to generate design models, such as data models, process models, and user-interface prototypes, it is relatively easy to make changes to the requirements
  - Since end users are involved throughout the design activity via the prototyping process, there is a relatively high probability of end-user ownership and acceptance of the system
  - Since user interfaces are prototyped and code is generated, the client has the chance to see semi-working functionality relatively early in the software process thus identifying analysis and design flaws before they become expensive to correct
  - Since CASE tools are used to generate a significant proportion of the system's code, the time required to perform some implementation tasks, such as coding and testing, can be significantly reduced

Rapid application development process model

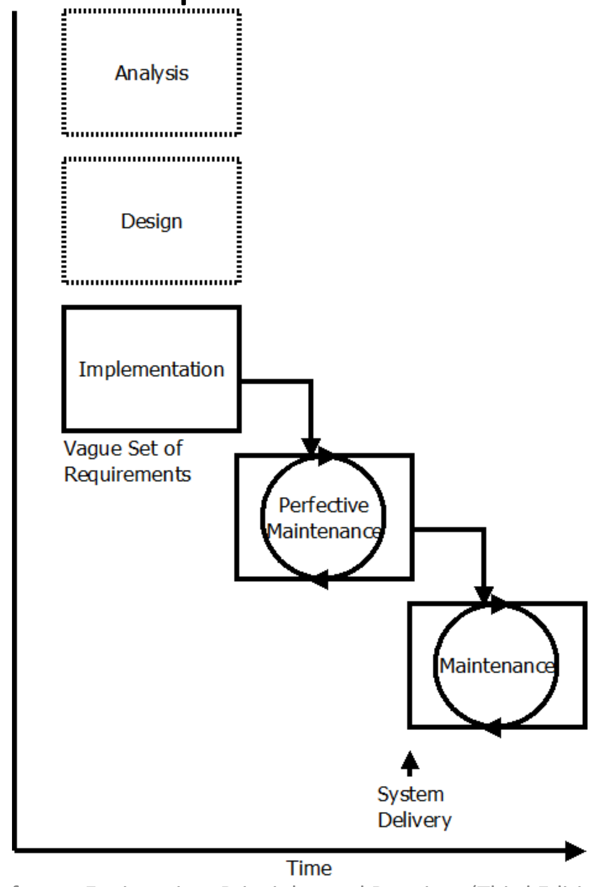
- Disadvantages
  - Since it is often necessary to get immediate clarification on requirements from end users outside of JAD sessions, breakdowns in team culture can arise due to overstepping the process
  - Since there is a strong dependence on collaborative teams within JAD sessions, breakdowns in team culture can negatively affect the software development process
  - Since JAD requires consensus decision making, it can sometimes be difficult to come to agreement in terms of software requirements and their priorities
  - Since RAD relies heavily on code generation and code reuse, this process model may not be well suited to highly customized systems

General advantages and disadvantages of the sequential process models

- Advantages
  - Since strong emphasis is placed on careful analysis and design, these process models work well for systems in which high quality is of major importance
  - Since these process models are highly structured, they reinforce the good software development practice of analysis-driven design and design-driven implementation
- Disadvantages
  - Since implementation is done relatively late in the SDLC, the client does not have the chance to see working functionality until late in the process
  - Since considerable emphasis is placed on upfront analysis, project progress can be delayed by analysis paralysis
  - Since these models are relatively inflexible by nature, they may not work well for projects that are exploratory in nature—that is, projects that do not have well-defined requirements

