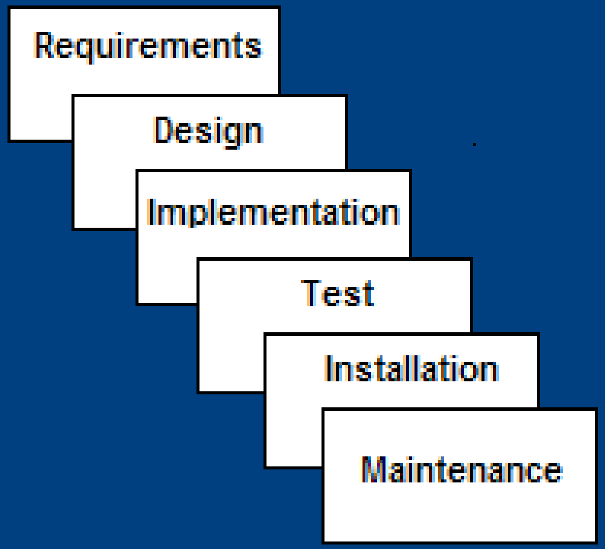
"You've got to be very careful if you don't know where you're going, because you might not get there."

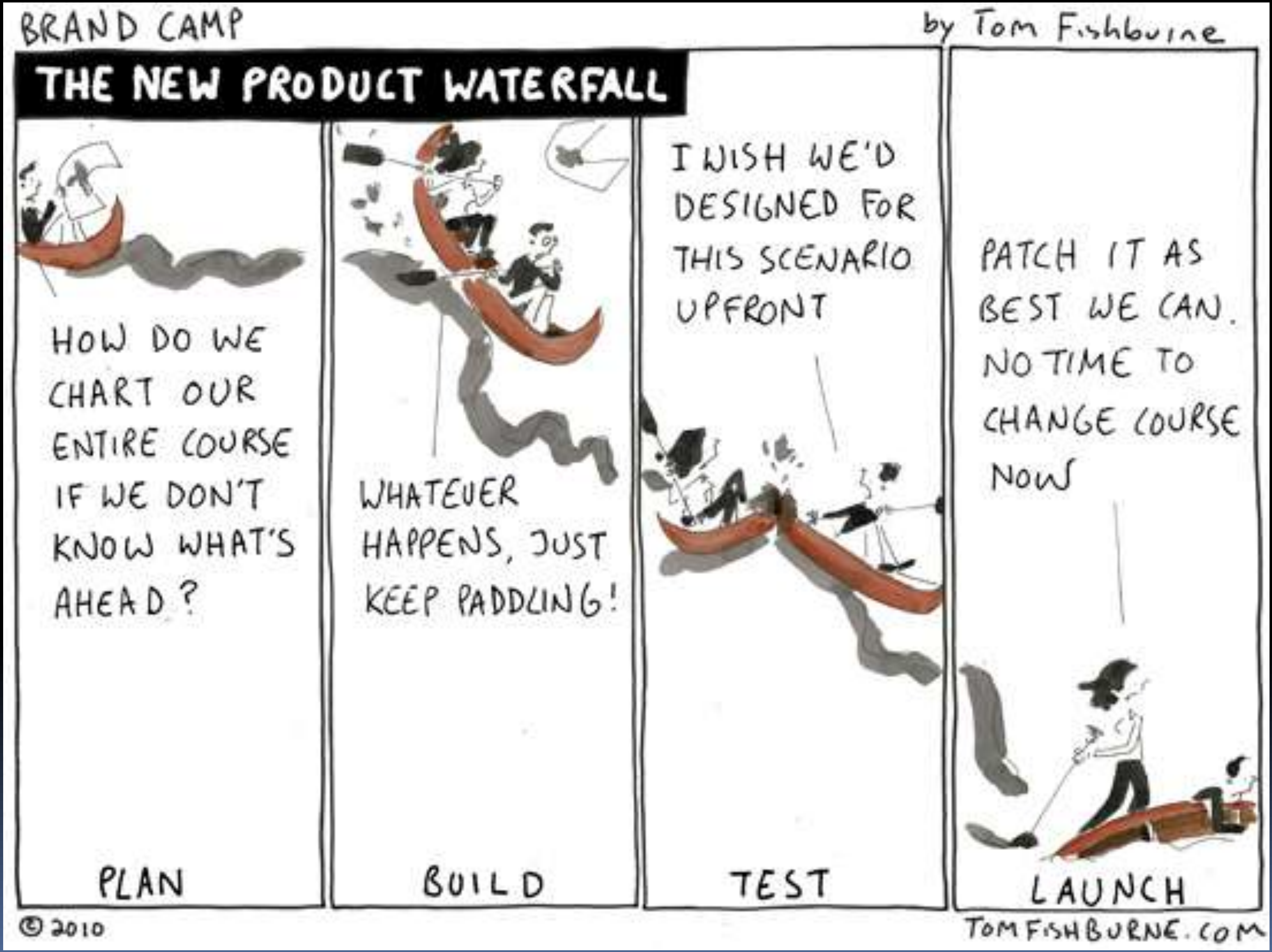
Yogi Berra

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Waterfall Model



- **Requirements** – defines needed information, function, behavior, performance and interfaces.
- **Design** – data structures, software architecture, interface representations, algorithmic details.
- **Implementation** – source code, database, user documentation, testing.


