Software Engineering 301: Software Analysis and Design

Process

Agenda

- Basic concepts
- Process models
- Risk and reward
- Teamwork and communication
- Death march

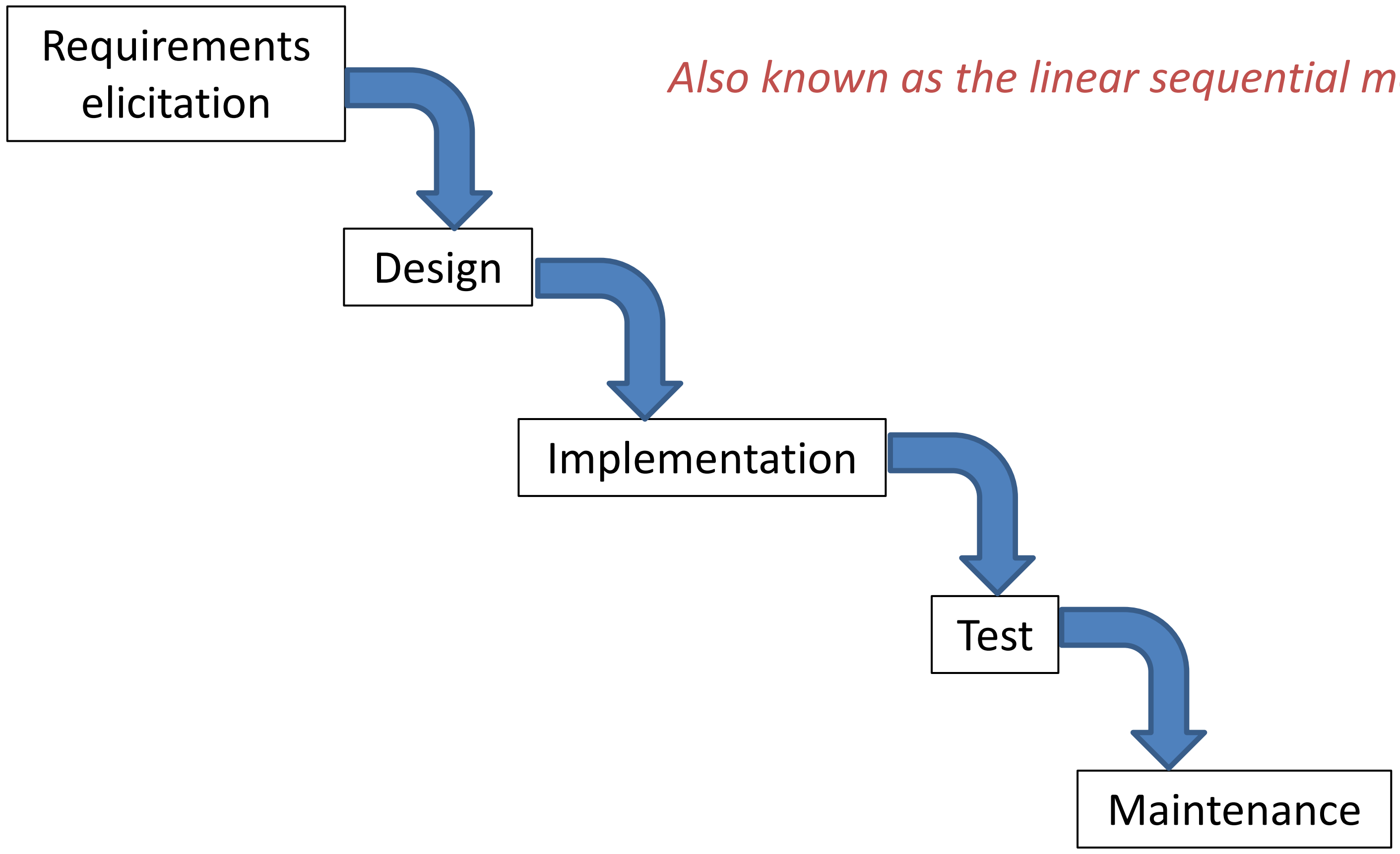
Process

- A description of the development strategy
  - Activities to be performed
  - Ordering constraints
  - Roles played by participants
  - Work products that are to be produced
- Assist in controlling and coordination of development projects
- Supports tracking and auditing

