

2.5a Software Engineering

10x10 – Introduction to Biomedical Informatics & Artificial Intelligence
William Hersh, MD, FACMI, FAMIA, FIAHSI
Copyright 2026, All Rights Reserved



Software engineering

“The application of a systematic, disciplined, quantifiable approach to the development, maintenance, and operation of software; that is, the application of engineering to software.”

- [Software Engineering Body of Knowledge \(SWEBOK\)](#)

A science itself as we learn the best practices for developing complex software systems (Sommerville, 2018)

SWEBOK major areas of software engineering

Software requirements

- Requirements analysis, specification, validation

Software design

- Structure and architecture

Software construction

- Reducing complexity, anticipating diversity

Software testing

Software maintenance

Software quality

- Finding defects

Software engineering topics in this unit

System design

Software project management

System development

System life-cycle

When things go wrong

Human-computer interaction

Use case analysis

Important issues for system design

Quality and style of interface

- Must be clear and consistent

Convenience and accessibility

- Devices where needed and in adequate numbers

Speed and response time

- Essential for busy clinicians

Reliability

- Crucial as we become more dependent on them

Security

- Essential as amount of data and connectivity grows

Integration

- Interoperability of systems and data essential

Software project management (Chemuturi, 2010)

Software requirements

Discussion summary

Use cases

Functional and non-functional requirements

Software requirements

Software requirements completely describes the behavior required of the software before it is designed, built, and tested

Requirements analysts (sometimes called business analysts) develop software requirements specifications through requirements elicitation that includes

- Interviews with users, stakeholders, and others whose perspective needs to be taken into account during the software design, development, and testing
- Observation of users at work
- Distribution of discussion summaries to verify the data gathered in interviews

Discussion summary

A requirements analyst uses a discussion summary to summarize information gathered during elicitation and validate it through a review

Notes gathered during the elicitation should fit into the discussion summary template

- Discussion Summary outline
- Project background
 - Purpose of project
 - Scope of project
 - Other background information
- Perspectives
 - Who will use the system?
 - Who can provide input about the system?
- Project Objectives
 - Known business rules
 - System information and/or diagrams
 - Assumptions and dependencies
 - Design and implementation constraints
- Risks
- Known future enhancements
- References
- Open, unresolved or TBD issues

Use cases

A use case is a description of a specific interaction that a user may have with the system

- Use cases are tools for describing the functionality of the software

Use cases do not describe any internal workings of the software or explain how it will be implemented

- They show how the steps that the user follows to use the software to carry out his or her work

All of the ways that the users interact with the software can be described through use cases

Functional and non-functional requirements

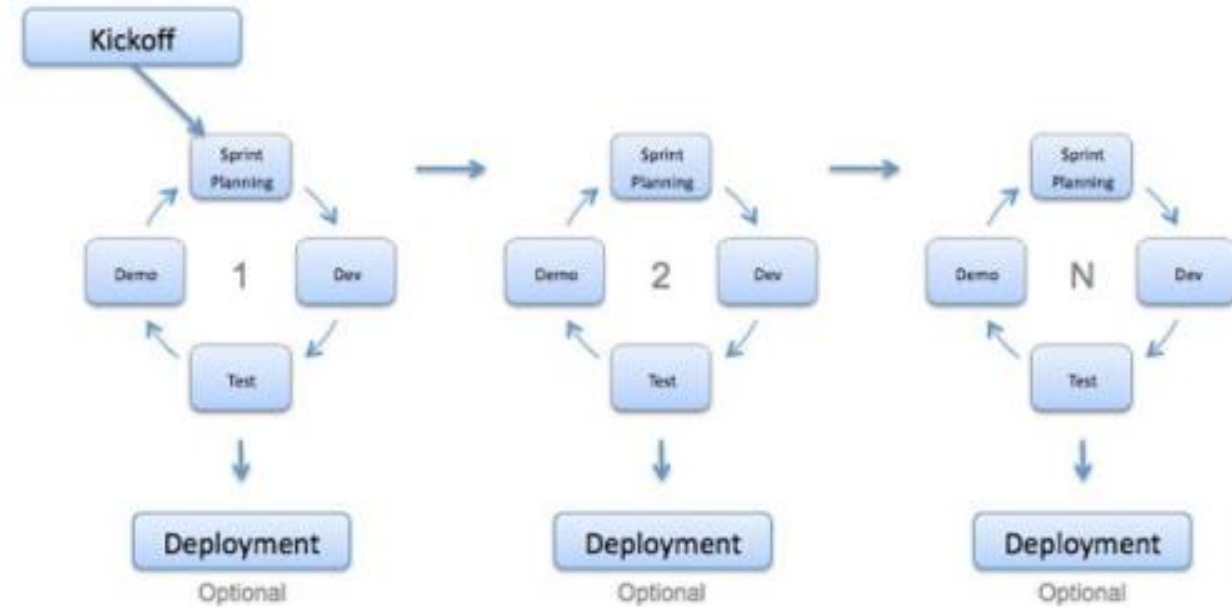
Functional requirements define the outward behavior required of the software project