BRAND CAMP
by Tom Fishburne
THE NEW PRODUCT WATERFALL
HOW DO WE CHART OUR ENTIRE COURSE IF WE DON'T KNOW WHAT'S AHEAD?
PLAN
WHATEVER HAPPENS, JUST KEEP PADDLING!
BUILD
I WISH WE'D DESIGNED FOR THIS SCENARIO UPFRONT
TEST
PATCH IT AS BEST WE CAN. NO TIME TO CHANGE COURSE NOW
LAUNCH
© 2010
TOMFISHBURNE.COM

Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Waterfall Deficiencies

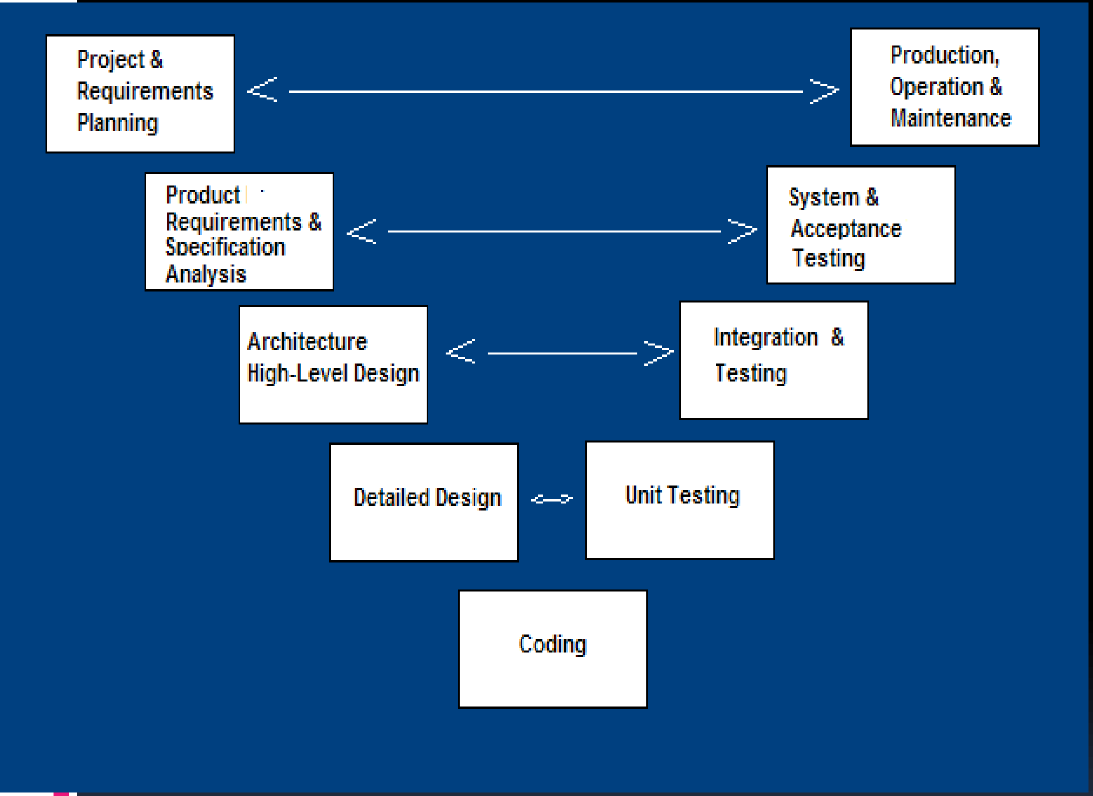
- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

  - High risk for new systems because of specification and design problems.
  - Low risk for well-understood developments using familiar technology.

V-Shaped SDLC Model


Project & Requirements Planning
Production, Operation & Maintenance
Product Requirements & Specification Analysis
System & Acceptance Testing
Architecture High-Level Design
Integration & Testing
Detailed Design
Unit Testing
Coding

- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development

V-Shaped Steps

- Project and Requirements Planning – allocate resources
- Product Requirements and Specification Analysis – complete specification of the software system
- Architecture or High-Level Design – defines how software functions fulfill the design
- Detailed Design – develop algorithms for each architectural component

- Production, operation and maintenance – provide for enhancement and corrections
- System and acceptance testing – check the entire software system in its environment
- Integration and Testing – check that modules interconnect correctly
- Unit testing – check that each module acts as expected
- Coding – transform algorithms into software

