Open Project Management

from an "open" perspective

UNIT 5

Instructor: Dr. Bradly Alicea

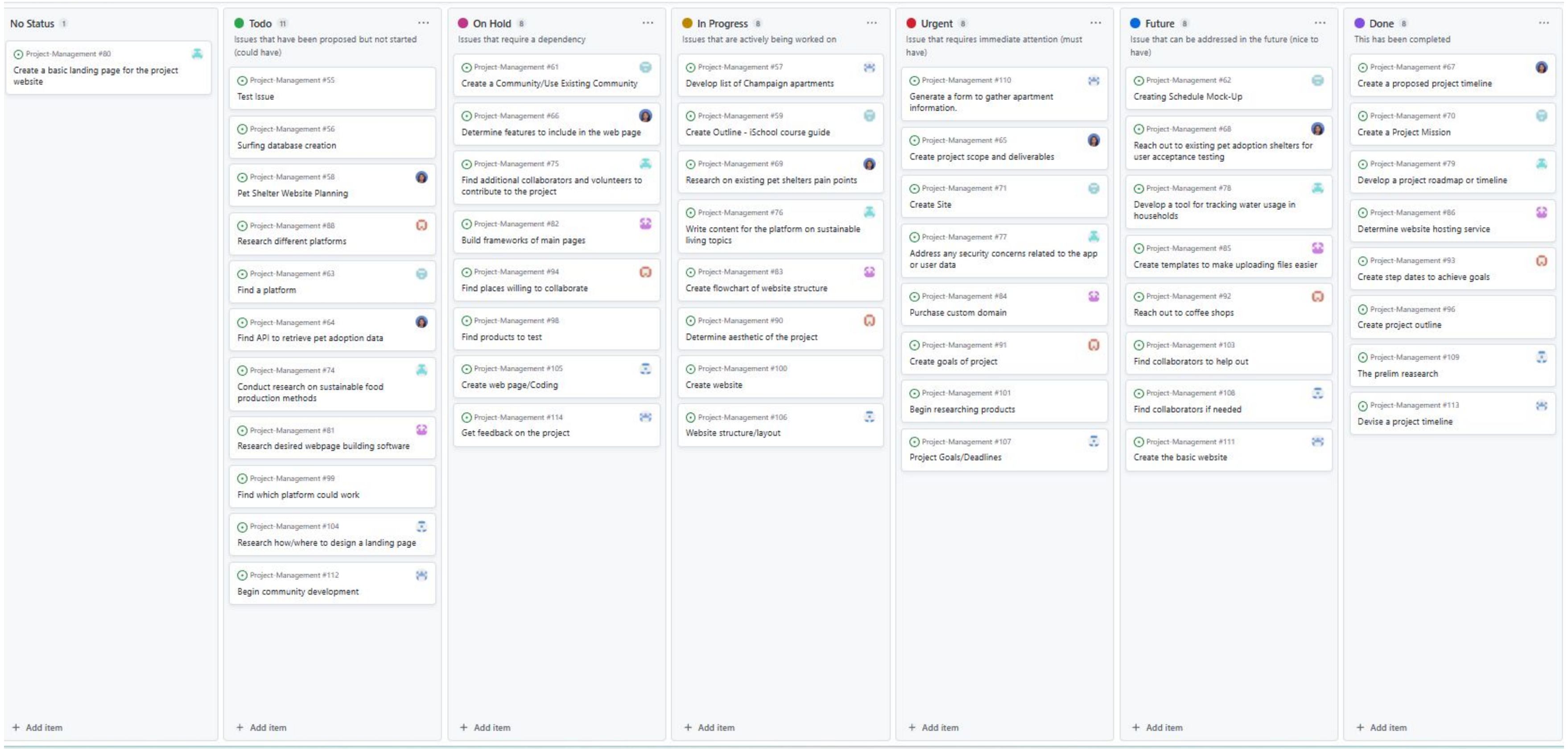
http://bradly-alicea.weebly.com

Lecture 19

Open Project Management

Welcome Back!