- Goal of requirement is to communicate the needed behavior in a clear and unambiguous manner
- Behavior in the requirement can contain lists, bullets, equations, pictures, references to external documents, or any other material that will help the reader understand what needs to be implemented

Non-functional requirements define characteristics of the software which do not change its behavior, such as

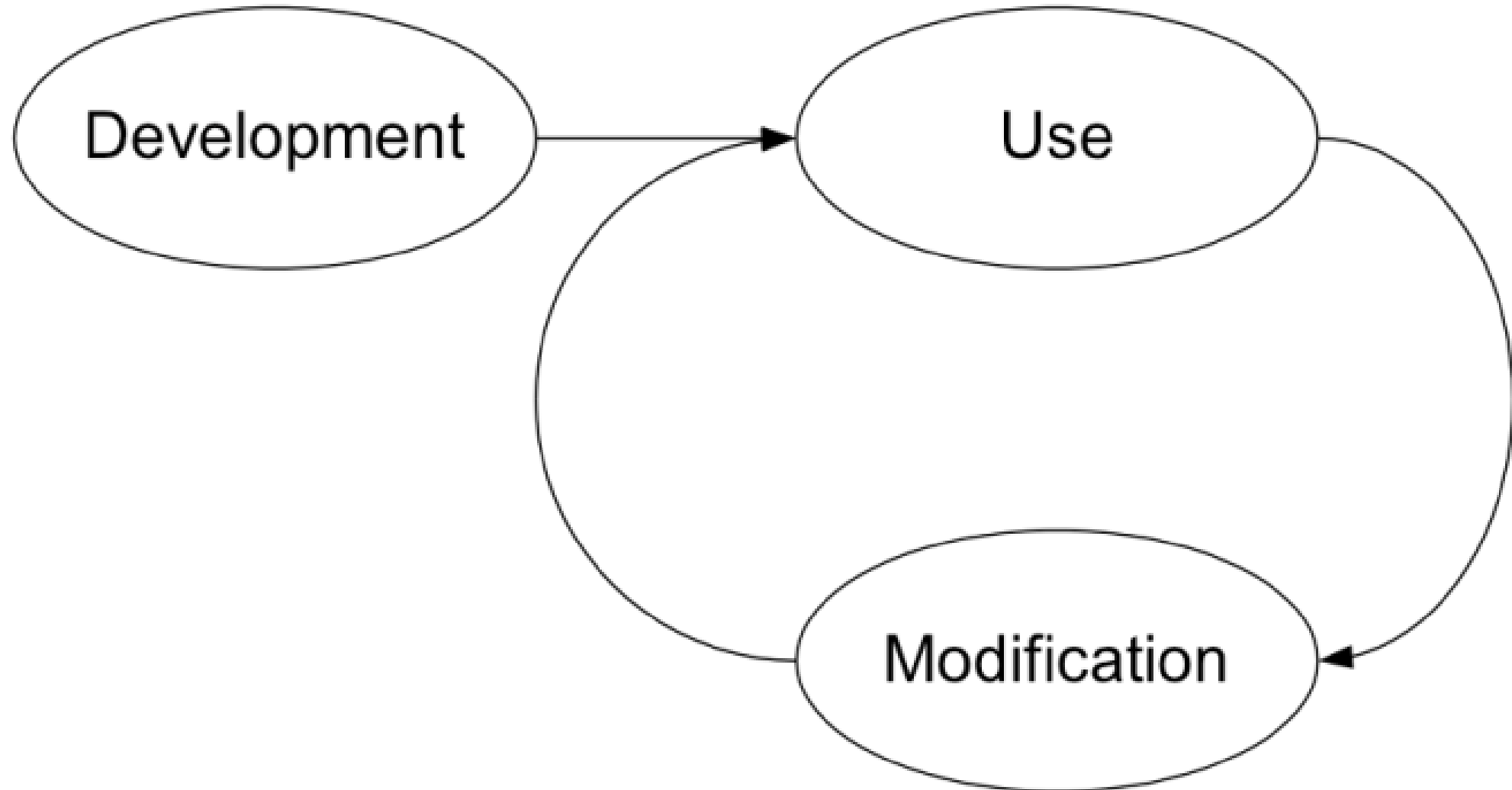
- How easy the software is to use
- How quickly it executes
- How reliable it is
- How well it behaves when unexpected conditions arise