V-Shaped Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

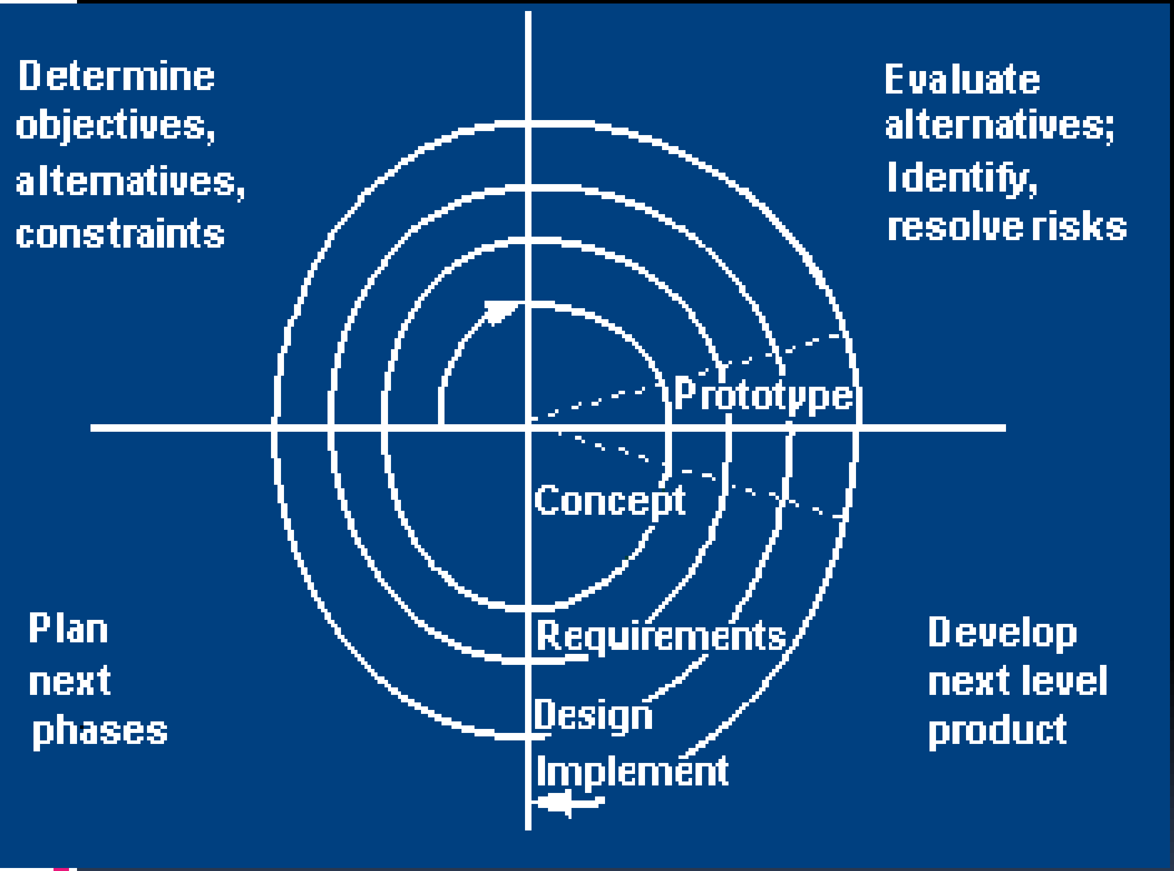
V-Shaped Weaknesses

- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

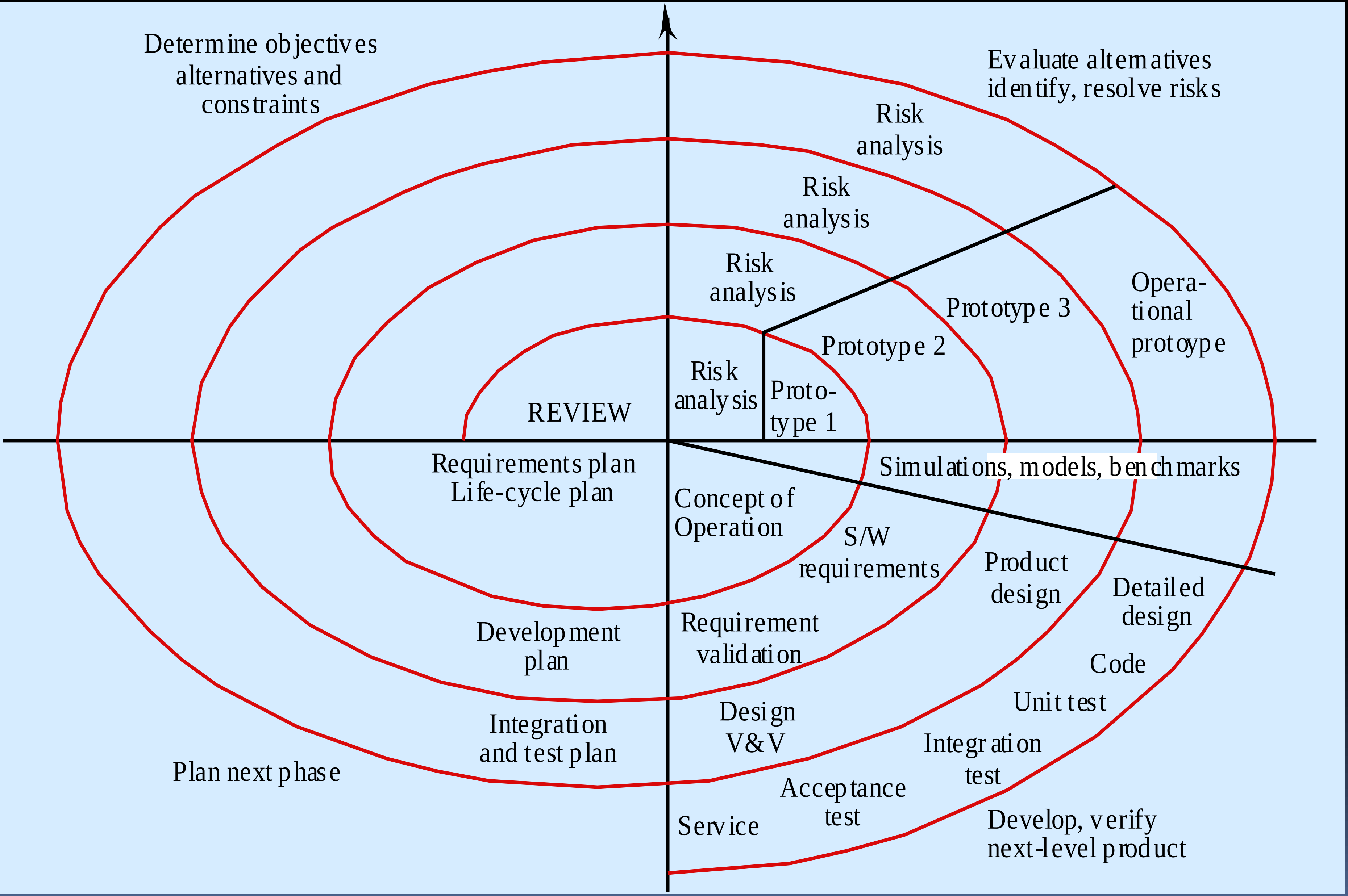
When to use the V-Shaped Model

- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

Spiral SDLC Model


Determine objectives, alternatives, constraints
Evaluate alternatives; Identify, resolve risks
Prototype
Concept
Requirements
Design
Implement
Plan next phases
Develop next level product

- Adds risk analysis, and 4gl RAD prototyping to the waterfall model
- Each cycle involves the same sequence of steps as the waterfall process model


Determine objectives alternatives and constraints
Evaluate alternatives identify, resolve risks
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Opera-tional protoype
Prototype 3
Prototype 2
Proto-type 1
REVIEW
Requirements plan Life-cycle plan
Concept of Operation
Simulations, models, benchmarks
S/W requirements
Product design
Detailed design
Code
Unit test
Integration test
Requirement validation
Development plan
Design V&V
Integration and test plan
Acceptance test
Service
Plan next phase
Develop, verify next-level product

Spiral Quadrant: Determine objectives, alternatives and constraints

- Objectives: functionality, performance, hardware/software interface, critical success factors, etc.
- Alternatives: build, reuse, buy, sub-contract, etc.
- Constraints: cost, schedule, interface, etc.

Spiral Quadrant: Evaluate alternatives, identify and resolve risks

- Study alternatives relative to objectives and constraints
- Identify risks (lack of experience, new technology, tight schedules, poor process, etc.
- Resolve risks (evaluate if money could be lost by continuing system development

Spiral Quadrant: Develop next-level product

- Typical activites:
  - Create a design
  - Review design
  - Develop code
  - Inspect code
  - Test product

Spiral Quadrant: Plan next phase

- Typical activities
  - Develop project plan
  - Develop configuration management plan
  - Develop a test plan
  - Develop an installation plan

