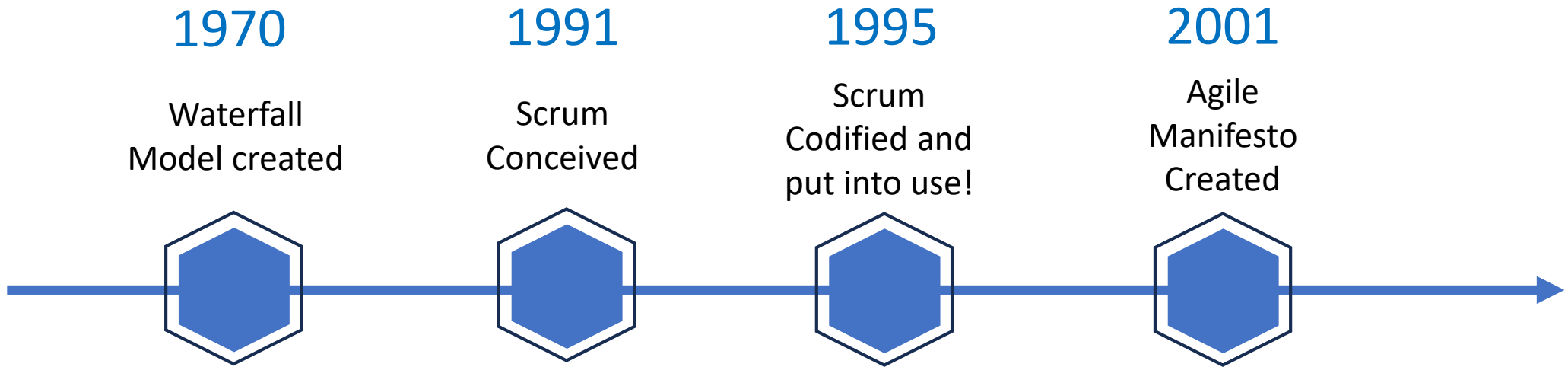




# Software Engineering

From Waterfall to Agile

# Waterfall to Agile



# SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

*Chairman: Professor Dr. F. L. Bauer*

*Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms*

Editors: Peter Naur and Brian Randell



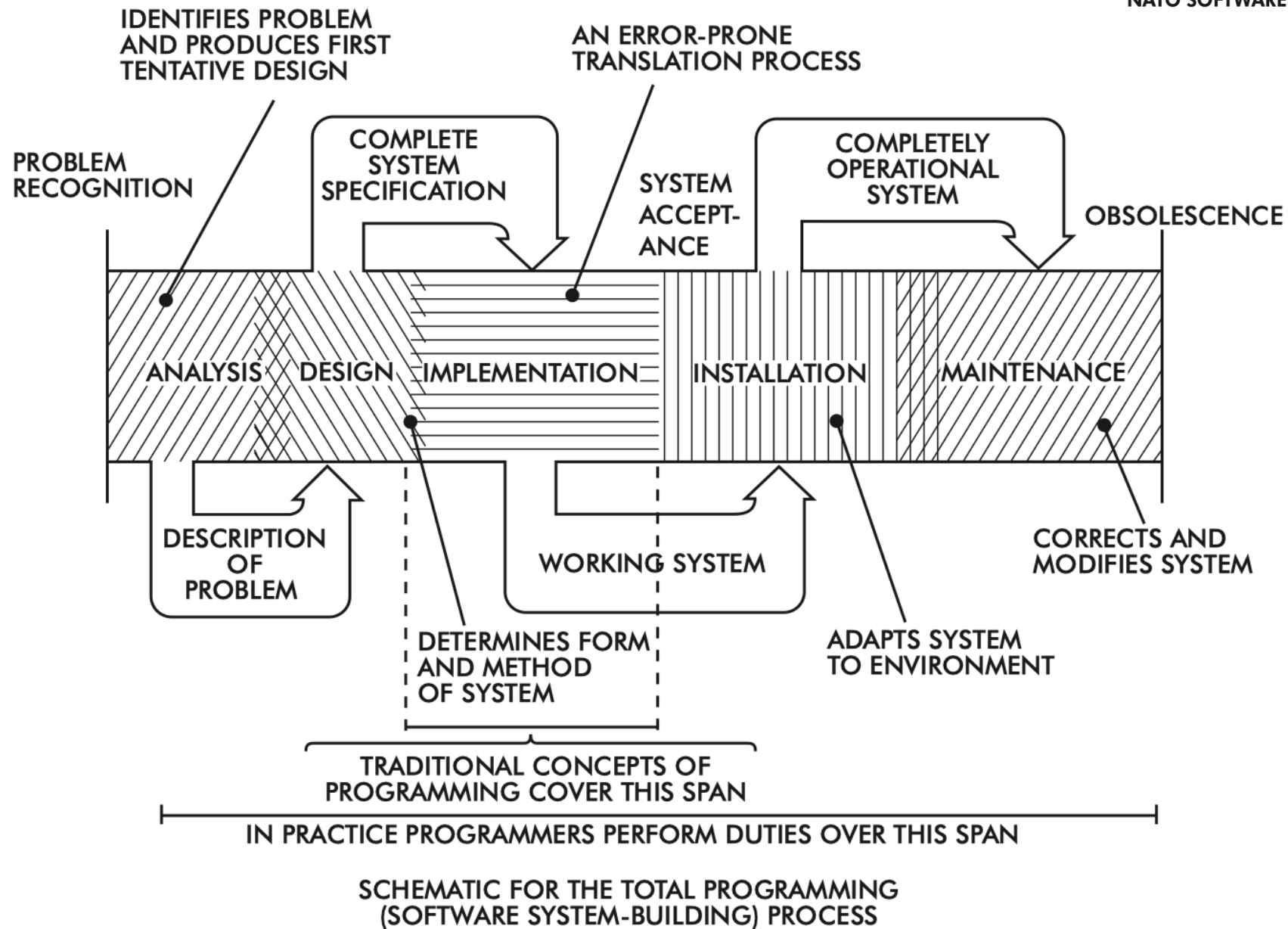
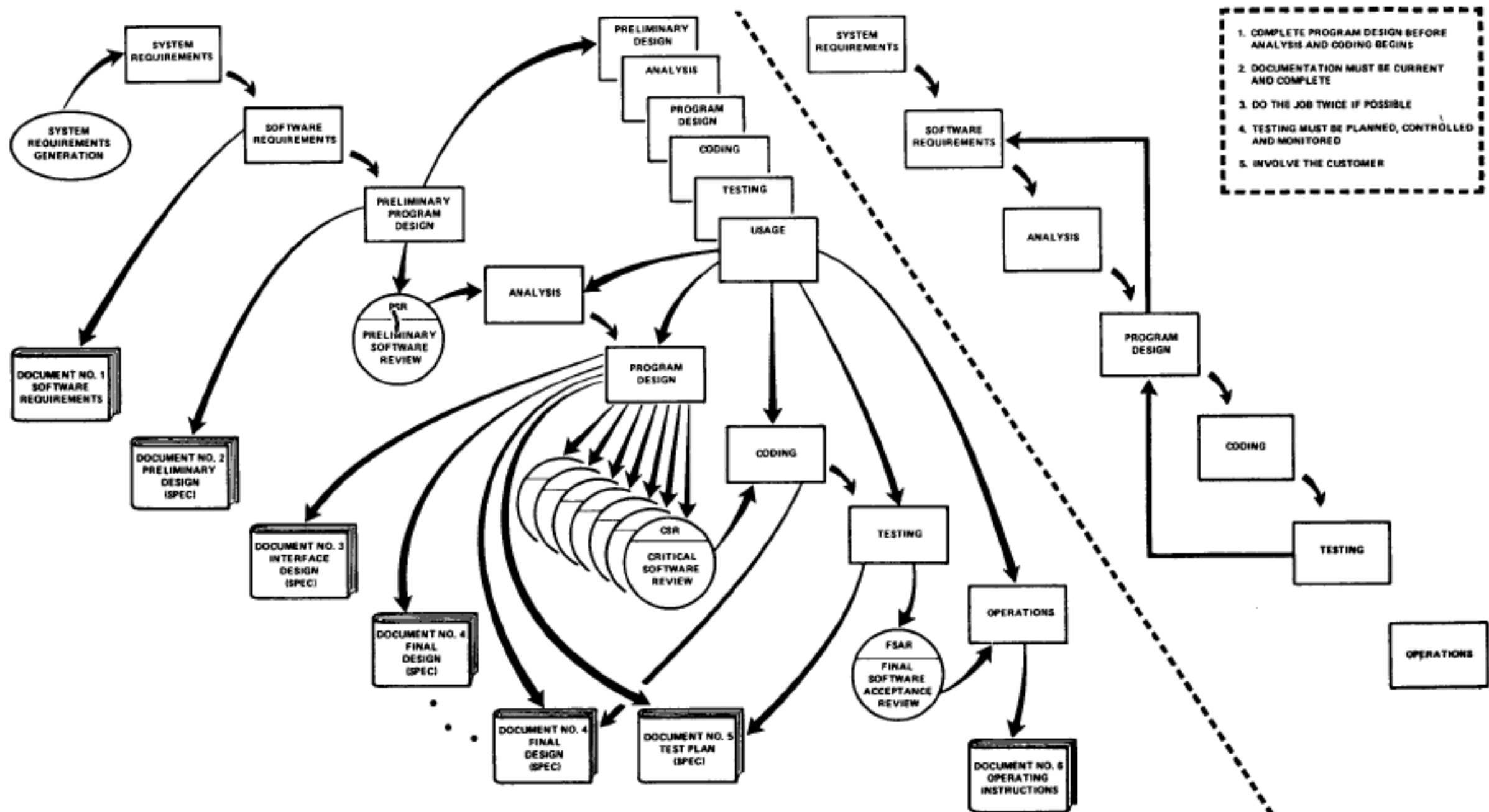
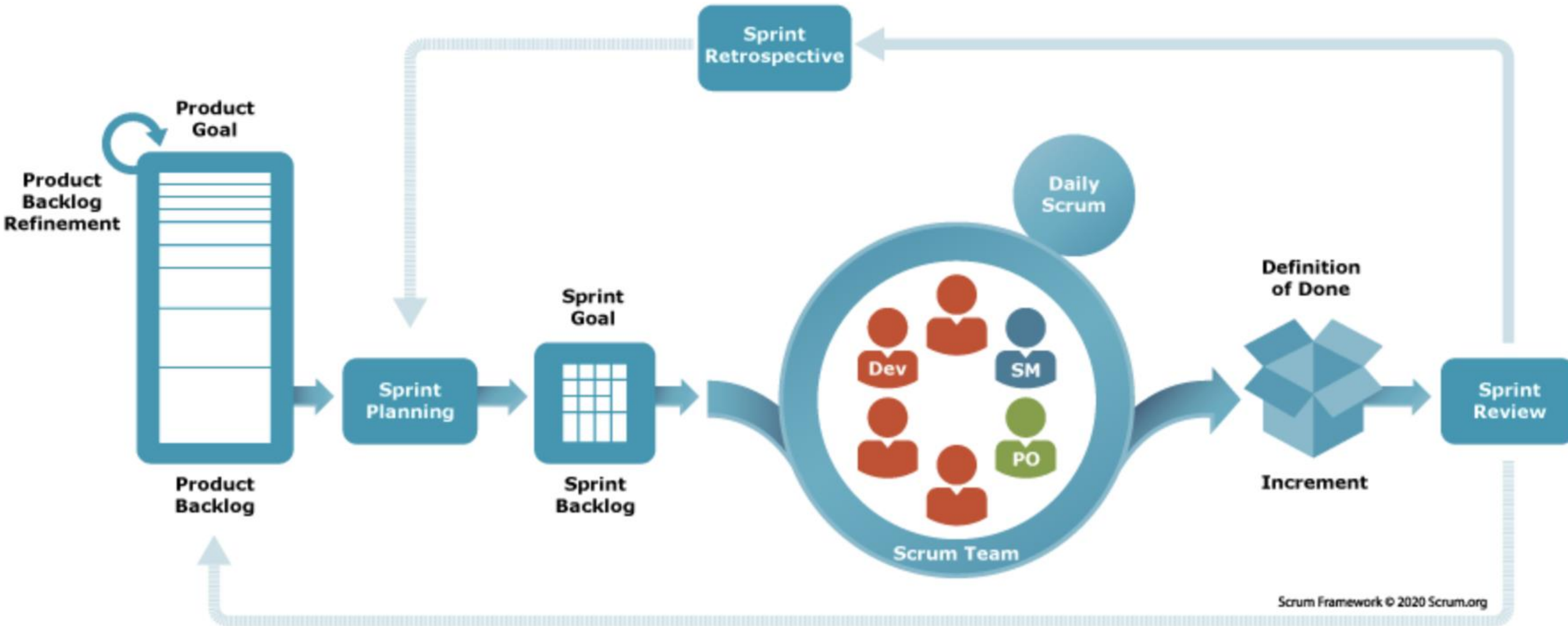


