Professional Issues in IT

Professional Software Development

A Program

- A program
  - Complete in itself
  - Ready to run
    - By the author
    - For the planned inputs
    - On the system on which it was developed
- Used for estimating productivity by individual programmer???

A Programming Product

- A program that can be
  - Run
  - Tested
  - Repaired
  - Extended
  - By anybody!
  - Many operating systems
  - Many sets of data
  - Require thorough testing
  - Need thorough documentation

A Programming System

- A collection of interacting programs
  - Precisely defined interfaces
  - Uses only prescribed budget of resources
    - Memory, I/O, Processor Time
  - Must be tested in all expected combinations with other system components

Programming System Product

- It is the truly useful object
- The intended product of most system programming efforts

Evolution of the programming system product

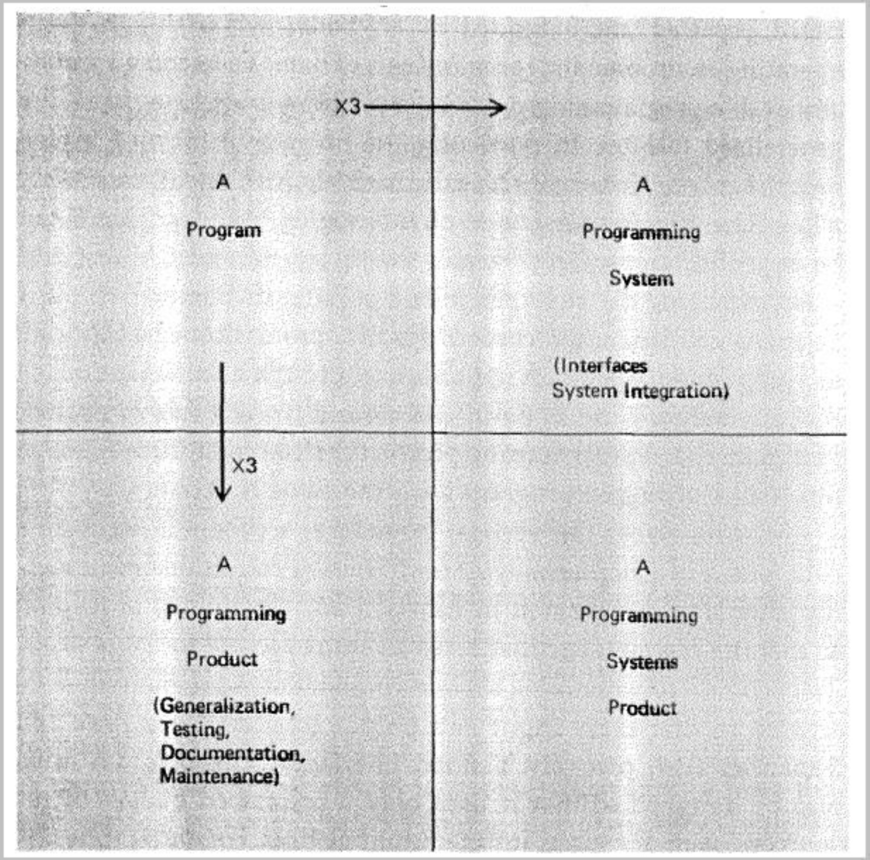
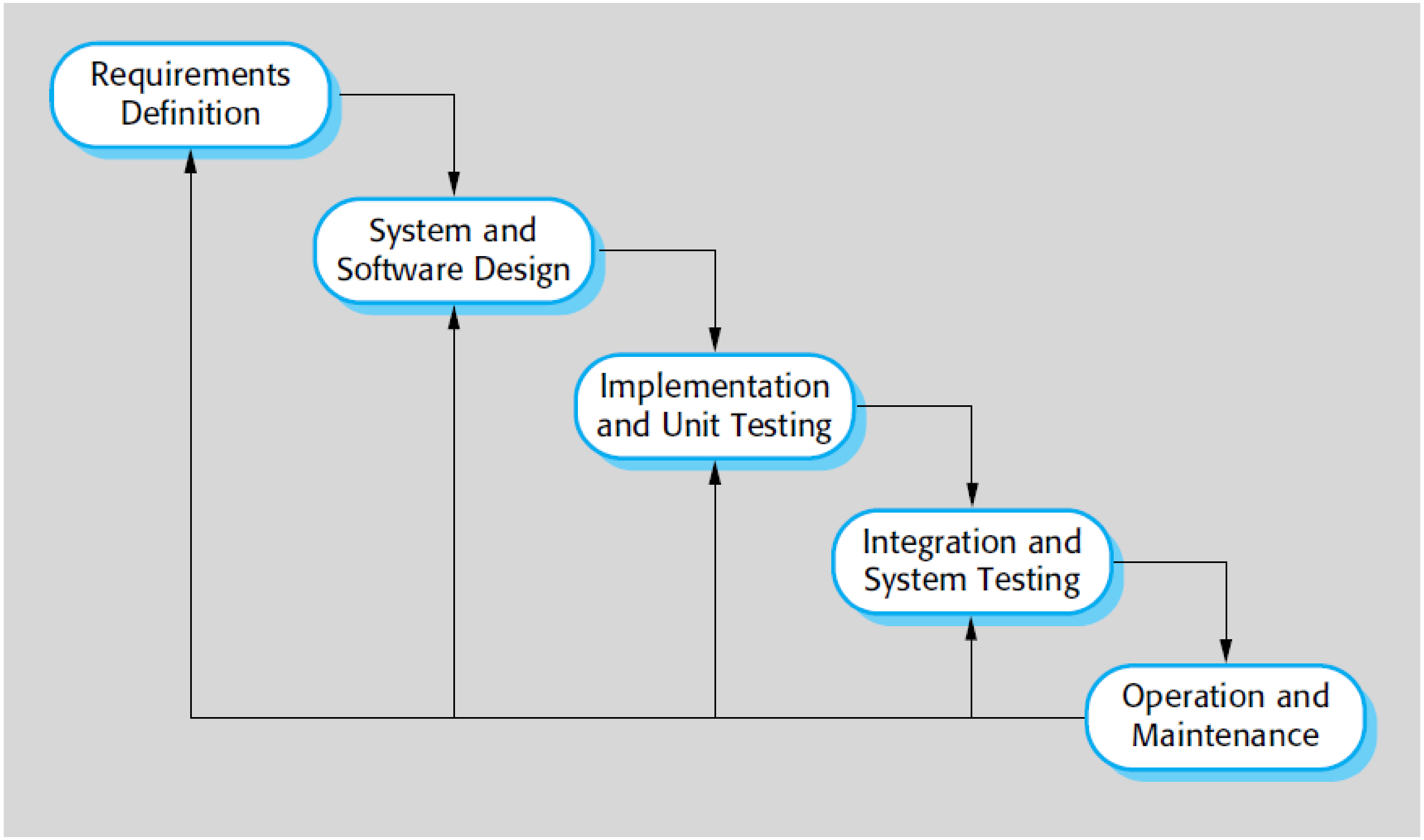