Spiral Model Strengths

- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral Model

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

AGILE SOFTWARE DEVELOPMENT LIFE CYCLES

Agile SDLC’s

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

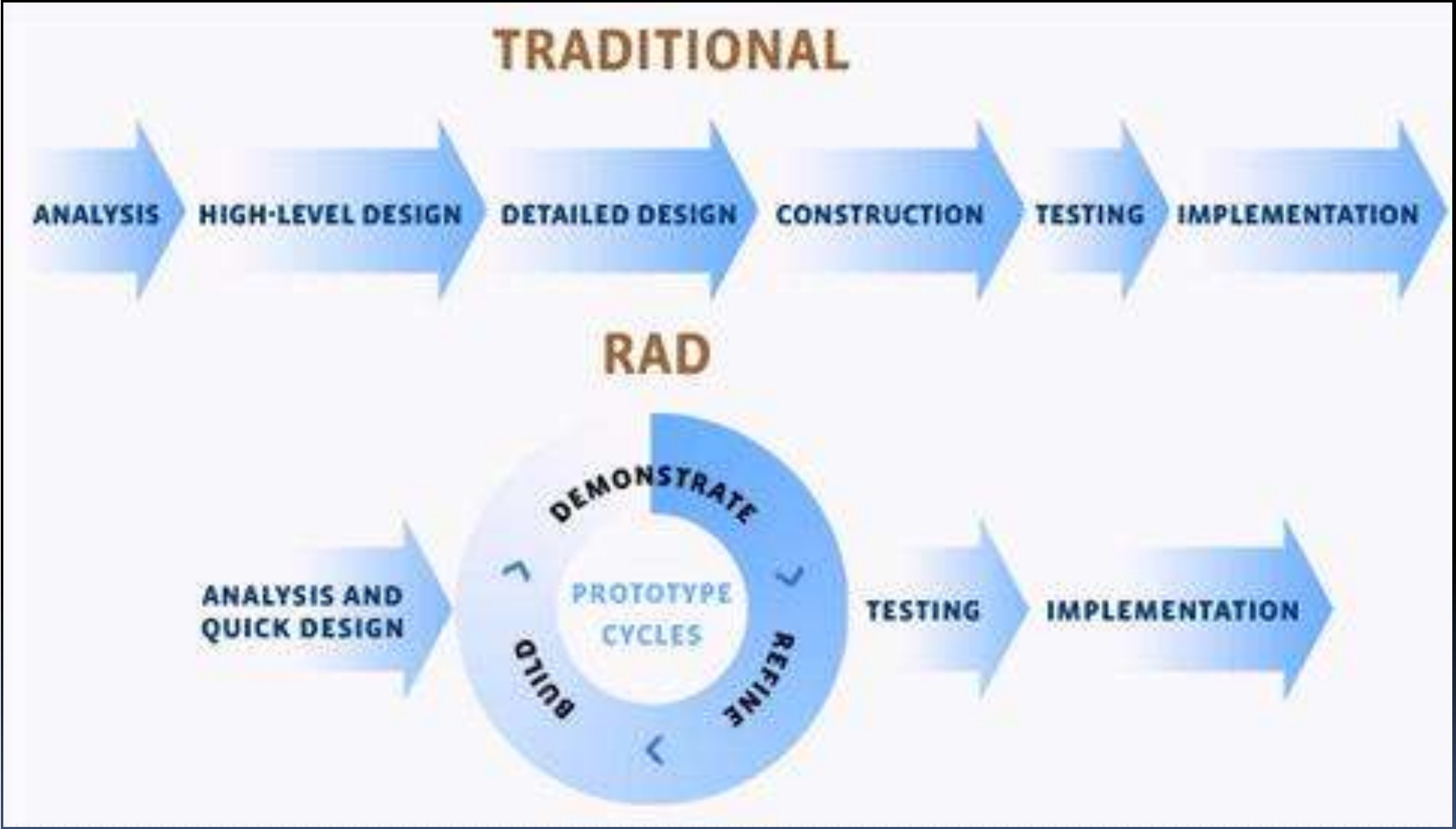
Some Agile Methods

- Rapid Application Development (RAD)
- Incremental SDLC
- Scrum
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rational Unify Process (RUP)

Agile vs Waterfall Propaganda

- http://www.youtube.com/watch?v=gDDO3ob-4ZY&feature=related

RAPID APPLICATION DEVELOPMENT (RAD) MODEL


TRADITIONAL
ANALYSIS
HIGH-LEVEL DESIGN
DETAILED DESIGN
CONSTRUCTION
TESTING
IMPLEMENTATION
RAD
ANALYSIS AND QUICK DESIGN
DEMONSTRATE
REFINE
BUILD
PROTOTYPE CYCLES
TESTING
IMPLEMENTATION

Rapid Application Model (RAD)

- Requirements planning phase (a workshop utilizing structured discussion of business problems)
- User description phase – automated tools capture information from users
- Construction phase – productivity tools, such as code generators, screen generators, etc. inside a time-box. ("Do until done")
- Cutover phase -- installation of the system, user acceptance testing and user training

Requirements Planning Phase

- Combines elements of the system planning and systems analysis phases of the System Development Life Cycle (SDLC).
- Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements.
- It ends when the team agrees on the key issues and obtains management authorization to continue.

RAD Strengths

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code (WYSIWYG).
- Uses modeling concepts to capture information about business, data, and processes.