Agile



2.5a - Hersh - AMIA

Software life cycle



Most software used in and out of healthcare today is commercial

Users and organizations have little control over code

Health IT projects are really not “IT” projects (Murphy, 2009) – “We are not in the business of IT; we are in the business of healthcare.”

Growing concern over “hold harmless” clauses in EHR software licensing agreements

- Need to balance with safety (Koppel, 2009)
- AMIA call for patient safety to trump all other concerns (Goodman, 2011)

EHR vendor response has been development of [Code of Conduct](#)

- Business practices make commitments around patient safety, interoperability, billing, documentation, privacy and security, and patient engagement



When things go wrong

Computer hardware is remarkably reliable and is becoming more so

However, software systems are very complex and it is difficult to anticipate all possible error conditions (Randell, 2000)

- Most dangerous programming errors (CWE/SANS, 2020)

Concern about safety of health IT software led to IOM report and debate for increased federal regulation (IOM, 2012)

Launching of Healthcare.gov demonstrated provided many software engineering lessons (Shirky, 2013; Tabbaa, 2018); different (fortunately) from many clinical implementations (Blumenthal, 2013)

Growing complexity of software and systems, and our dependency on them, leads some to worry about coming “software apocalypse” and our need for better software engineering (Somers, 2017)



Also concerns over software used in biomedical research

Many scientific researchers write code but are not always well-versed in best practices of testing and error detection (Merali, 2010; Sainani, 2011)

Scientists have history of relying on problematic software, e.g.,

- Commercial software may produce invalid results (Schütz, 2014), casting doubt on biology, e.g., drug bioequivalence (Hirschler, 2014)
- Default settings in fMRI analysis software (Eklund, 2016; Eklund, 2019)
- Bug in Python library giving incorrect results in computational chemistry analyses (Neupane, 2019)
- Genes needing to be renamed because names modified by Microsoft Excel, usually into dates (Vincent, 2020), e.g., Membrane Associated Ring-CH-Type Finger 1 with abbreviation MARCH1 converted into date, 1-Mar
 - Persisted in literature despite known problem (Abeysooriya, 2021)
 - Excel can now be configured to not automatically convert dates (Davis, 2023)
- Of R programs published for replicability, 74% failed with errors and 56% still failed after code cleaning (Trisovic, 2022)
- Important program for particle physics maintained by a single retiree (von Hippel, 2022)

Concerns over software in biomedical research (cont.)

May not select best software packages for their work (Joppa, 2013)

These and other problems have led to best practice recommendations

- Publishing of source code, workflows, and computational environments for research (Stodden, 2016)
- Good-enough practices for scientific computing (Wilson, 2017)
- Guidelines for researchers developing Web applications (Burnett, 2021; Saia, 2022)