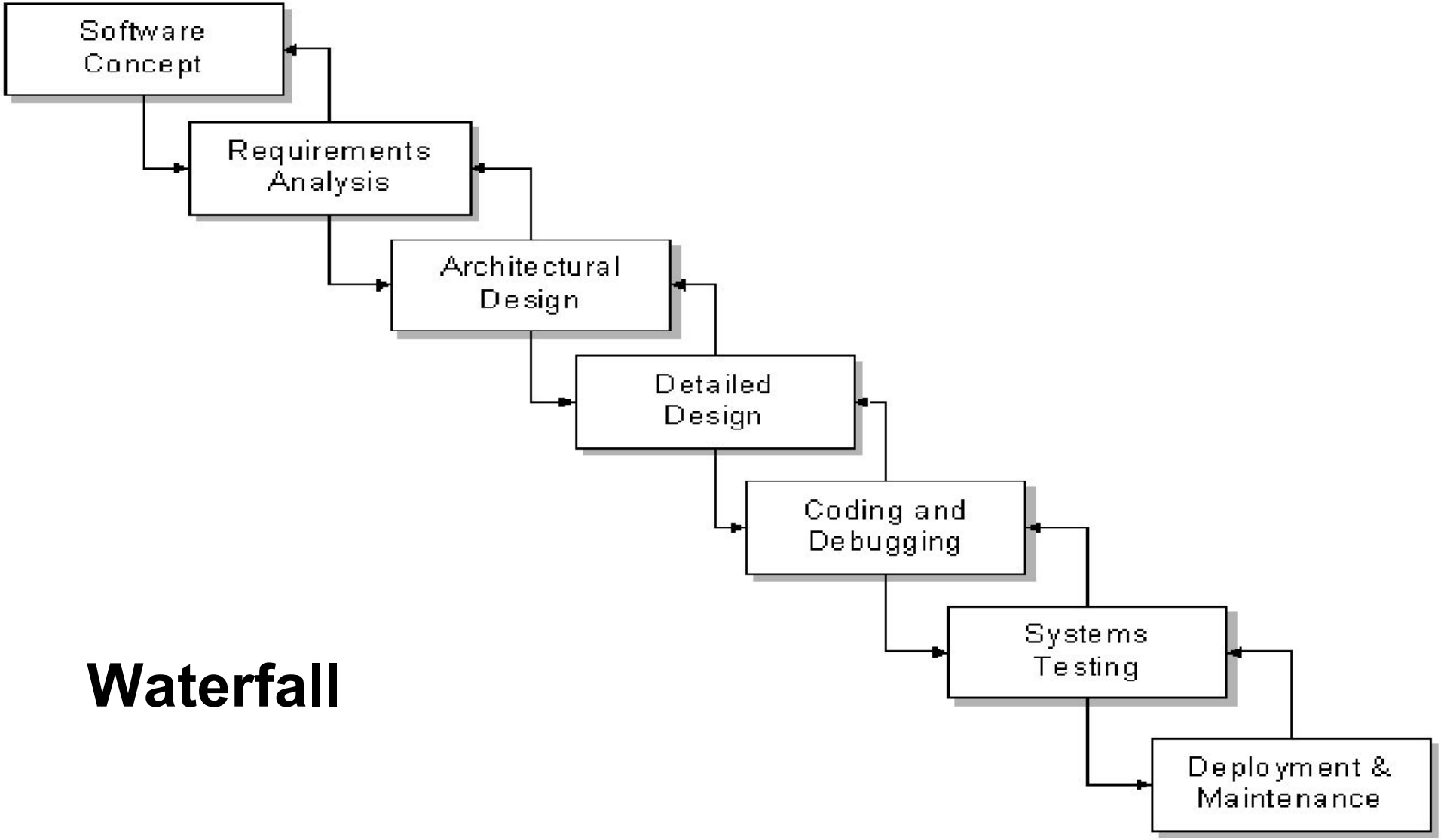
Towards developing a product

Brief introduction to lifecycles
and
Requirements

Software Project Lifecycles

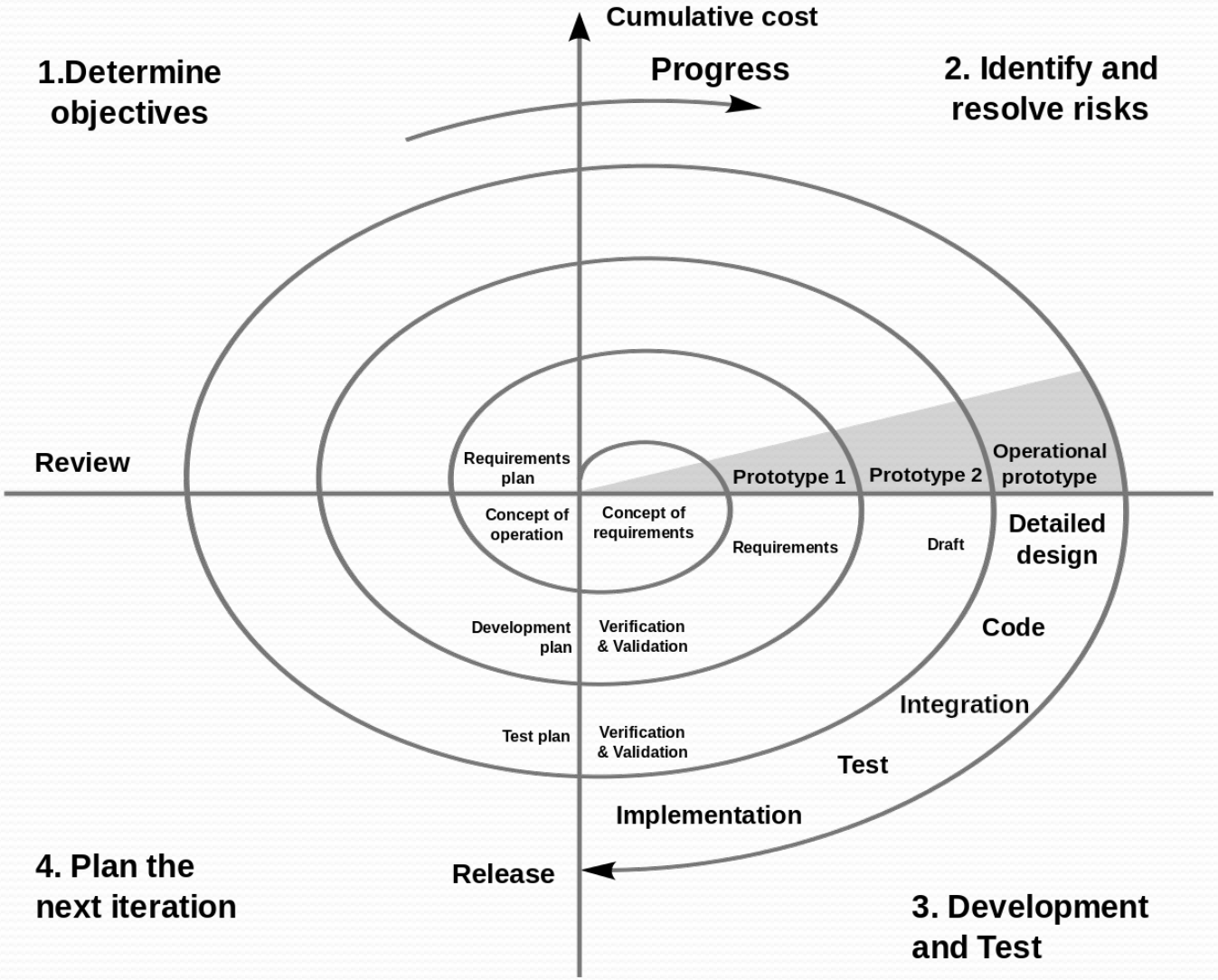
- Projects are decomposed into phases
- Project Lifecycle describes the project in terms of its phases
- Each phase
  - Produces a set of deliverables
- Phase is finished when the deliverables are completed