Iterative Process Models

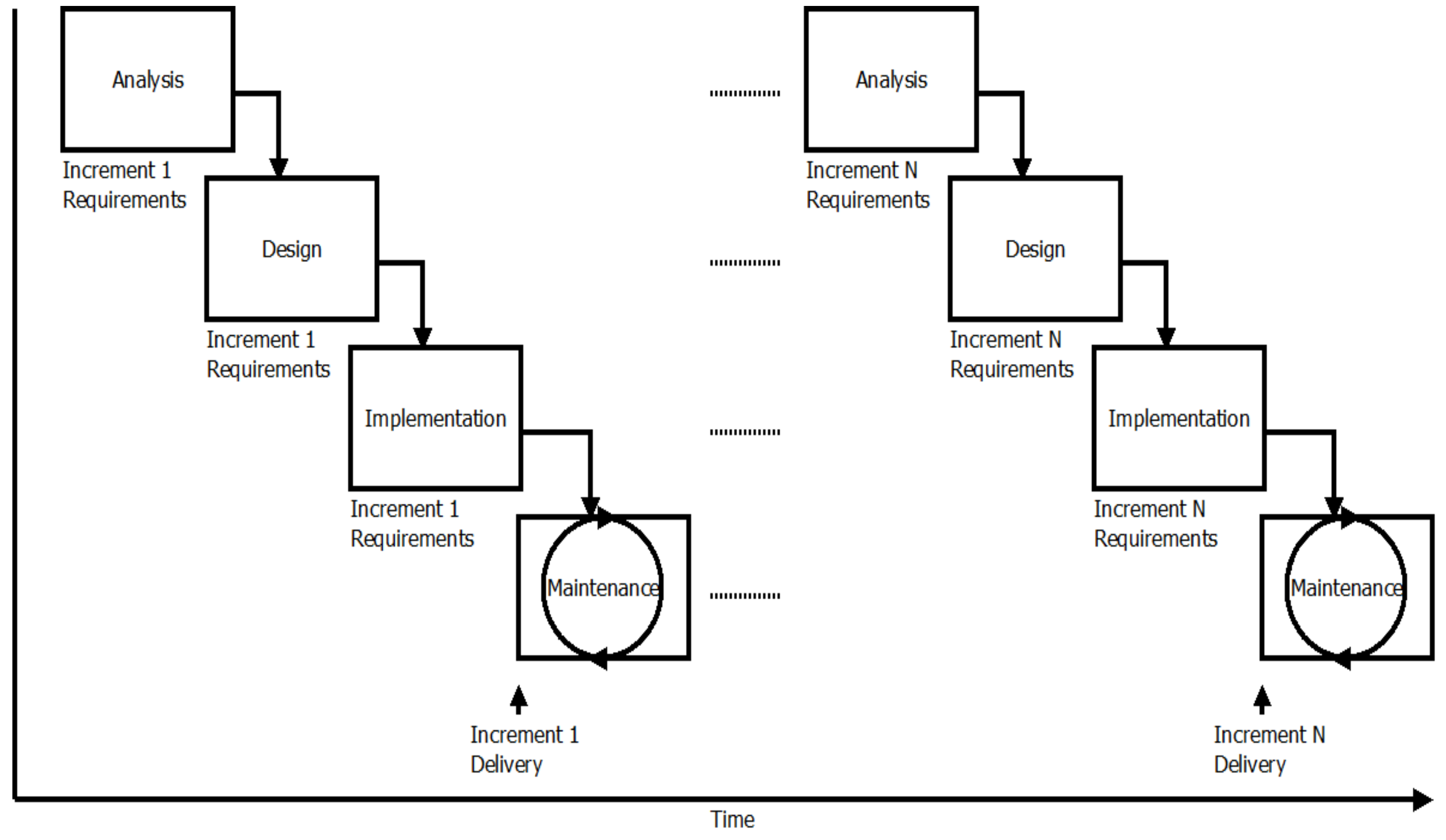
Figure 4-4. Build and fix process model



Build and fix process model

- Advantages
  - No training in a software develop process is required
  - Since little to no upfront analysis and design is done, this model is relatively cost-effective in the short run
- Disadvantages
  - Since little to no upfront analysis and design is done, this model leads to systems that require a good deal of maintenance
  - Since little to no upfront planning is done, it is difficult to predict a reasonably accurate delivery date
  - Since formalized testing is not planned or stressed, this model usually results in systems of relatively poor quality