Progress on Quiz 2



No Status 1

- Project-Management #80 — Create a basic landing page for the project website
- + Add item

Todo 11 — Issues that have been proposed but not started (could have)

- Project-Management #55 — Test Issue
- Project-Management #56 — Surfing database creation
- Project-Management #58 — Pet Shelter Website Planning
- Project-Management #88 — Research different platforms
- Project-Management #63 — Find a platform
- Project-Management #64 — Find API to retrieve pet adoption data
- Project-Management #74 — Conduct research on sustainable food production methods
- Project-Management #81 — Research desired webpage building software
- Project-Management #99 — Find which platform could work
- Project-Management #104 — Research how/where to design a landing page
- Project-Management #112 — Begin community development
- + Add item

On Hold 8 — Issues that require a dependency

- Project-Management #61 — Create a Community/Use Existing Community
- Project-Management #66 — Determine features to include in the web page
- Project-Management #75 — Find additional collaborators and volunteers to contribute to the project
- Project-Management #82 — Build frameworks of main pages
- Project-Management #94 — Find places willing to collaborate
- Project-Management #98 — Find products to test
- Project-Management #105 — Create web page/Coding
- Project-Management #114 — Get feedback on the project
- + Add item

In Progress 8 — Issues that are actively being worked on

- Project-Management #57 — Develop list of Champaign apartments
- Project-Management #59 — Create Outline - iSchool course guide
- Project-Management #69 — Research on existing pet shelters pain points
- Project-Management #76 — Write content for the platform on sustainable living topics
- Project-Management #83 — Create flowchart of website structure
- Project-Management #90 — Determine aesthetic of the project
- Project-Management #100 — Create website
- Project-Management #106 — Website structure/layout
- + Add item

Urgent 8 — Issue that requires immediate attention (must have)

- Project-Management #110 — Generate a form to gather apartment information.
- Project-Management #65 — Create project scope and deliverables
- Project-Management #71 — Create Site
- Project-Management #77 — Address any security concerns related to the app or user data
- Project-Management #84 — Purchase custom domain
- Project-Management #91 — Create goals of project
- Project-Management #101 — Begin researching products
- Project-Management #107 — Project Goals/Deadlines
- + Add item

Future 8 — Issue that can be addressed in the future (nice to have)

- Project-Management #62 — Creating Schedule Mock-Up
- Project-Management #68 — Reach out to existing pet adoption shelters for user acceptance testing
- Project-Management #78 — Develop a tool for tracking water usage in households
- Project-Management #85 — Create templates to make uploading files easier
- Project-Management #92 — Reach out to coffee shops
- Project-Management #103 — Find collaborators to help out
- Project-Management #108 — Find collaborators if needed
- Project-Management #111 — Create the basic website
- + Add item

Done 8 — This has been completed

- Project-Management #67 — Create a proposed project timeline
- Project-Management #70 — Create a Project Mission
- Project-Management #79 — Develop a project roadmap or timeline
- Project-Management #86 — Determine website hosting service
- Project-Management #93 — Create step dates to achieve goals
- Project-Management #96 — Create project outline
- Project-Management #109 — The prelim reasearch
- Project-Management #113 — Devise a project timeline
- + Add item

Wikipedia Contribution Model

discussed in “Reinventing Discovery” (Michael Nielsen)

Dynamic Division of Labor (DDL).

* one person does *x* amount of work, another picks up and does *y* amount of work

Flexible micro-contributions lower the barrier to entry:

* change a single line of code, or make edits to a Wikipedia page.

Wikipedia Contribution Model

discussed in “Reinventing Discovery” (Michael Nielsen)

Dynamic Division of Labor (DDL).