Software Project Lifecycles



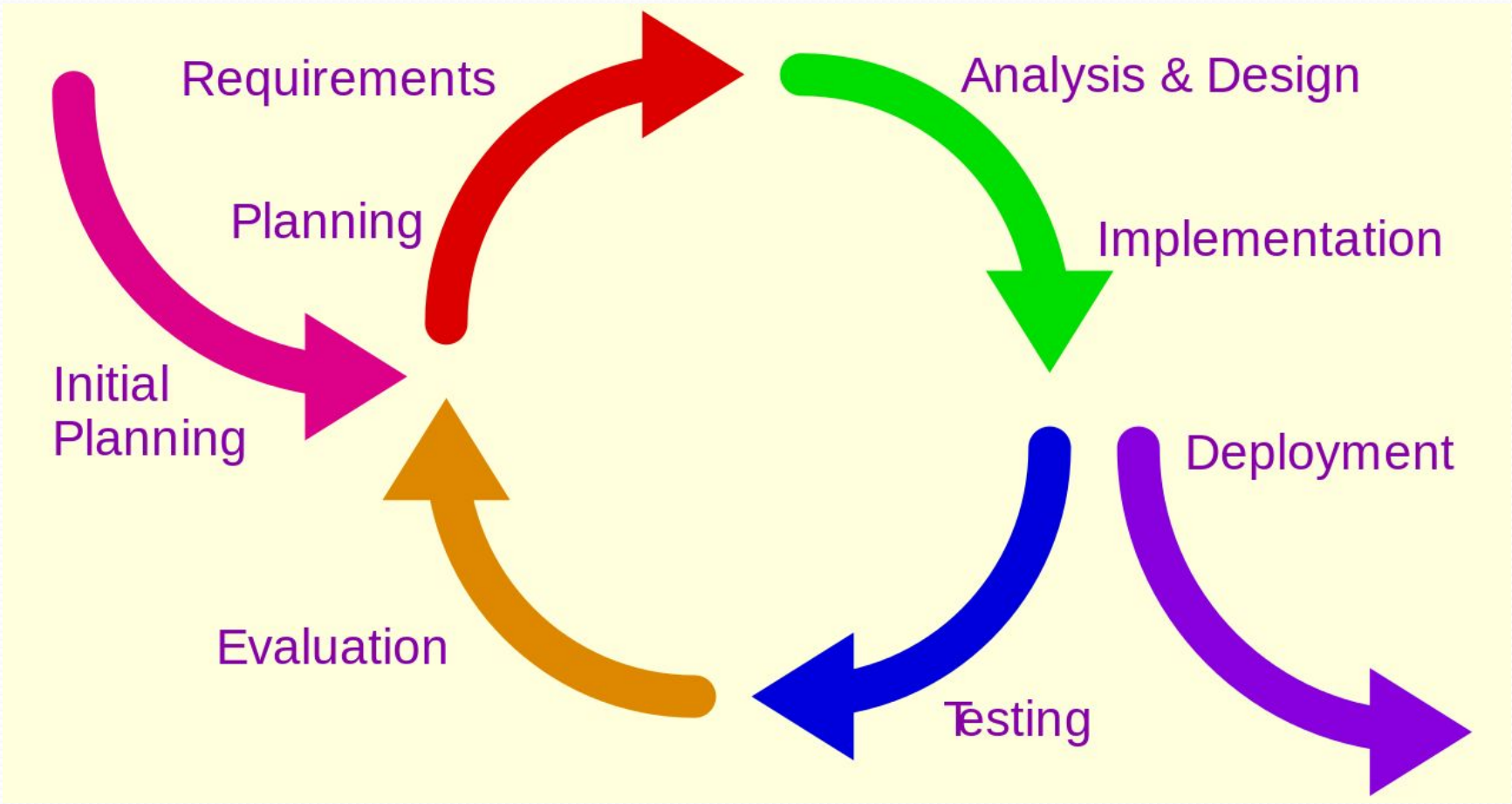
Waterfall

Spiral Lifecycles



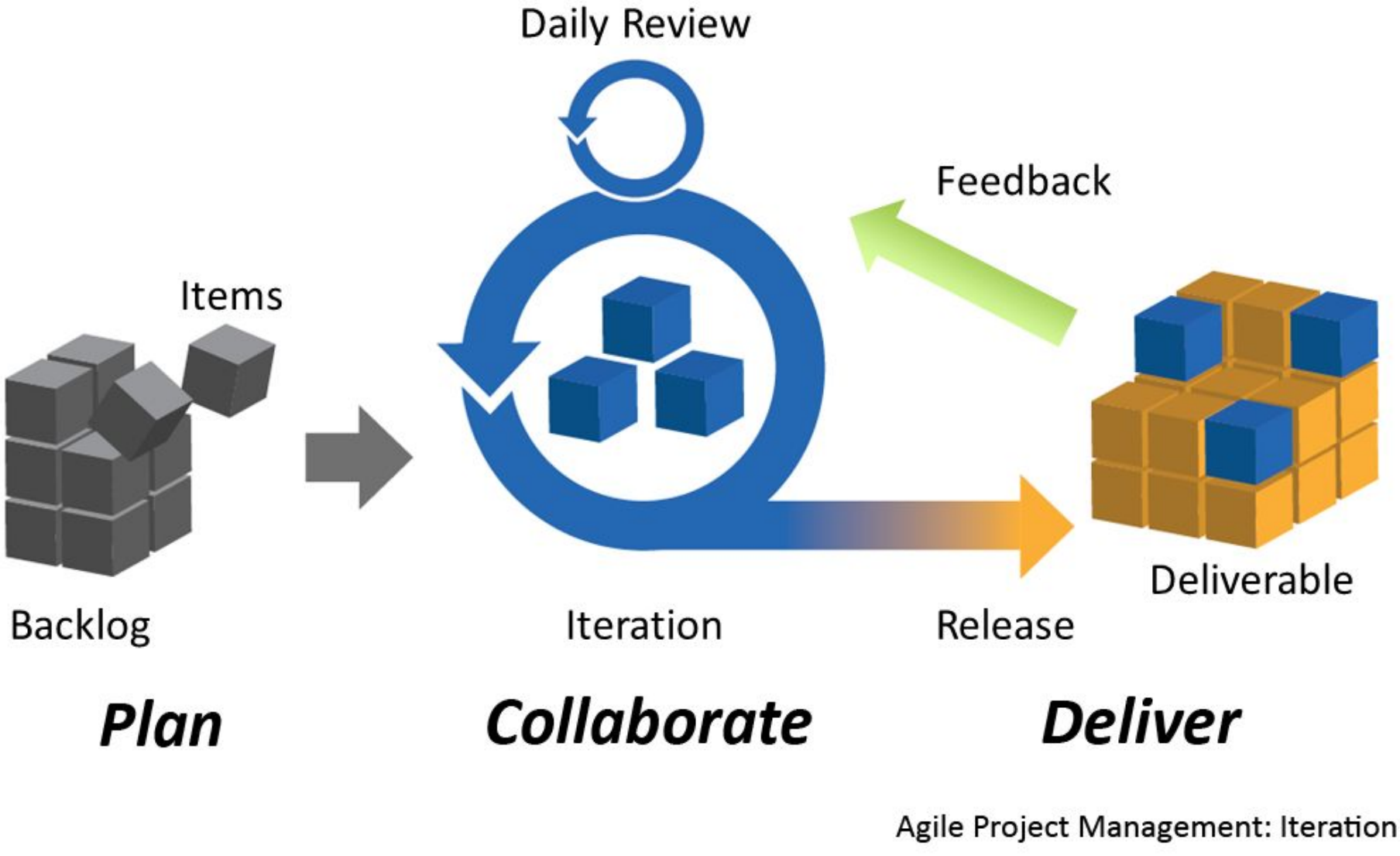
By Connyderivative work: Spiral_model_(Boehm,_1988).png: Marctroyderivative work: Conan (talk) - File:Spiralmodel_nach_Boehm.png, Spiral_model_(Boehm,_1988).png, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9000950

Iterative Lifecycle



By Aflafla1 - Iterative development model V2.jpg , User:Westerhoff, CC0, https://commons.wikimedia.org/w/index.php?curid=34159246