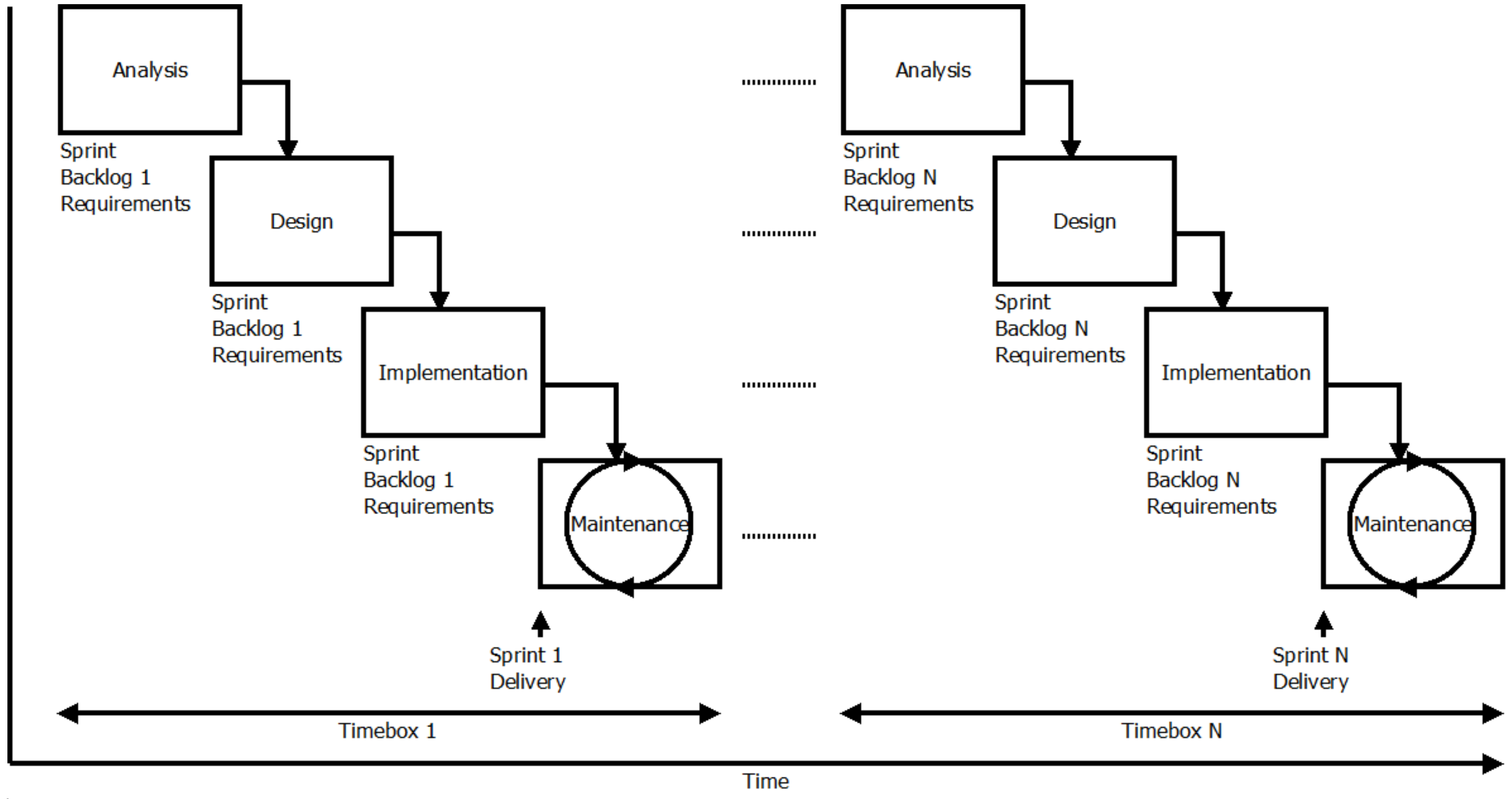
Figure 4-5. Incremental process model



Incremental process model

- Advantages
  - N/A
- Disadvantages
  - If not managed carefully, this model can easily degrade into a build and fix process model