Fig. 1.1 Evolution of the programming systems product

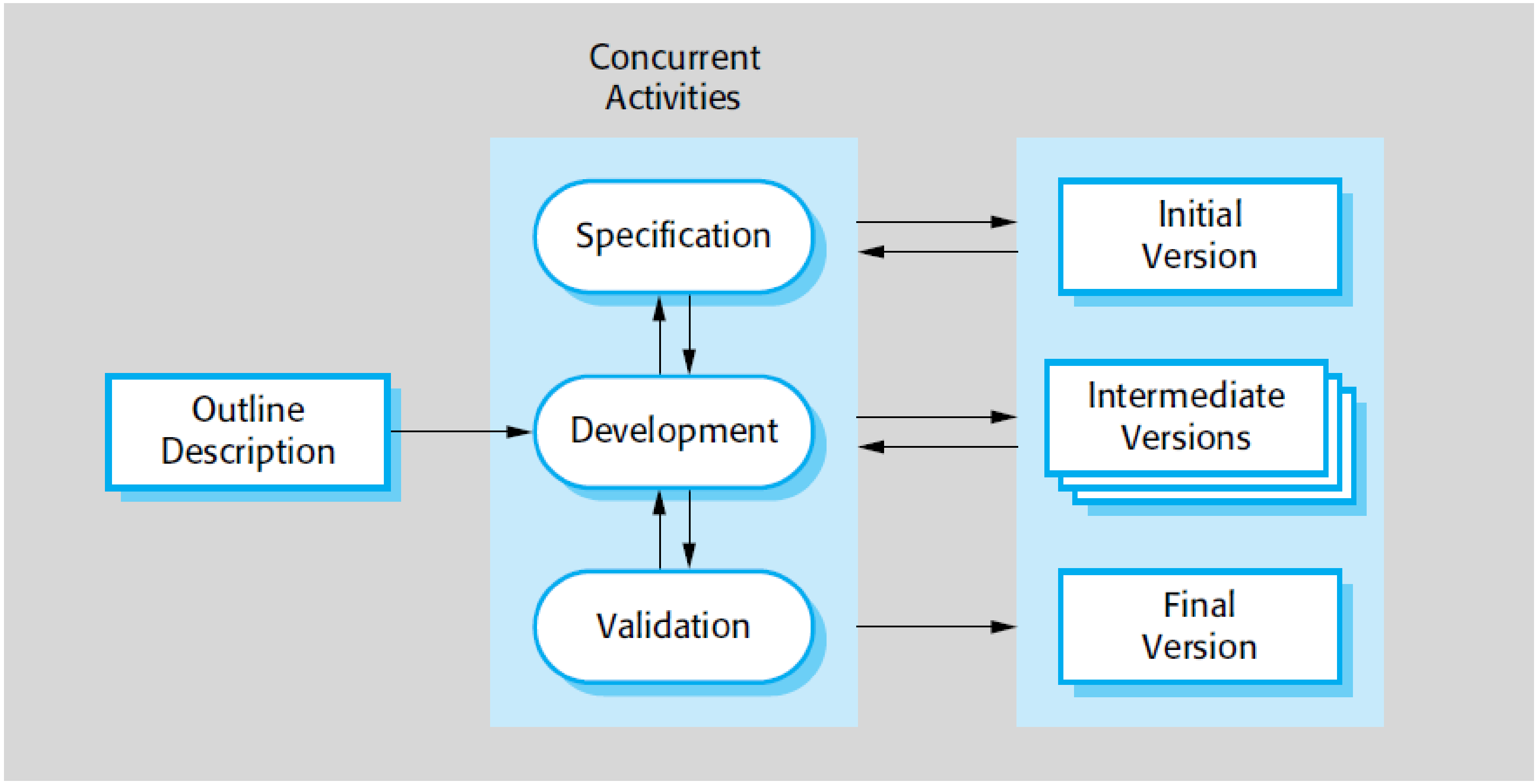
Generic Process Models

1. The waterfall model
2. Incremental development
3. Integration and configuration

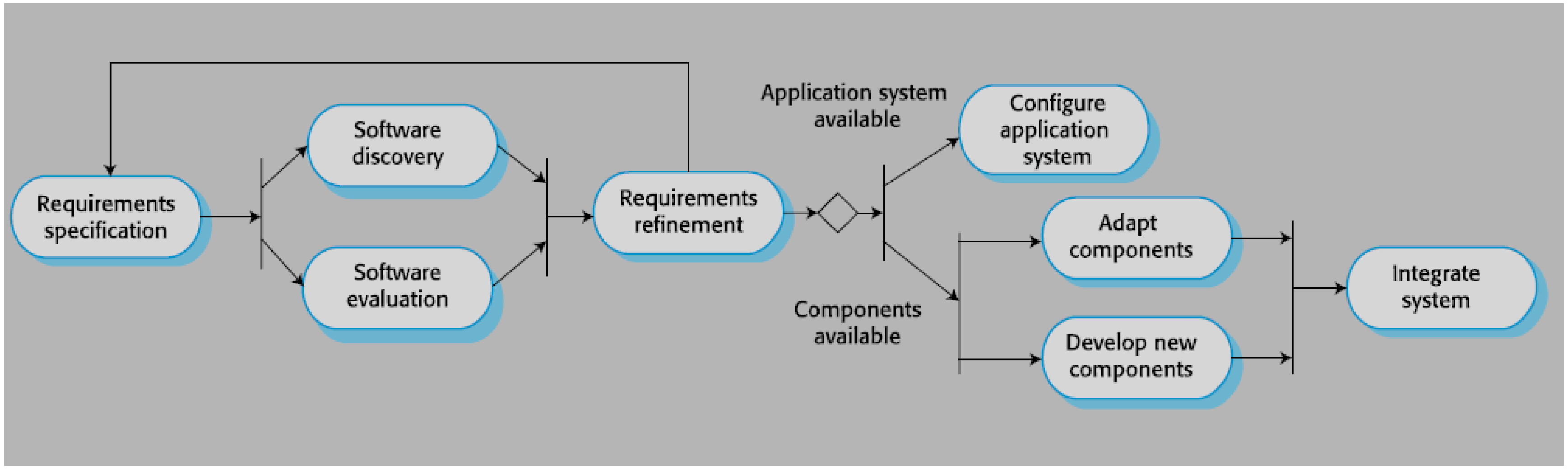
The waterfall model


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

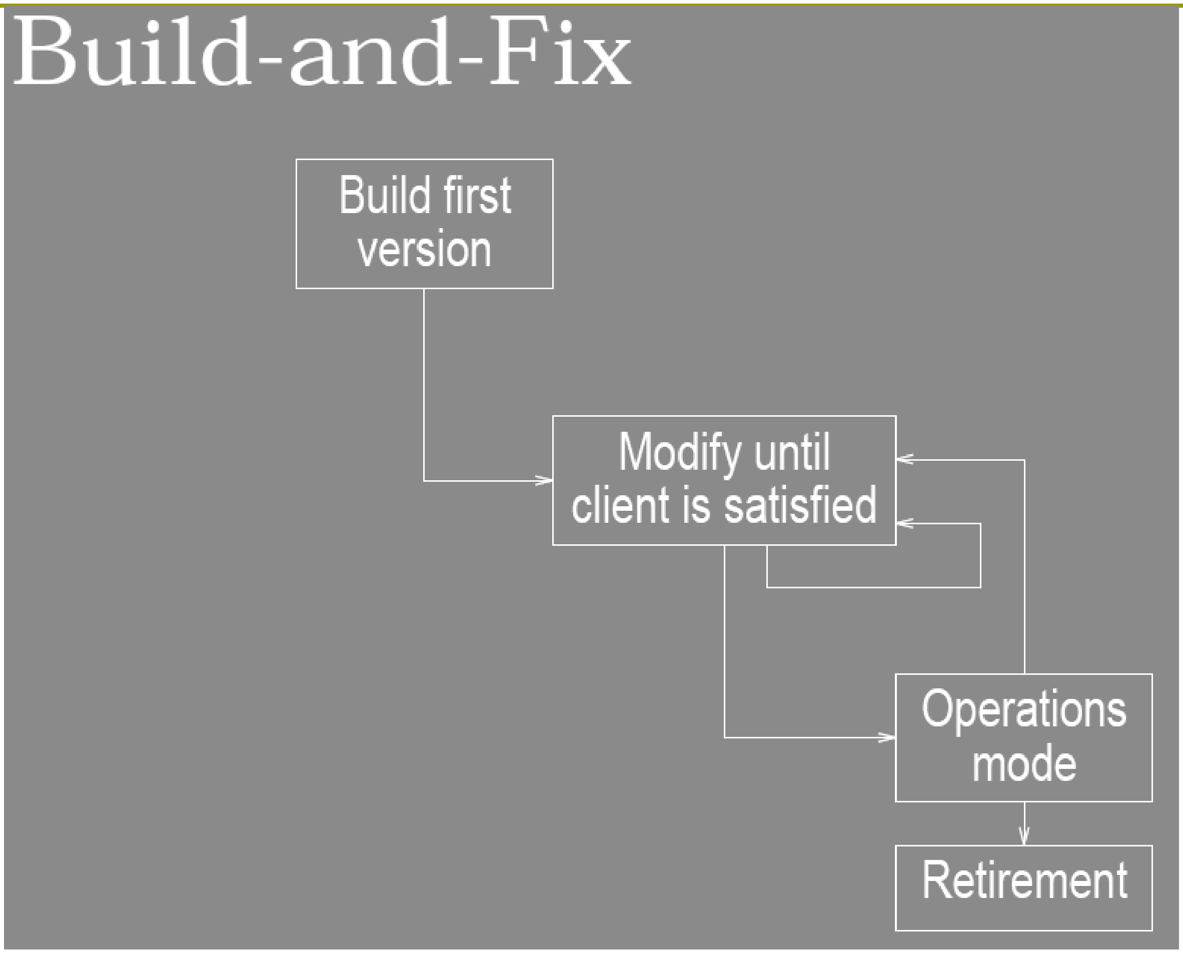
Incremental development



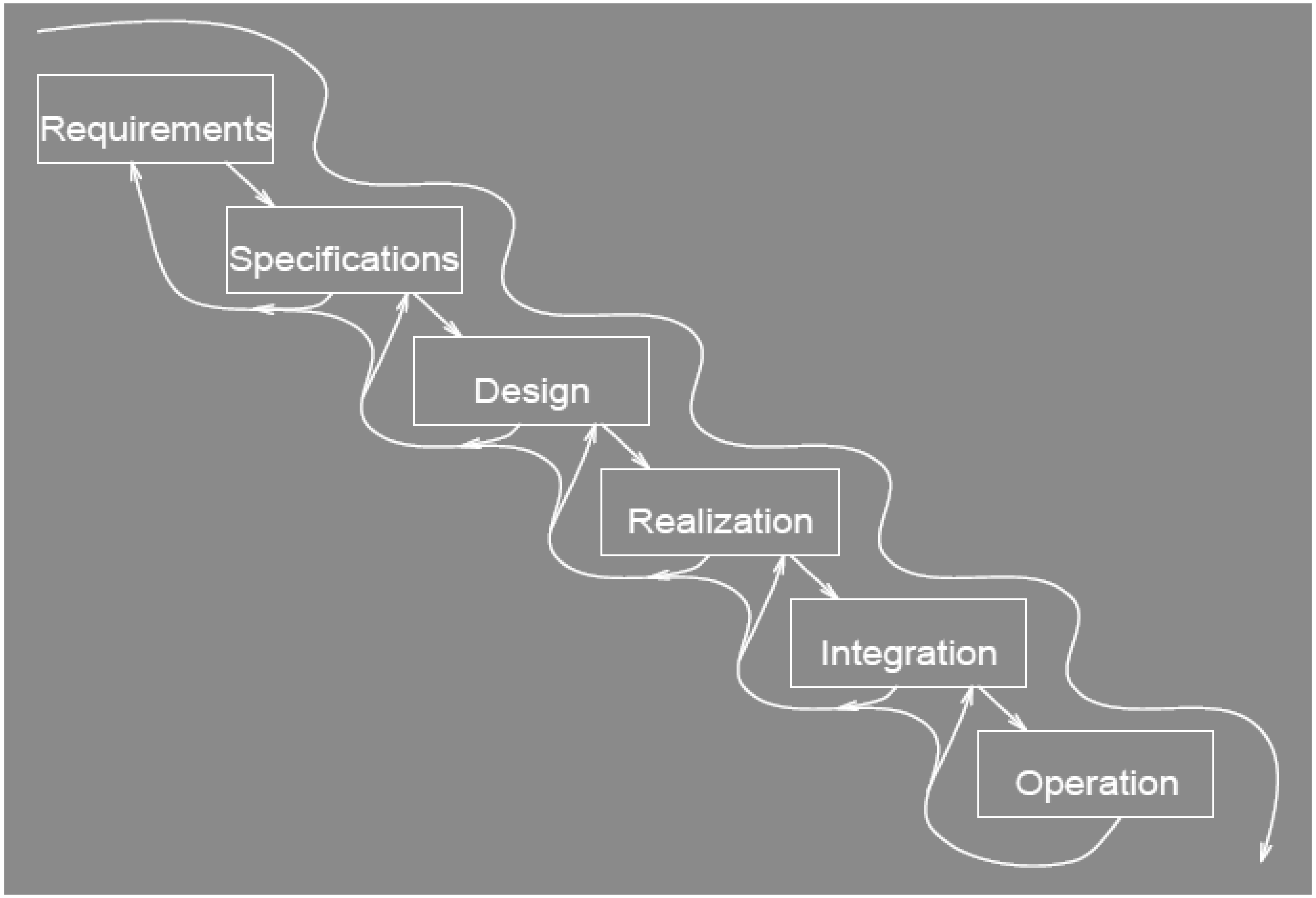