Process model

- Remember:
  - Design patterns need to be tailored to fit a situation
  - You can think of a design pattern as a generalization of a set of designs
- Similarly:
  - A process model is a generalization of a set of processes
  - Describes the “big picture” about how a process should be conducted
- Sometimes, process models are just called processes, but this is sloppy

The Waterfall Model

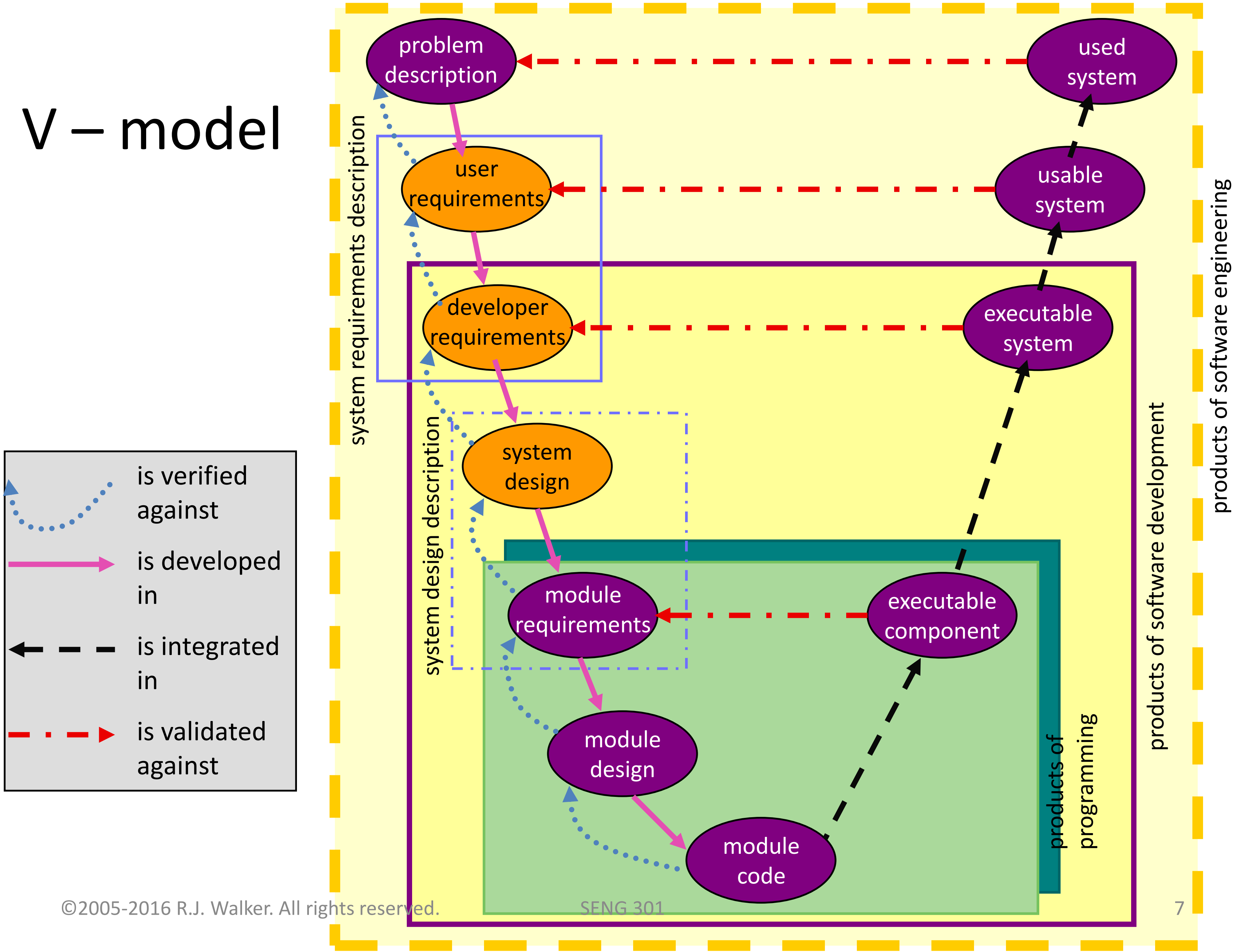


Also known as the linear sequential model

Issues with the Waterfall Model

- Real projects rarely follow a sequential flow
  - no feedback & change support
- Stating all requirements at the beginning of a project is difficult
- Customer must have patience
- Applicable for projects that are **very** well understood
  - i.e., ones that will not change (much)