RAD Weaknesses

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

When to use RAD

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- High performance not required
- Low technical risks
- System can be modularized

User Design Phase

- Users interact with systems analysts and develop models and prototypes that represent all system processes, inputs, and outputs.
- Typically use a combination of Joint Application Development (JAD) techniques and CASE tools to translate user needs into working models.
- *A* continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.

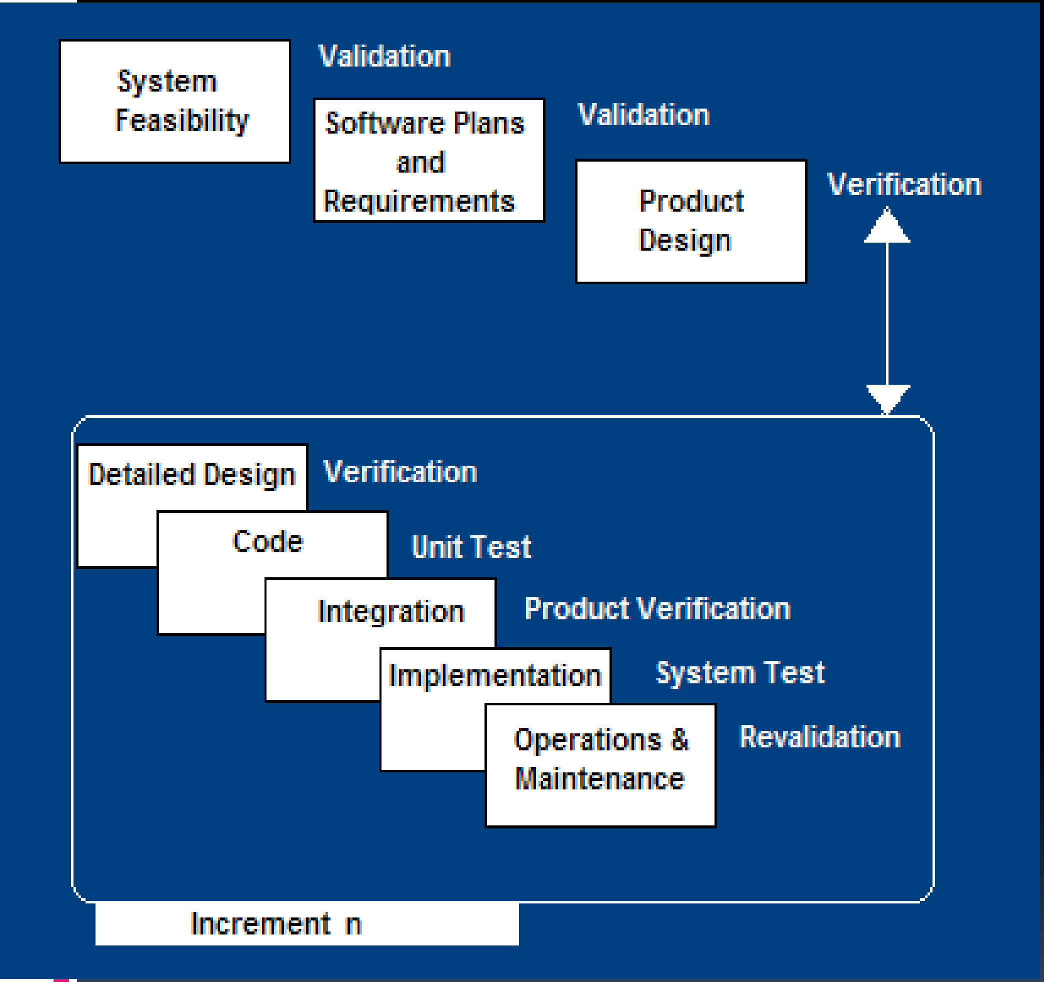
JAD Techniques

- http://en.wikipedia.org/wiki/Joint_application_design

CASE Tools

- http://en.wikipedia.org/wiki/Computer-aided_software_engineering

Incremental SDLC Model



- Construct a partial implementation of a total system
- Then slowly add increased functionality
- The incremental model prioritizes requirements of the system and then implements them in groups.
- Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.

Incremental Model Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses “divide and conquer” breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