Integration and configuration



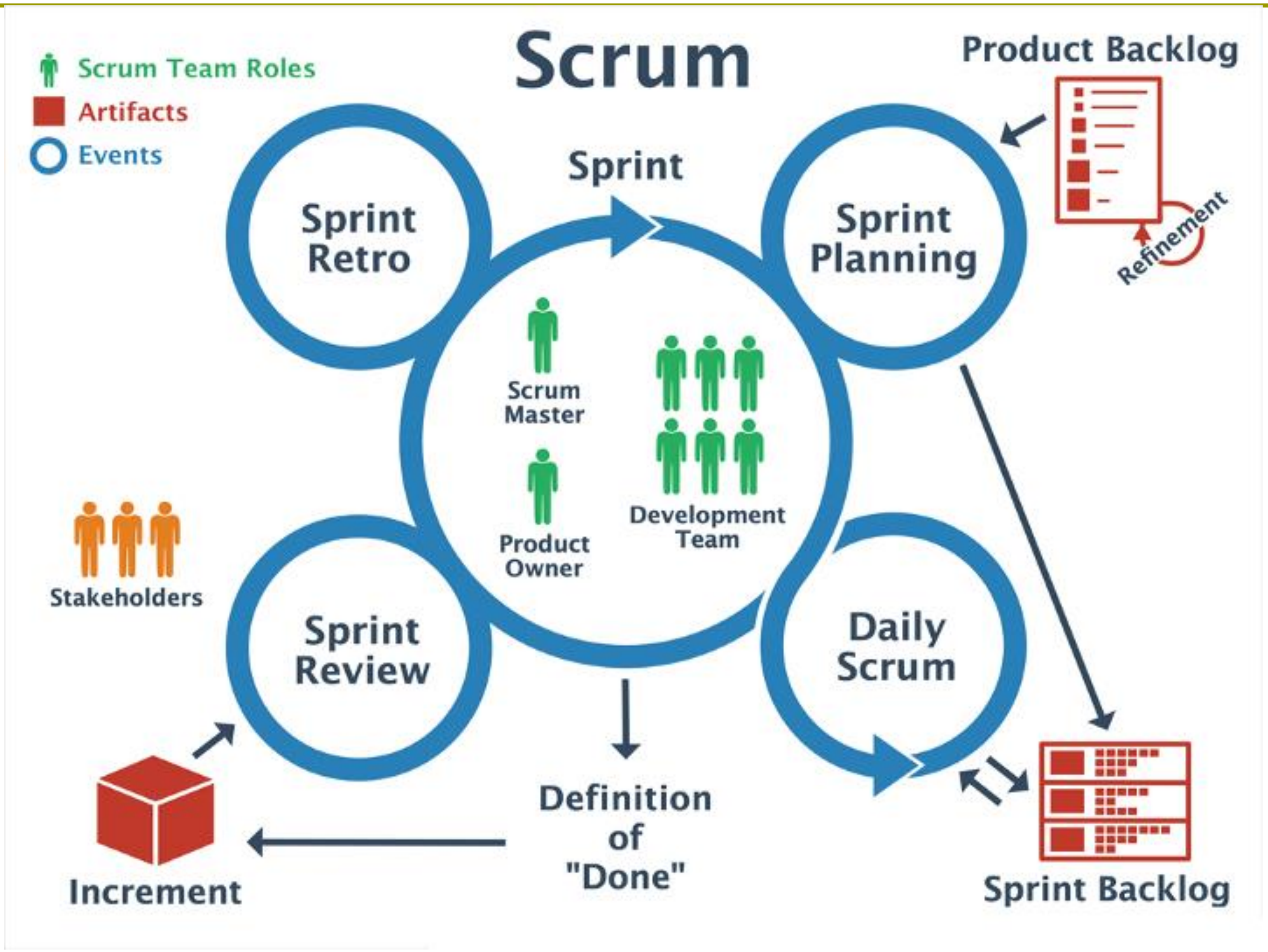
Common SPM models: Build-N-Fix



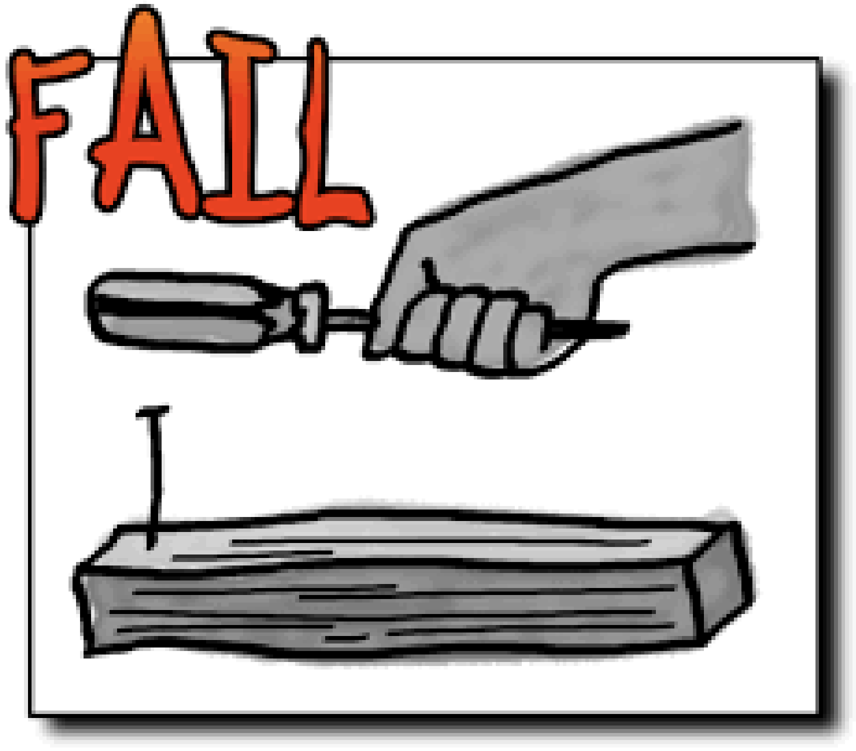
Common SPM models: Waterfall cycle



Modern software models: Agile or scrum



A software developers toolkit: Which ones do you have?



A software developers toolkit: Which ones do you have?

- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **“I wrote code 3 years ago, no documentation, have to go back and understand it…”**