V – model



problem description
used system
user requirements
usable system
developer requirements
executable system
system design
executable component
module requirements
module design
module code
system requirements description
system design description
products of software engineering
products of software development
products of programming
is verified against
is developed in
is integrated in
is validated against

Incremental versus iterative models

- Usually treated as one in the same, but they are not

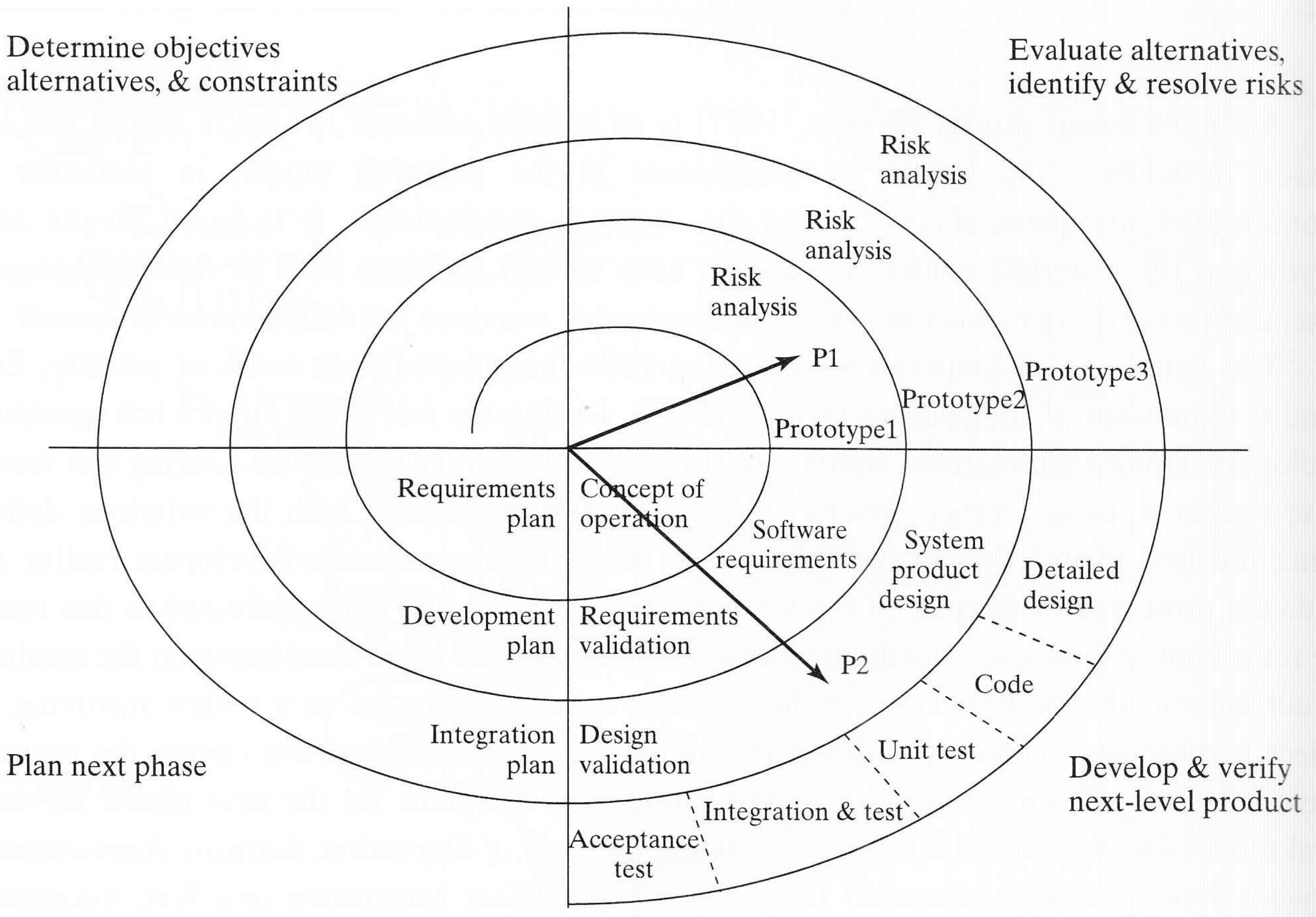
- Iterative: successive approximations to the right solution
- Incremental: work on portions of the system at a time
- Combination: permits portions to be revised as needed