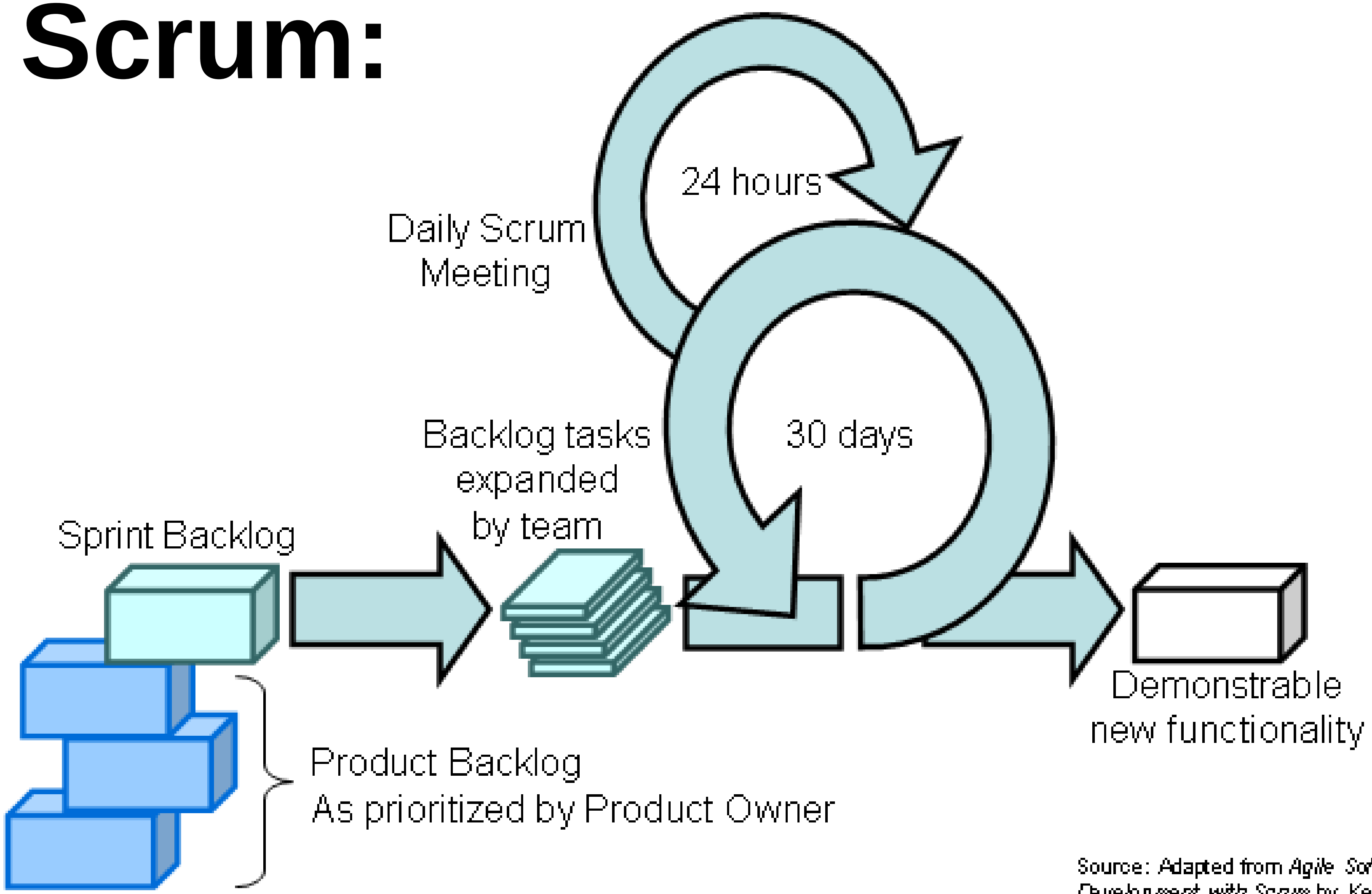
- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