Figure 2. From Selig: Documentation for service and users. Originally due to Constantine.



1970 Winston W. Royce's Final model, published in 'Managing the Development of Large Software Systems'





# 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:

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  
Mike Beedle  
Arie van Bennekum  
Alistair Cockburn  
Ward Cunningham  
Martin Fowler

James Grenning  
Jim Highsmith  
Andrew Hunt  
Ron Jeffries  
Jon Kern  
Brian Marick

Robert C. Martin  
Steve Mellor  
Ken Schwaber  
Jeff Sutherland  
Dave Thomas



## Principles behind the Agile Manifesto

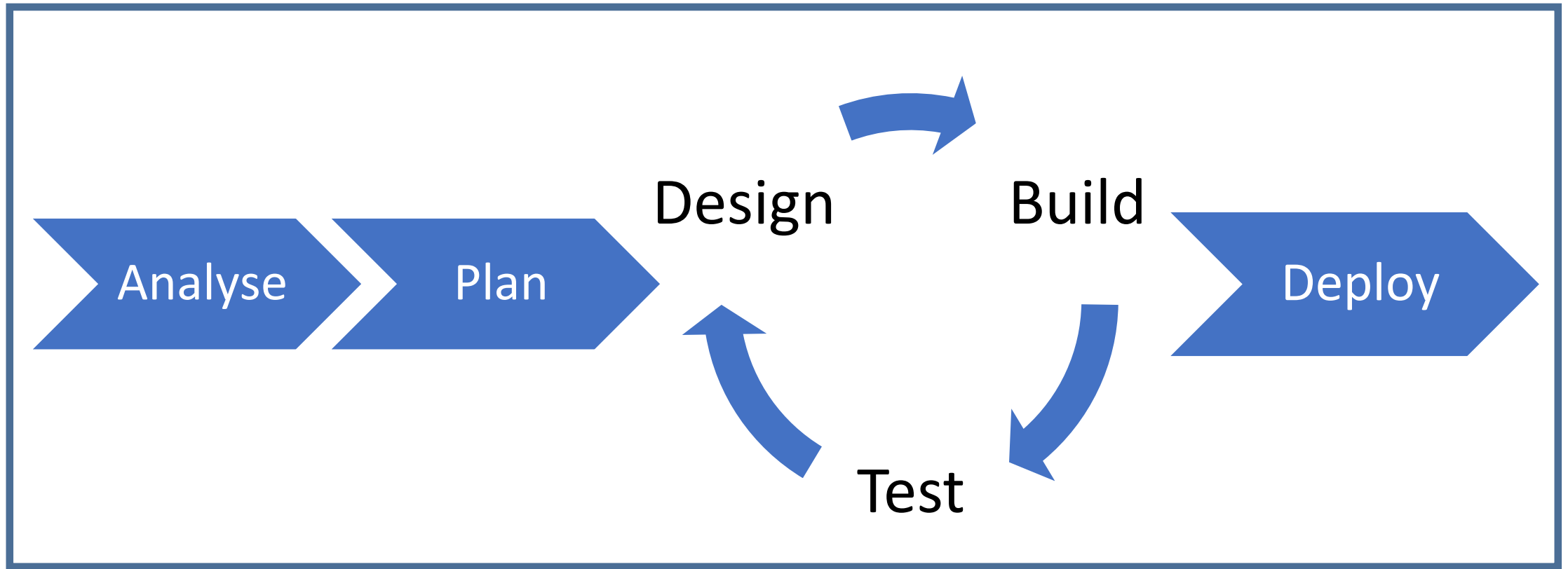
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