- Modern process models are generally incremental and iterative
- Such models are sometimes called “evolutionary”

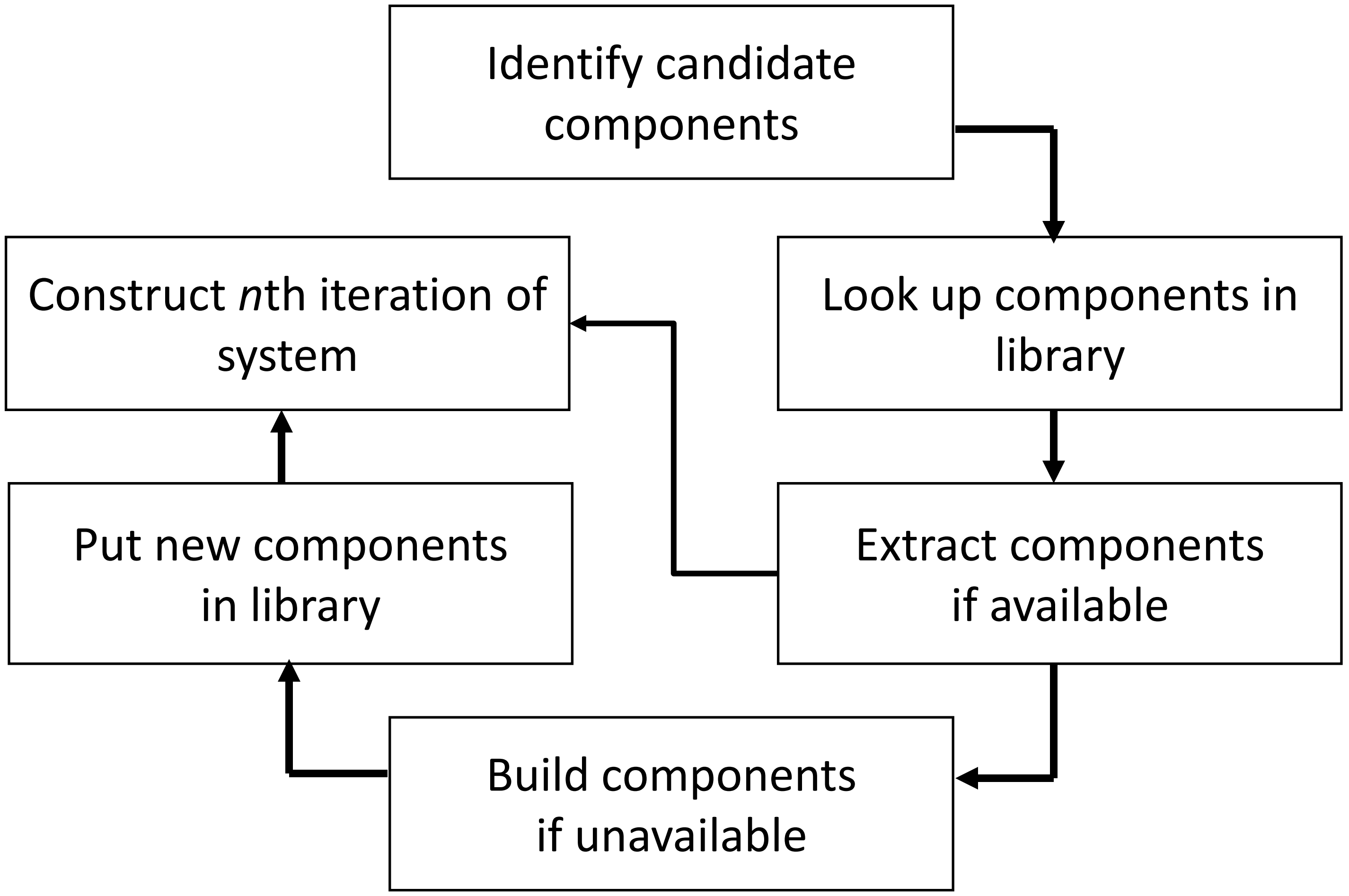
Spiral model

- Focuses on addressing risks incrementally
- Same activities as Waterfall, plus risk management, reuse, and prototyping
- Each spiral (called a "round") is composed of four phases:
  - exploring alternatives, defining constraints, identifying objectives (e.g. improve quality, extend functionality)
  - manage risks associated with these solutions
  - realize and validate the system
  - plan next round