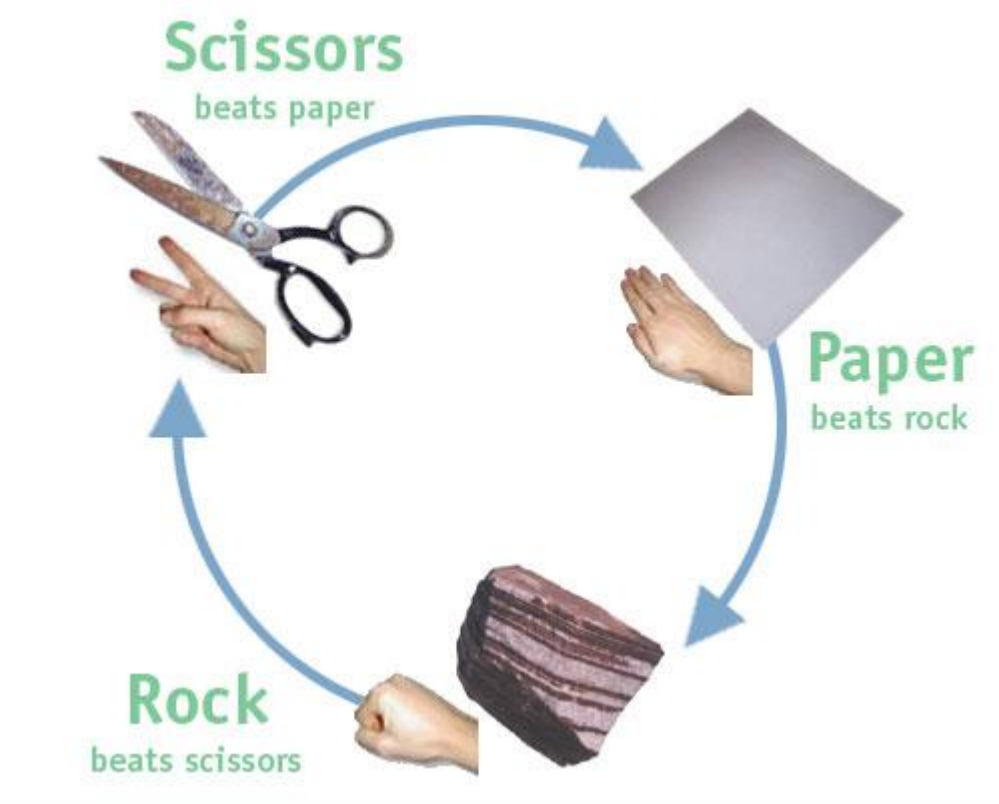
Agile Lifecycle



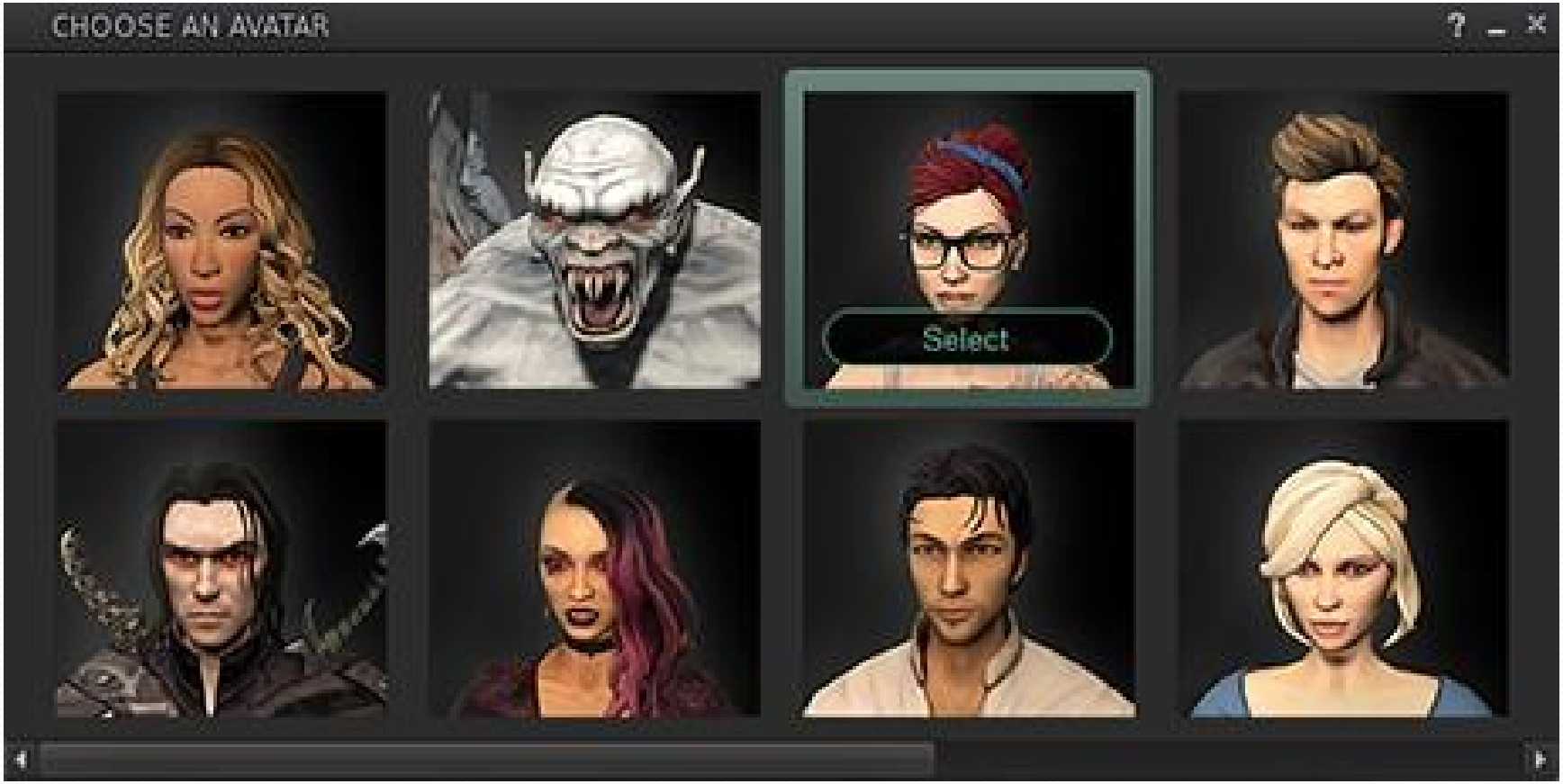
Requirements Engineering

Understanding **WHAT** to build

How do you specify a Rock Scissors Paper game for children?



Second Life Type of Avatar



SECOND LIFE MARKETPLACE

Items | Merchants/Stores | Search this store | Cart (0)

Search in: All categories | Show maturity levels: General | Keywords | Search

General maturity level: Recreation and Entertainment » Miscellaneous Fun

Kermit's Flying Pig (Box)

Zoom

L$0

Add To Cart

Kermit Rutkowski
Visit The Store

Use It Now

This item will be delivered directly to you or a friend in Second Life, unpacked and ready to use. No land or sandbox required.

★★★★★ Reviews (2)

Permissions:
✓Copy ✓Modify ~~Transfer~~

Prim count: 14

Add To Favorites

Like 0

Share this item |

Flag this item

Details | Features | Contents | Reviews (2)

This a box of two pigs. One to wear and fly around and a smaller one to follow you around. Turn off your AO and wear the pig. Rez the smaller one to follow you around. Click on the riding pig to play sound. Hope you have fun!

Sponsored links


Autodesk Revit 2013 - rac_basic_sample_project.rvt - 3D View: Aerial
Graphic Display Options
Model Display
Style: Shaded
Show Edges
Transparency: 0
Silhouettes: Wide Lines
Shadows
Cast Shadows
Show Ambient Shadows
Lighting
Scheme: Exterior: Sun only
Sun Setting: Summer Solstice
Artificial Lights: Artificial Lights...
Sun: 30
Ambient Light: 60
Shadows: 50
Photographic Exposure
Background
Save as View Template...
OK
Cancel
Apply
Perspective
Drawing: rac_basic_sample_project.rvt - 3D View: Aerial
Main Model

Requirements engineering

- Process of figuring out
  - Services the customer needs
  - Constraints of operation

- It is about WHAT will be built!