## Principles behind the Agile Manifesto

1. Our **highest priority** is to satisfy the customer through **early and continuous delivery** of **valuable software**.
2. Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a **couple of weeks** to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the **environment** and support they need, and **trust them to get the job done**.
6. The **most efficient and effective method** of conveying information to and within a development team is face-to-face conversation.
7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. **Continuous attention** to technical excellence and **good design enhances agility**.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At **regular intervals**, the **team reflects** on how to become more effective, then tunes and **adjusts its behavior accordingly**.

# Agile implementation: a typical 'sprint'

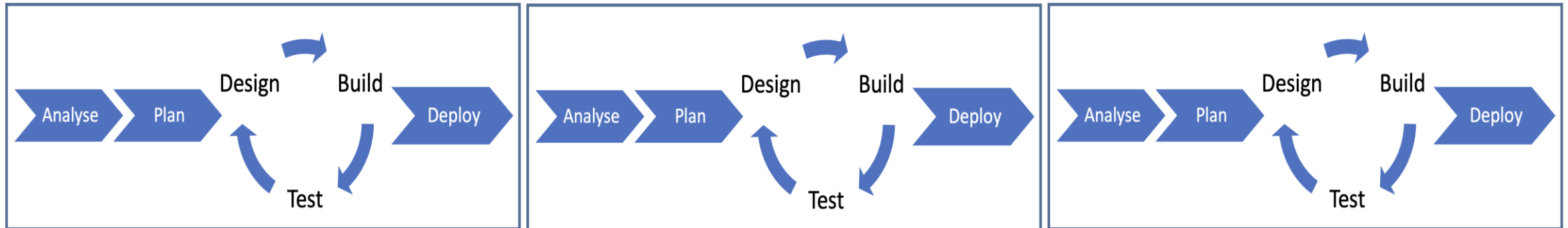




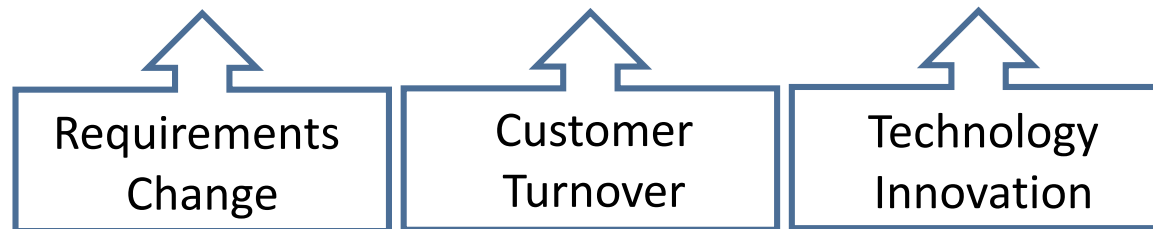
# Waterfall



# Agile



## Project Timeline



## Assignment task

Your task is to form a group (recommended minimum of two members, maximum of four) to:

- Collaborate and develop a game in a language and framework of your choice.
- Document the development stages of your game by making regular commits to your combined GitHub repository.
- Present (as a group) your game concept and a live demonstration of how your game can be played.

## Guidelines and advice

### Basic requirements for the development of Game:

- Select a suitable Object-Oriented Programming language (e.g. Python, Java, C#, C++, JS etc) and framework (PyGame, Greenfoot, MonoGame, Unity etc).
- Create a 2D game with a minimum of one level.
- The playable character should have some form of attribute (health, energy, magic, strength etc) level which can increase or decrease.
- The playable character should be able to collect and store items and then use items.
- There should be some form of 'opposition' (e.g. enemies, time limit, reduction in energy as player moves around the level etc.) that makes the objective of the game more challenging to achieve.



The image features the GitHub Octocat logo, a stylized grey octopus-like creature with large eyes and a single visible tail, centered within a dark purple circle. This circle is surrounded by a light grey ring, all set against a black background. The word "GitHub" is written in white, sans-serif font across the middle of the Octocat's body.