Spiral model



The component assembly model


Identify candidate components
Look up components in library
Extract components if available
Build components if unavailable
Put new components in library
Construct *n*th iteration of system

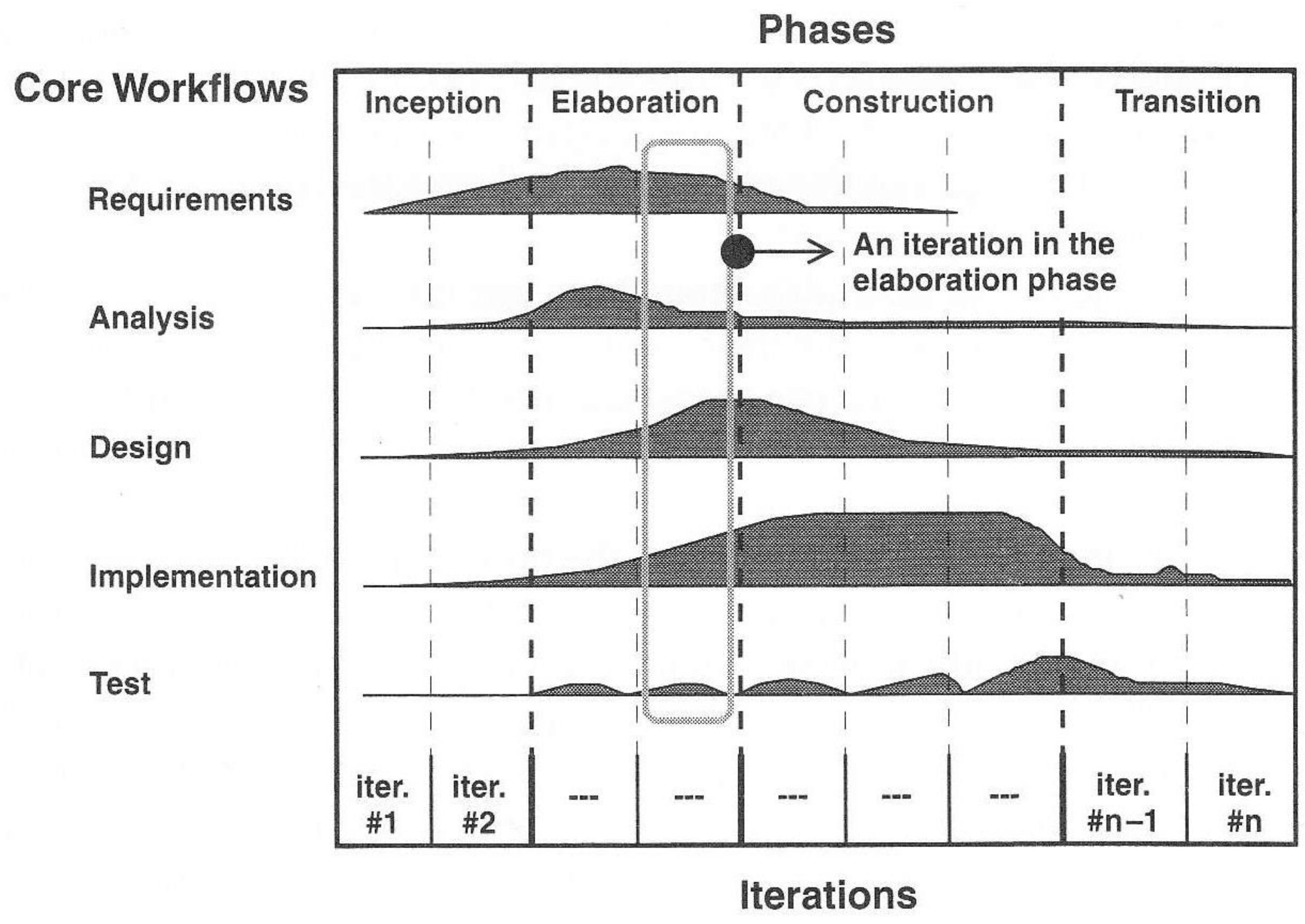
Component assembly model

- Object technologies provide technical basis
- Software reuse
  - reliability, cost reduction
- Problems:
  - Finding components
  - Are components reusable?
  - Adaptation (software systems aren't made from Lego blocks)