When to use the Incremental Model

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

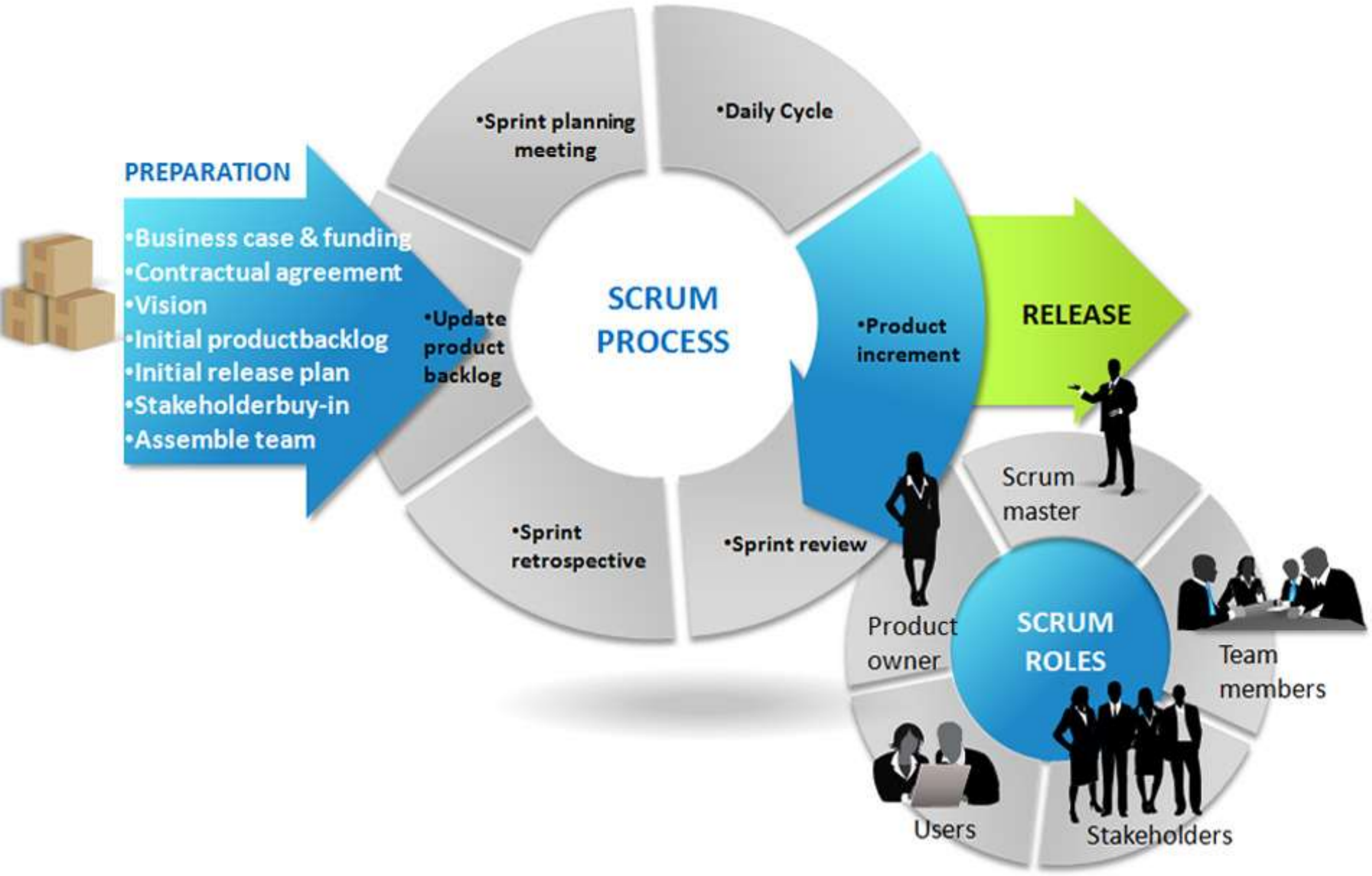
SCRUM

Scrum:


24 hours
Daily Scrum Meeting
Backlog tasks expanded by team
30 days
Sprint Backlog
Demonstrable new functionality
Product Backlog
As prioritized by Product Owner

Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

SCRUM PROCESS



- Scrum in 13 seconds:
  - http://www.youtube.com/watch?v=9DKM9HcRnZ8&feature=related
- Scrum in 10 minutes:
  - http://www.youtube.com/watch?v=Q5k7a9YEoUI
- More scrum slides:
  - http://www.mountaingoatsoftware.com/system/presentation/file/129/Getting-Agile-With-Scrum-Cohn-NDC2010.pdf?1276712017
  - Scalability of scrum addressed on slides 33-35

Scrum advantages

- Agile scrum helps the company in saving time and money.
- Scrum methodology enables projects where the business requirements documentation is hard to quantify to be successfully developed.
- Fast moving, cutting edge developments can be quickly coded and tested using this method, as a mistake can be easily rectified.

Scrum advantages

- It is a lightly controlled method which insists on frequent updating of the progress in work through regular meetings. Thus there is clear visibility of the project development.
- Like any other agile methodology, this is also iterative in nature. It requires continuous feedback from the user.
- Due to short sprints and constant feedback, it becomes easier to cope with the changes.

Scrum advantages

- Daily meetings make it possible to measure individual productivity. This leads to the improvement in the productivity of each of the team members.
- Issues are identified well in advance through the daily meetings and hence can be resolved in speedily
- It is easier to deliver a quality product in a scheduled time.

Scrum advantages

- Agile Scrum can work with any technology/ programming language but is particularly useful for fast moving web 2.0 or new media projects.
- The overhead cost in terms of process and management is minimal thus leading to a quicker, cheaper result.

Scrum disadvantages

- Agile Scrum is one of the leading causes of scope creep because unless there is a definite end date, the project management stakeholders will be tempted to keep demanding new functionality is delivered.
- If a task is not well defined, estimating project costs and time will not be accurate. In such a case, the task can be spread over several sprints.
- If the team members are not committed, the project will either never complete or fail.

Scrum disadvantages

- It is good for small, fast moving projects as it works well only with small team.
- This methodology needs experienced team members only. If the team consists of people who are novices, the project cannot be completed in time.
- Scrum works well when the Scrum Master trusts the team they are managing. If they practice too strict control over the team members, it can be extremely frustrating for them, leading to demoralisation and the failure of the project.

Scrum disadvantages

- If any of the team members leave during a development it can have a huge inverse effect on the project development
- Project quality management is hard to implement and quantify unless the test team are able to conduct regression testing after each sprint.