GitHub

A man with dark hair and glasses, wearing a blue jacket over a red shirt, is sitting at a desk. His hands are clasped in front of him, and he has a thoughtful expression. The background is a blurred office or home workspace with warm lighting. A yellow spherical object is visible on the desk in the foreground.

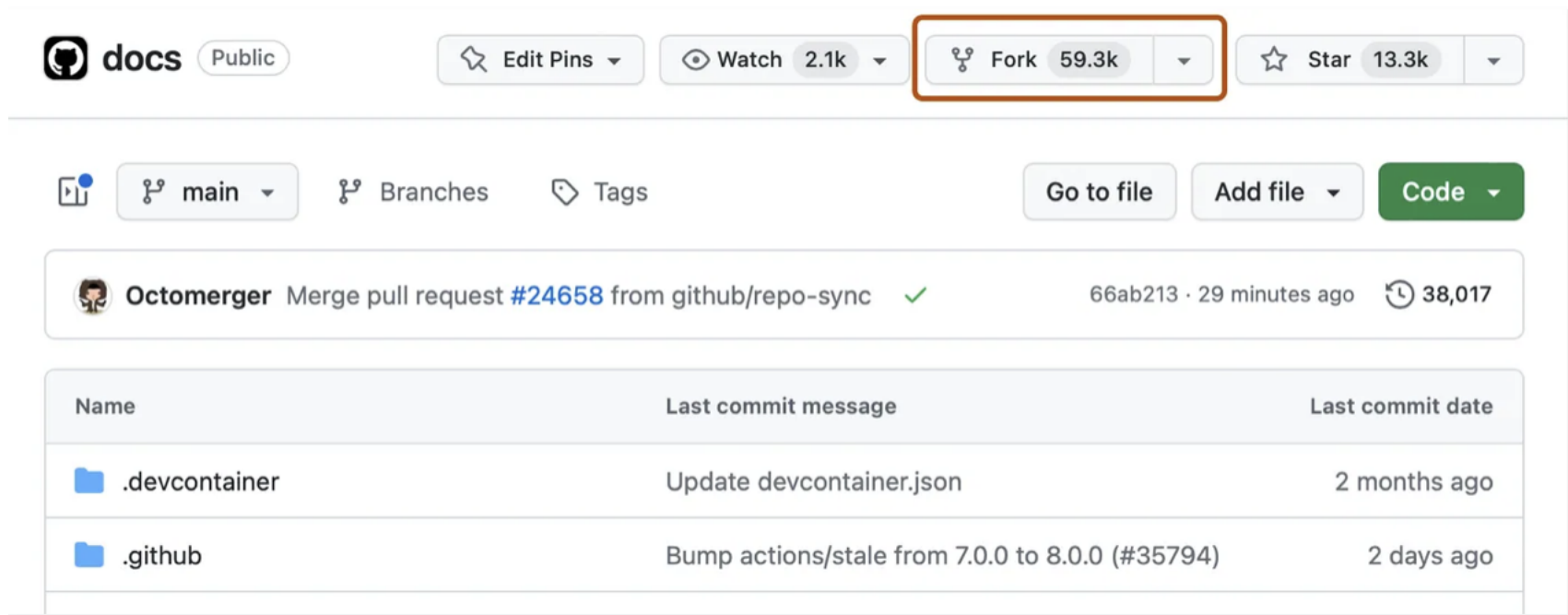
**What is  
GitHub?**





Forking or  
branching?