* one person does *x* amount of work, another picks up and does *y* amount of work

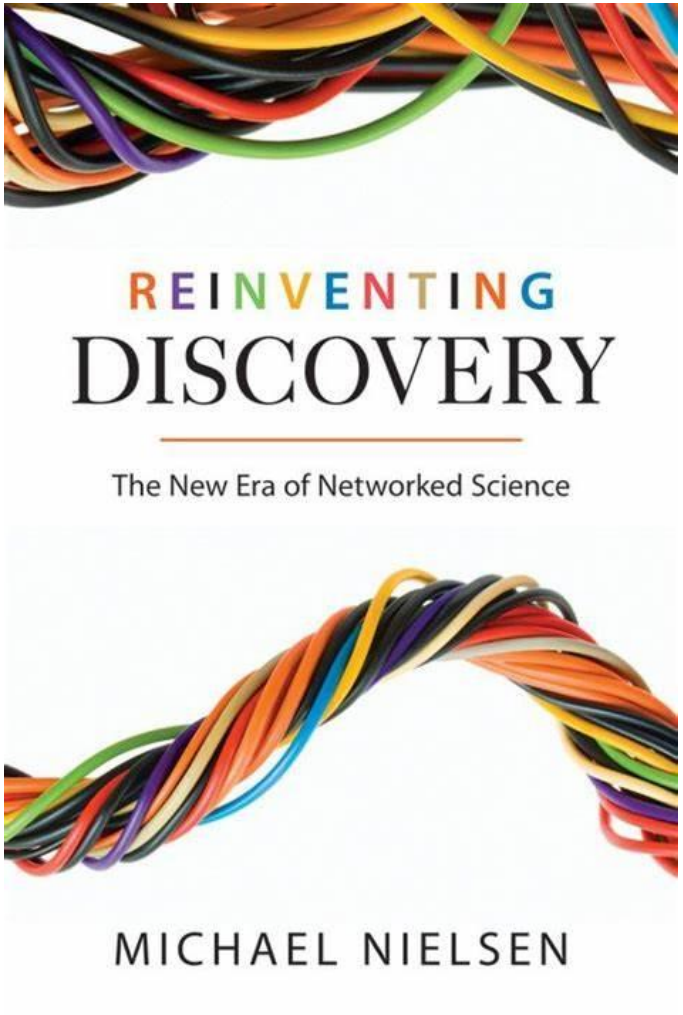
Flexible micro-contributions lower the barrier to entry:

* change a single line of code, or make edits to a Wikipedia page.

Project transferability raises the infrastructural costs:

* annotated code, documentation, contributions as easily-defined tasks.

Which combination of strategies increases range of ideas contributed at minimal organizational cost?


REINVENTING DISCOVERY

The New Era of Networked Science

MICHAEL NIELSEN

Focus on open science: openness and transparency can accelerate your timeline.

Two enabling phenomena

- amplifying collective intelligence.
- networked science.

Modularizing and decentralizing projects tend to make them more accessible.

Release Life Cycle

From Chapter 8, “Program Management for Open Source Projects”
(Ben Cotton).

Not *how long*, but *how many*?

- semantic versioning (*x.x.x*), only support the latest few releases.
- release only once or twice in the entire project life cycle.

Release Life Cycle

From Chapter 8, “Program Management for Open Source Projects” (Ben Cotton).

Not *how long*, but *how many*?

- semantic versioning (*x.x.x*), only support the latest few releases.
- release only once or twice in the entire project life cycle.

Life Cycle: set of phases (alpha/beta, alpha/beta).

- Alpha: developmental releases that have passed Continuous Integration (CI).
- Beta: get feedback from potential users.

Support Cycle

What features do you include, make functional, and maintain?

- the more features you have the more support you need.
- the longer your life cycle, the more support you need.
- release model: calendar, feature, and whim-based.