Why Develop Requirements Specs?

I believe that on any non-trivial project (more than about 1 week of coding or more than 1 programmer), if you don't have a spec, you will *always* spend more time and create lower quality code.

Joel Spolsky

http://www.joelonsoftware.com

Requirement

- Descriptions of
  - system services
  - constraints

- Gathered during the requirements engineering process.

Use of Requirement Specs

- Design
- Communicate
- Test

Types of Requirements

- Functional
  - Behavior of system
  - From user's point of view
- Non-functional
  - Non behavior related constraints
  - Constraints on the services
    - i.e timing, development process, standards,
  - Apply to whole system rather than functions

Types of requirement

- **User** requirements
  - Written for customers
  - Natural Language
  - Diagrams
- **System** requirements
  - Detailed descriptions system functions, services, and operational constraints.

Requirement Language

- Requirements are often written in natural language (e.g., English).
  - inherently ambiguous
  - should be **reviewed** by an independent party to identify ambiguous language so that it can be corrected

Formal Specification -- VDM

p1 : Path = mk_token("A1North");
p2 : Path = mk_token("A1South");
p3 : Path = mk_token("A66East");
p4 : Path = mk_token("A66West");
lights : map Path to Light
= {**p1** |-> <Red>,
p2 |-> <Red>,
p3 |-> <Green>,
p4 |-> <Green>};

Clear description

- Must be precise
- Ambiguous requirements
  - Different interpretation
- How can we avoid ambiguity?

Guidelines for writing requirements

- Use a standard format
- Be consistent
- Use **shall** for mandatory requirements
- Use **should** for desirable requirements
- Highlight key parts
- Use **structure** to group related requirements
- Enumerate !!! (Why?)

Non-functional requirements

- System properties and constraints
  - Up time
  - Response time
  - Storage requirements
  - Usability
- Process requirements
  - IDE
  - Programming language
  - Development method.

Verifiable Non-functional Requirement Description

- Verifiable non-functional requirement
  - Measurable
  - Can be tested

- Difficult to state **precisely** → difficult to verify.

Metrics for nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames

Metrics for NF Req. (cont.)

Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Number of target systems

Good Software Requirement Specifications (SRS)

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

Correctness

- With external objects
  - Incorrect descriptions of real objects
  - Ex: Blue background vs Green background
- Logical (A x B vs A / B)
- Temporal (A after B vs A and B simultaneously)
- Note: Use consistent and precise **terminology**
- Agreement with terminology in a project team is crucial

Completeness & Consistency

- Should be complete & consistent
- Complete
  - All required functionality is stated
- Consistent
  - There are no conflicts between requirements

- In practice: Impossible

Requirements engineering processes

- Requirements elicitation
- Requirements analysis
- Requirements validation
- Requirements management
- In practice
  - iterative activity
  - processes are interleaved.

Requirements elicitation and analysis

- Requirements discovery
- Requirements classification and organization
- Requirements prioritization and negotiation
- Requirements specification