# Forking: Creates a copy of the entire repository

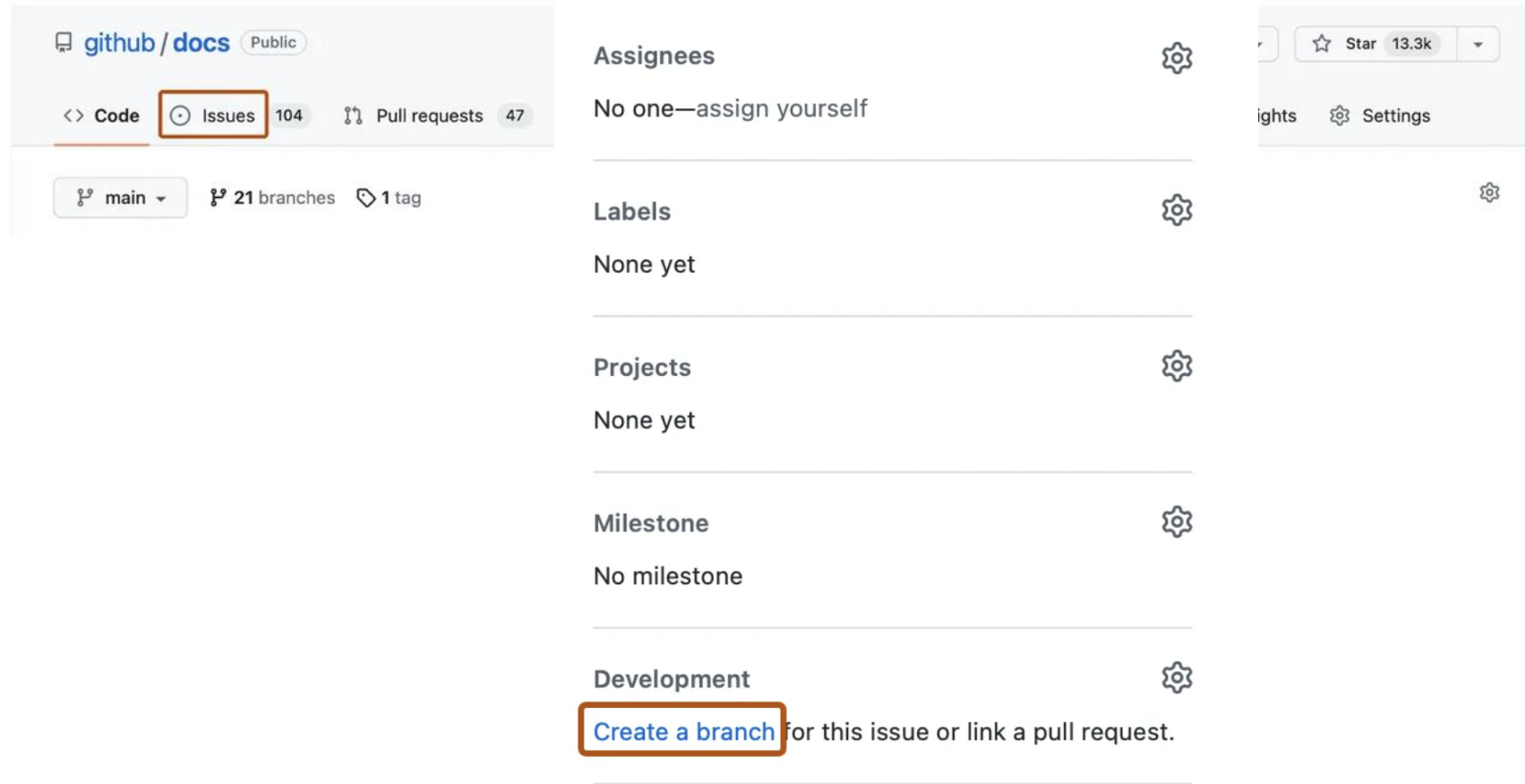


The screenshot shows the GitHub interface for a repository named 'docs'. The 'Fork' button, which has a fork icon and the text 'Fork 59.3k', is highlighted with a red rectangular box. Other interface elements include the repository name 'docs' with a 'Public' badge, 'Edit Pins', 'Watch 2.1k', 'Star 13.3k', and a 'main' branch selector. Below the repository header, there is a notification for a merged pull request and a table of files.

Name	Last commit message	Last commit date
 .devcontainer	Update devcontainer.json	2 months ago
 .github	Bump actions/stale from 7.0.0 to 8.0.0 (#35794)	2 days ago

<https://docs.github.com/en/get-started/quickstart/fork-a-repo>

# Branching: portion of the repo (branch of a tree) to isolate work on an issue



<https://docs.github.com/en/issues/tracking-your-work-with-issues/creating-a-branch-for-an-issue>