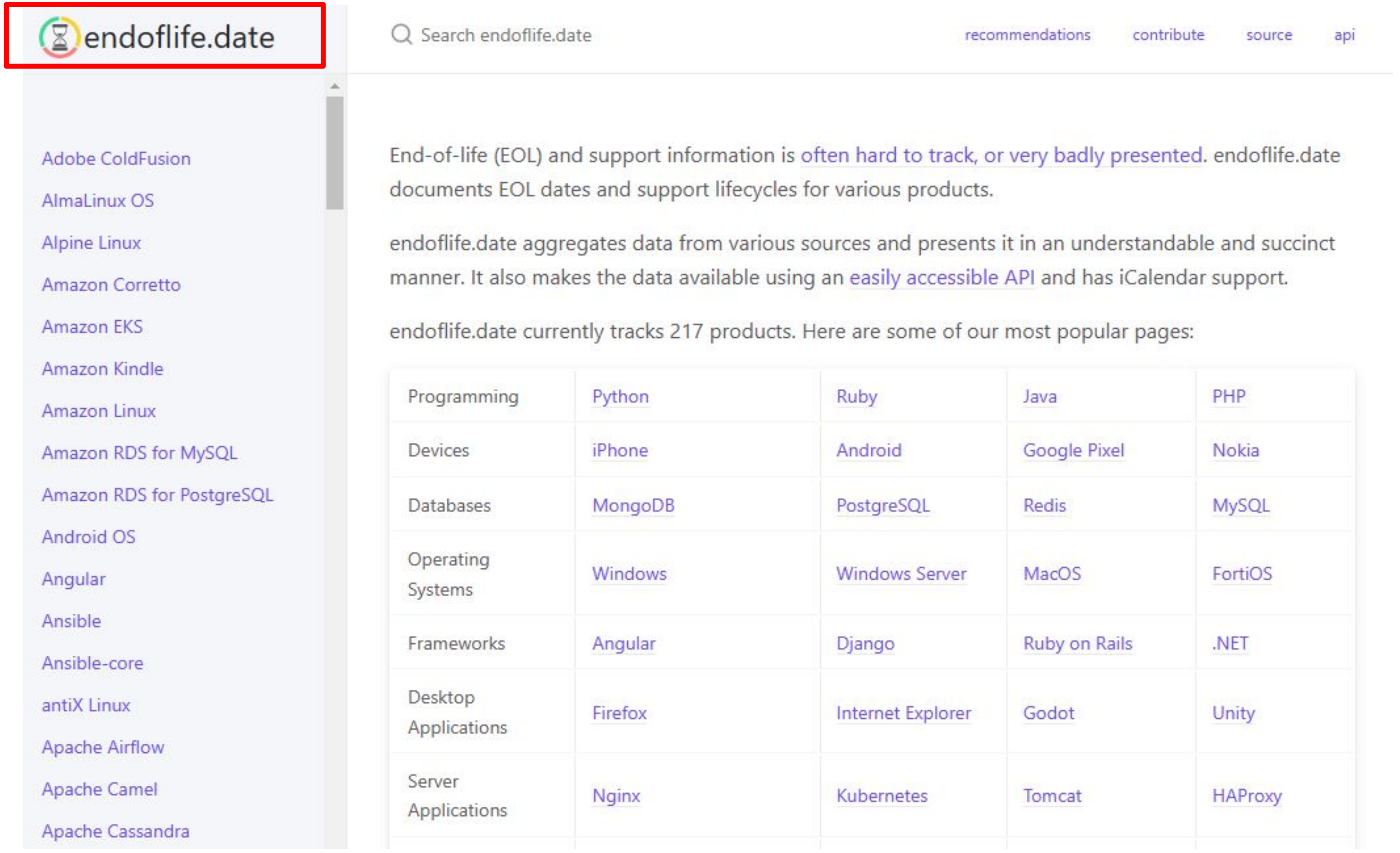
Support Cycle

What features do you include, make functional, and maintain?

- the more features you have the more support you need.
- the longer your life cycle, the more support you need.
- release model: calendar, feature, and whim-based.

Support: provide technological help in limited cases, long-term support for paying customers.