A software developers toolkit: Which ones do you have?

- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **“I just compiled and ran code, now it is not working”**

A software developers toolkit: Which ones do you have?

- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **“I write code at home and on laptop, which version is correct one”**

A software developers toolkit: Which ones do you have?

- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **"In my code file, I need to find all numbers between 34-38 and replace them with 70-75"**
  - **"Need to delete all comments"**
  - **"Need to find all variables with no numerics"**

A software developers toolkit: Which ones do you have?

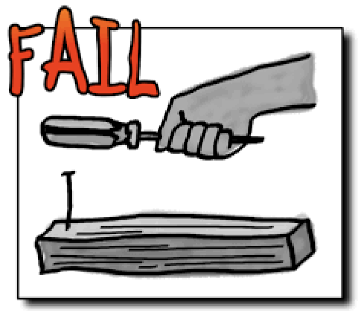
- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **“I wrote a piece of software, how do I test it is working?”**

A software developers toolkit: Which ones do you have?

- **Beyond writing code: Which of these common software writing situations can you handle?**
  - **“I have files**
    - **Main.c**
    - **Main1.c**
    - **Main2.c**
    - **Main25Dec.c**
    - **MainFinal.c**
    - **MainFinale.c**
    - **MainFinale1Final.c**

A software developers toolkit: Which ones do you have?