Figure 4-6. Scrum process model

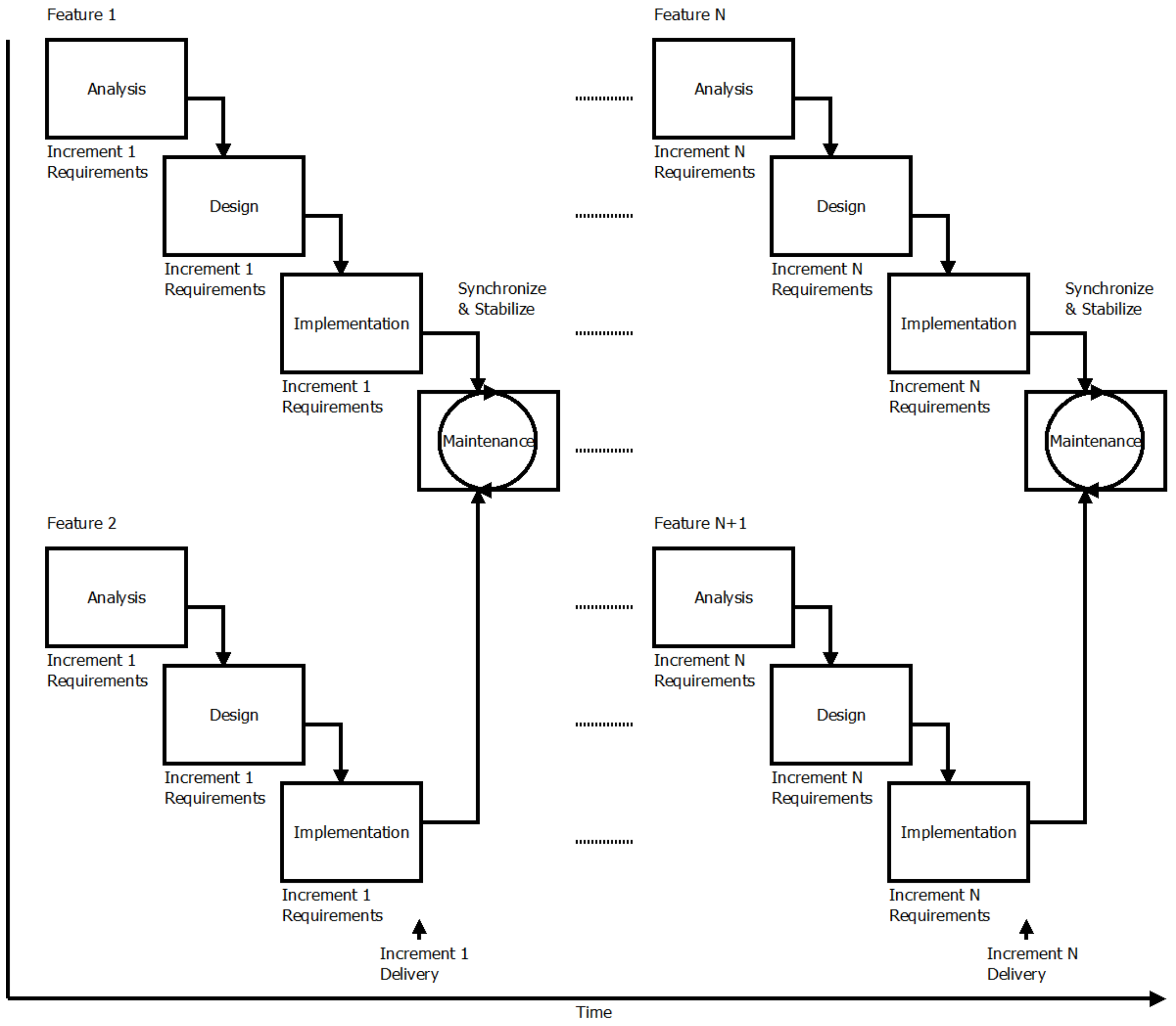


Analysis
Sprint Backlog 1 Requirements
Design
Sprint Backlog 1 Requirements
Implementation
Sprint Backlog 1 Requirements
Maintenance
Sprint 1 Delivery
Timebox 1
Analysis
Sprint Backlog N Requirements
Design
Sprint Backlog N Requirements
Implementation
Sprint Backlog N Requirements
Maintenance
Sprint N Delivery
Timebox N
Time

Scrum process model

- Advantages
  - N/A
- Disadvantages
  - Since some members of the development team may have other responsibilities within their own organizations, it can be difficult to keep them focused on matters relating to the current sprint
  - Since there are multiple people involved in the process of defining sprints, it may be difficult to prioritize the addition of new requirements and the modification of existing ones

Figure 4-7. Synchronize and stabilize process model


Feature 1
Analysis
Increment 1 Requirements
Design
Increment 1 Requirements
Implementation
Increment 1 Requirements
Synchronize & Stabilize
Maintenance
Feature 2
Analysis
Increment 1 Requirements
Design
Increment 1 Requirements
Implementation
Increment 1 Requirements
Increment 1 Delivery
Feature N
Analysis
Increment N Requirements
Design
Increment N Requirements
Implementation
Increment N Requirements
Synchronize & Stabilize
Maintenance
Feature N+1
Analysis
Increment N Requirements
Design
Increment N Requirements
Implementation
Increment N Requirements
Increment N Delivery
Time

Synchronize and stabilize process model

- Advantages
  - Since software requirements are relatively loosely defined, modifications to these requirements are permitted at any time during the project if they lead to an improved end-user experience
  - Since changes to requirements can be made at any time during the software development process, the system can be responsive to changes in the market
  - Since concurrent analysis, design, and implementation of features is performed, the project can be completed in less time
- Disadvantages
  - Since software requirements are relatively loosely defined early in the software development process, and since modifications to these requirements are permitted at any time during the project, the complete system specification is only available at the time of the system's delivery

General advantages and disadvantages of the iterative process models

- Advantages
  - Since only a core set of software requirements must be known upfront, less upfront analysis and design is required
  - Since the system is designed and developed in relatively small increments, the client has the chance to see working functionality early in the software process
  - Since each increment produces working functionality, the system can be placed into production immediately after its first increment is released
  - Since the system is introduced into the client's organization gradually via increments, the potential negative effects of imposing a new system all at once are reduced
  - Since the system is delivered in increments, end users have the opportunity to imagine and suggest additional features as they use the system
  - Since system increments provide additional functionality, project progress is highly visible
  - Since the system is delivered in increments, functionality can be implemented as time and money permit
  - Since continuous integration testing is performed, integration testing is relatively easy compared to the sequential process models in which integration testing is done all at once

General advantages and disadvantages of the iterative process models

- Disadvantages
    - Since new functionality is continually being added to the system, defects can easily be introduced that affect current functionality
    - Since new functionality is continually being added to the system, the client may be inclined to ask for additional functionality that was not originally agreed to, thus leading to scope creep

Activity

- In pairs, write 1-2 paragraphs on which software process model you prefer to work under and why (this may be from experience or from learning about them). It’s ok if you prefer different models, just mention both. Mention specific advantages or situations. Mention which role you desire. Post the text with both partners’ names to Piazza as a note.