- support phase: set a time interval that makes sense (one year for free, lifetime for paying customers).
- maintain/support cycle: make changes/record of problems to address in new release.



endoflife.date

Search endoflife.date

recommendations contribute source api

Adobe ColdFusion
AlmaLinux OS
Alpine Linux
Amazon Corretto
Amazon EKS
Amazon Kindle
Amazon Linux
Amazon RDS for MySQL
Amazon RDS for PostgreSQL
Android OS
Angular
Ansible
Ansible-core
antiX Linux
Apache Airflow
Apache Camel
Apache Cassandra

End-of-life (EOL) and support information is often hard to track, or very badly presented. endoflife.date documents EOL dates and support lifecycles for various products.

endoflife.date aggregates data from various sources and presents it in an understandable and succinct manner. It also makes the data available using an easily accessible API and has iCalendar support.

endoflife.date currently tracks 217 products. Here are some of our most popular pages:

Programming	Python	Ruby	Java	PHP
Devices	iPhone	Android	Google Pixel	Nokia
Databases	MongoDB	PostgreSQL	Redis	MySQL
Operating Systems	Windows	Windows Server	MacOS	FortiOS
Frameworks	Angular	Django	Ruby on Rails	.NET
Desktop Applications	Firefox	Internet Explorer	Godot	Unity
Server Applications	Nginx	Kubernetes	Tomcat	HAProxy

Schedule Model

Calendar-based: cycle ends when a certain date is reached (regular timing).

- Raspberry Pi: March 14 (3/14). Fedora Linux: third Tuesday of April.

Schedule Model

Calendar-based: cycle ends when a certain date is reached (regular timing).

- Raspberry Pi: March 14 (3/14). Fedora Linux: third Tuesday of April.

Feature-based: scope and size of release-dependent (irregular timing).

- how many features you complete work in a certain period of time. Bundle interdependent features. Fewer releases to worry about and support.

Schedule Model

Calendar-based: cycle ends when a certain date is reached (regular timing).

- Raspberry Pi: March 14 (3/14). Fedora Linux: third Tuesday of April.

Feature-based: scope and size of release-dependent (irregular timing).

- how many features you complete work in a certain period of time. Bundle interdependent features. Fewer releases to worry about and support.

Whim-based: make a release whenever you want (irregular timing).

- release whenever work is complete. Less intra-release structure to manage.

Issues and Milestones

Milestones: points in time that define goals and major releases.

- dependencies determine whether milestones are met.

Issues: project issues out to meet milestones.

- issues can be reused for sequential releases.

Issues and Milestones

Milestones: points in time that define goals and major releases.

- dependencies determine whether milestones are met.

Issues: project issues out to meet milestones.

- issues can be reused for sequential releases.

Documentation and release notes

Accommodate external conflicts and opportunities (WWDC)

Release candidates

Translations

Communicating schedule

Managing Feature Cycles

From Chapter 9, “Program Management for Open Source Projects”
(Ben Cotton).

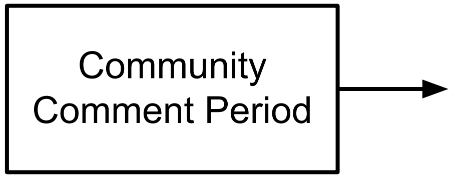
Manage features as a series of issues and milestones.

- templates: can be used to define scope, testing plan, contingency plan, and rationale.
- scale and approval process: who decides what is included in a formal release? What is too detailed, and what is too trivial.

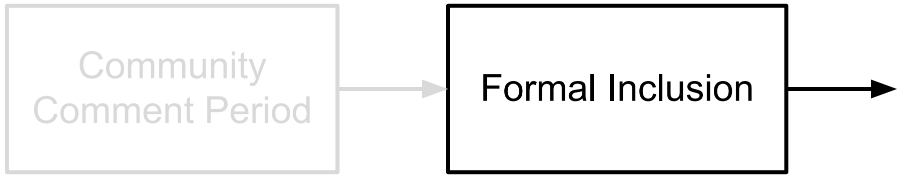
How are features enforced?