(Rational) Unified Process (RUP)

- Extends the ideas of the spiral model
- Distinguishes "cycles" in the lifetime of a system
  - essentially, level of maturity:
    - inception
    - elaboration
    - construction
    - transition

RUP


Phases
Core Workflows
Inception
Elaboration
Construction
Transition
Requirements
An iteration in the elaboration phase
Analysis
Design
Implementation
Test
iter. #1
iter. #2

---
iter. #n−1
iter. #n
Iterations

Prototyping

- Throw together an experimental system, with no expectation that it will be good enough
  - allows major issues to be identified, risks reduced
- Difficult to plan and control
- Easy to assume that the experiment covered all issues
- If experiment is successful, desire will be to not throw the prototype away
  - cost of fixing it versus re-developing it
- Note that prototyping is not a complete development process model

Build-and-fix

- More technically, “rapid application development”
- Developed in response to weaknesses of Waterfall processes
- An iterative approach, working from prototypes
- No focus on quality, leading to poor changeability in practice

Cowboy coding / hacking

- Developers do whatever they want whenever they want, without any input from others
- Lack of overall plan
- Utterly dependent on the individual’s skill and insight
- Agile methods do not encourage this approach
- Often is the practical approach taken to prototyping

Agile methods

- Frequent delivery of working software
  - by incrementally adding tested functionality
- Change is central
- Close communication and negotiability of everything (within team and with other stakeholders)
- Emphasis on individual skills
- Hybrids with plan-based approaches possible

Which is better?

- Predictive (plan-driven) approaches
  - high criticality
  - junior developers
  - stable requirements
  - large team
  - culture demands order
- Adaptive (“agile”) approaches
  - low criticality
  - senior developers
  - volatile requirements
  - small team
  - culture is supportive of negotiation

What is risk?

“That’s risky behaviour.”

“There’s a risk of snow later today.”

“The risk is unacceptably high.”

- Probability (of a negative event) seems to be involved
- Which is worse?
  - A likely event with mild consequences
  - An unlikely event with dire consequences

Risk

Often referred to as a "threat"

- risk(*event*) = cost(*event*) x probability(*event*)
- "Cost" could be:
  - The loss of life
  - The loss of money required to pay for recovery
  - The loss of revenue
  - The loss of market share
  - The loss of social status/esteem
  - The loss of career
- Obviously, some of these costs are difficult to quantify, and to compare

Risk management

- Identification
- Assessment
- Treatment
- Contingency planning

- Balance
  - Too little analysis and planning: nasty surprises
  - Too much: poor use of resources, leading to additional threats

Identification

- Two points to start from:
  - Source analysis
    - Where can threats come from?
      - Stakeholders, employees, management, technology, government, natural disasters, …
  - Problem analysis
    - What specific threats can we foresee?
      - For the industry, organization, team, project, individual, …
    - What impact might those threats have?
      - Regulation, liabilities, resources, technological costs

Assessment

- Estimation of likelihood
- Estimation of impact

- Sometimes easy
- Sometimes complete guesses (are they useful then?)

Treatment

- Avoidance or elimination
  - Do something so the threat should not happen
  - The cost of this might outweigh the risk
- Reduction or mitigation
  - Do something so that the risk is decreased
  - Same drawback as above
- Retention
  - Do nothing up front; cope if it actually happens
  - This should be a conscious decision!!!
- Transfer
  - Some sort of insurance policy (similar to reduction)

Contingency plans

- Threats that you treat via retention or reduction may happen
- Consider, ahead of time, what to do in such a situation
  - This can often reduce the total cost (e.g., speeding up recovery, avoiding catastrophe)
- Again, it’s about balance