- IDE and debugger
- Compare files and find differences
- Work with regular expressions
- Versioning and sharing
- Software testing
- Continuous Itegration



Visual Studio

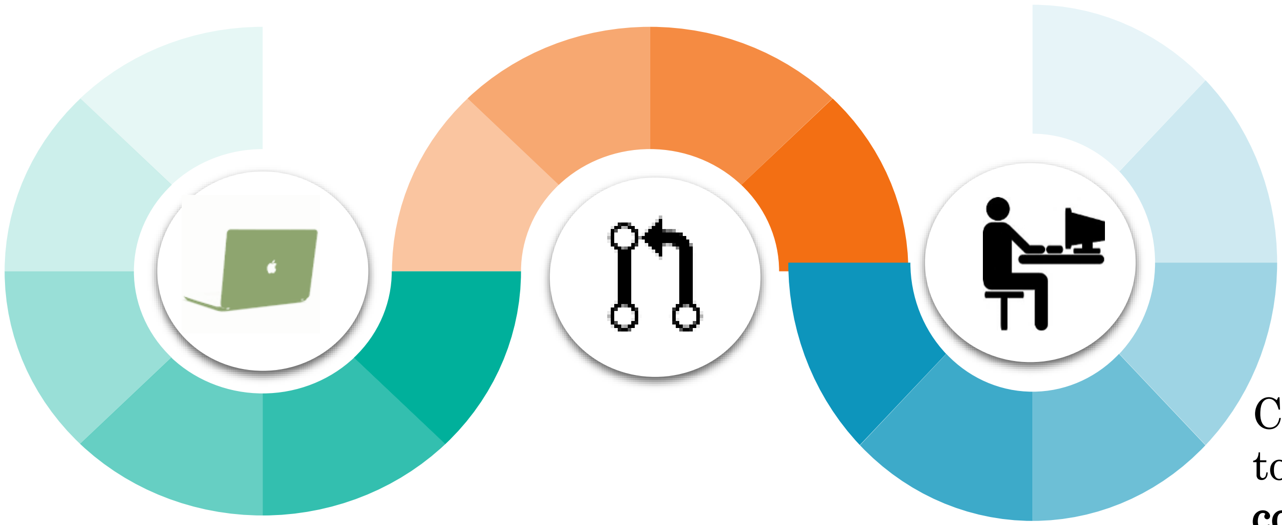
Jenkins

DIFFMERGE

Expresso

Using source control-steps

Connect local code
directory with Git repo
Commit code



Create **Github account**
Create new rep on Github
Install **TortoiseSVN**

Checkout from Git
to **make working
copy**
Work on it
Commit and add
comments

"*Code complete*, A practical handbook of software construction" Steve McConnell

Some tips to write professional code

- 1: Don't write code
  - I/O, sort, search, webIO, images…
  - Instead, reuse…
- 2: Don't do hard things, do easy things
  - Avoid tricky algorithms that you don't understand
  - Create a new module for a new task

"*Code complete*, A practical handbook of software construction" Steve McConnell

Complexity of Software construction

- Essential
  - the fashioning of the complex conceptual structures that compose the abstract software entity
- Accidental
  - the representation of these abstract entities in programming languages and the mapping of these onto machine languages within space and speed constraints

Complexity of Software construction

- How the accidental part is addressed
  - High level languages
  - IDEs
  - Toolkits and Frameworks
  - Design languages (UML etc.)

Complexity of Software construction

- How the essential part is to be addressed
  - Buy vs build
  - Rapid Prototyping
  - Growing software organically
  - Training of conceptual designers

Working in industry is different from college assignments

- Gather and analyze requirements when they aren't directly given to you
- Design and analyze architecture with near endless possibilities
- Create test plans and act on them to evaluate and improve the quality of a system
- Work collaboratively on a team of people with different backgrounds and experience levels

Working in industry is different from college assignments

- Estimate and plan work *even if you don't know exactly what to build*
- Communicate effectively with stakeholders *who have different needs that don't necessarily align*
- Negotiate schedule, budget, quality, and features *without disappointing stakeholders*

Reference

- Frederick P. Brooks, Jr. – The Mythical Man-Month, 2nd Edition – Chapter 1 & 16
- https://softwareengineering.stackexchange.com/questions/119470/differences-between-programming-in-school-vs-programming-in-industry