- feature wranglers: open-source leaders or centralized managers.
- feature lifecycle: proposal window → timeline → completion path.

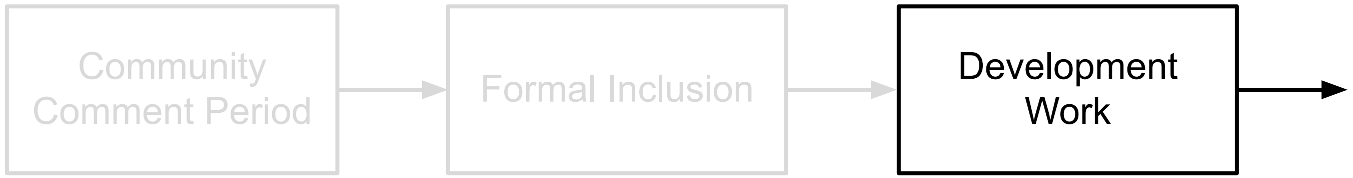
Feature Lifecycles



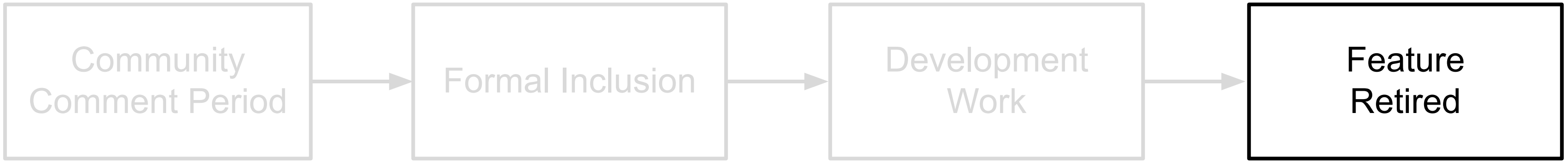
Feature Lifecycles



Feature Lifecycles



Feature Lifecycles



Understanding the open source software life cycle

https://www.redhat.com/en/resources/open-source-software-life-cycle-brief

Stages of open-source software development

1. collaborators and users engage with software architecture and develop support infrastructure.

2. software matures as people become dependent on it. A variety of uses, for a variety of skill levels.

3. new version releases occur, software becomes the basis of new development opportunities.

4. software is no longer viable, community breaks down.

Models for Software Development Lifecycle

Ruparelia, N. (2010). Software development lifecycle model. A*CM SIGSOFT Software Engineering Notes*, doi:10.1145/1764810.1764814

Software development lifecycle models can be linear, iterative, or hybrid (linear/iterative).

Models for Software Development Lifecycle

Ruparelia, N. (2010). Software development lifecycle model. A*CM SIGSOFT Software Engineering Notes*, doi:10.1145/1764810.1764814

Software development lifecycle models can be linear, iterative, or hybrid (linear/iterative).

1) Waterfall (iterative feedback)

operational analysis → operational specification → design/coding specifications → development → testing → deployment → evaluation

Models for Software Development Lifecycle

Ruparelia, N. (2010). Software development lifecycle model. A*CM SIGSOFT Software Engineering Notes*, doi:10.1145/1764810.1764814

Software development lifecycle models can be linear, iterative, or hybrid (linear/iterative).