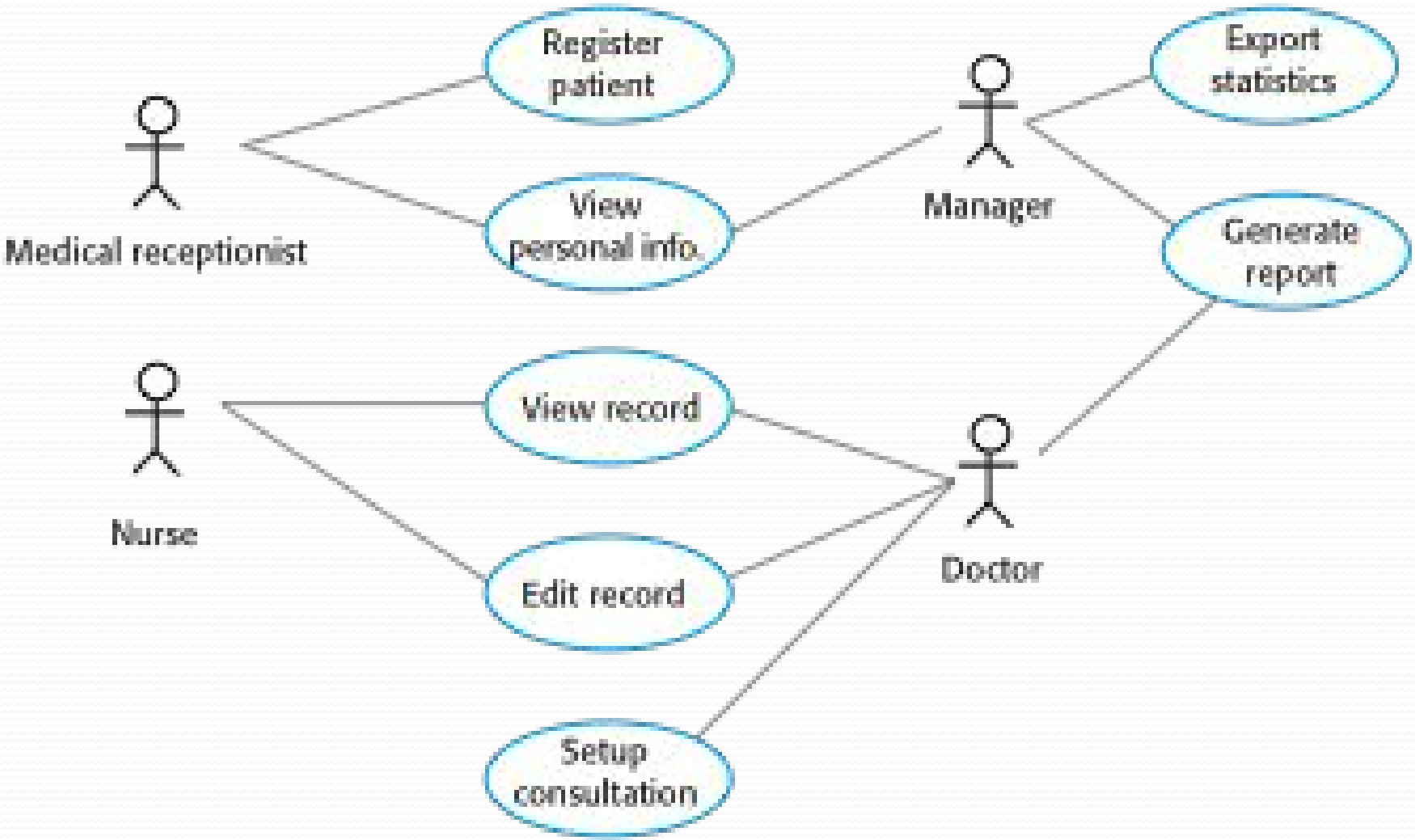
Scenarios

- Scenarios are real-life examples
- Consists of
  - Initial situation
  - Normal flow of events
  - What can go wrong
  - Information about other concurrent activities
  - Terminating situation

Use cases

- Scenario based technique in the UML
- Identifies the **actors** and the interaction
- A set of **use cases** should describe all possible interactions with the system.

Use cases for Hospital System



Requirements validation

- Do the requirements define the system that the customer really wants?
- Requirements error is very costly

Requirements checking

- Validity. Does the system provide the functions which support customer needs?
- Consistency. Are there requirements conflicts?
- Completeness. Are all functions required by the customer included?
- Realism. Can the requirements be implemented given available budget/technology
- Verifiability. Can the requirements be checked?

Requirements validation

- Requirements reviews
  - Systematic manual analysis of requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

Review checks

- Verifiability
  - Is the requirement realistically testable?
- Comprehensibility
  - Is the requirement properly understood?
- Traceability
  - Is the origin of the requirement clearly stated?
- Adaptability
  - Can the requirement be changed without a large impact on other requirements?

Learnings

- What software requirements are
- Elicitation
- How to write requirements
- Good practices
- Validation