1) Waterfall (iterative feedback)

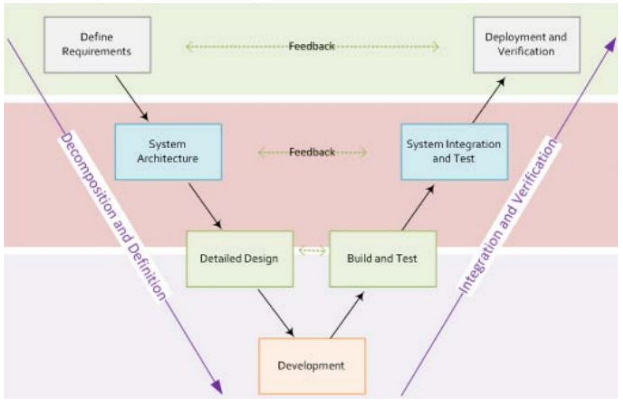
operational analysis → operational specification → design/coding specifications → development → testing → deployment → evaluation

2) B-model (extension of the waterfall model)

Linear phase	inception → define requirements → design → production → accept
Maintenance cycle	operation → inception → analysis → design → production → acceptance

Models for Software Development Lifecycle (con't)

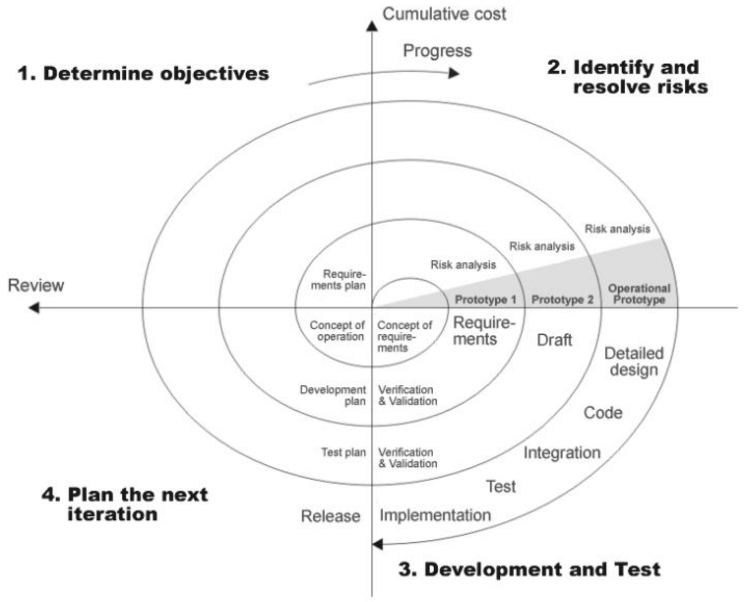
3) U-model: decompose requirement down to the development stage, then integrate and verify to the deploy stage.


Define Requirements
Feedback
Deployment and Verification
System Architecture
Feedback
System Integration and Test
Detailed Design
Build and Test
Development
Decomposition and Definition
Integration and Verification

Models for Software Development Lifecycle (con't)

3) U-model: decompose requirement down to the development stage, then integrate and verify to the deploy stage.

4) Spiral model: start small, think big.



Models for Software Development Lifecycle (con’t)

3) U-model: decompose requirement down to the development stage, then integrate and verify to the deploy stage.

4) Spiral model: start small, think big.

5) Unified Process Model (iterative, integrated with UML): architecture-based, case driven.

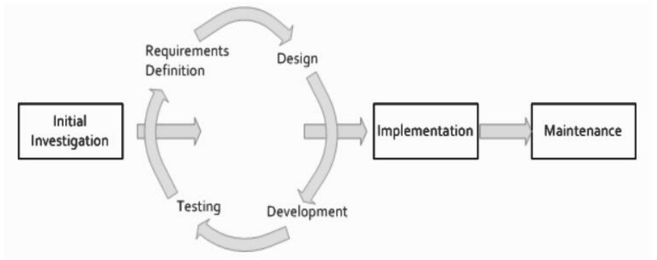
Models for Software Development Lifecycle (con't)

3) U-model: decompose requirement down to the development stage, then integrate and verify to the deploy stage.

4) Spiral model: start small, think big.

5) Unified Process Model (iterative, integrated with UML): architecture-based, case driven.

6) Cathedral and Bazaar: release early, release often, listen to your customers.


Initial Investigation
Requirements Definition
Design
Development
Testing
Implementation
Maintenance