Risk management plan

- At organizational and project levels, this can be a good thing to write down explicitly
- At an individual level, it’s a good thing to at least think about
- During a crisis, you won’t have time to carefully consider what to do
  - Scrambling towards a solution may make the situation worse

Kinds of software development threats

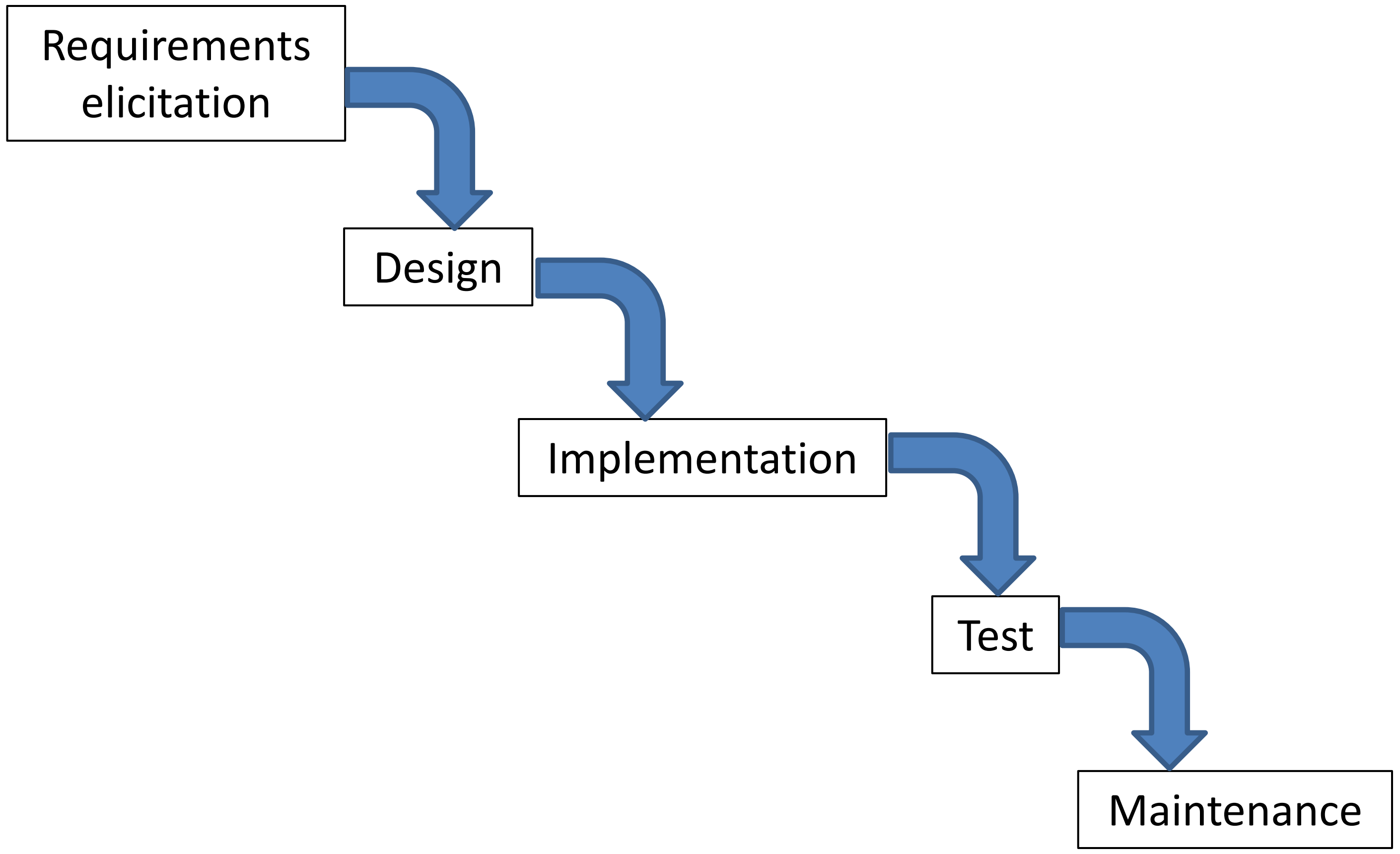
- Managerial
  - Over-schedule/over-budget
  - Exit of key personnel
  - Entry of new personnel
  - Dependencies on other projects out of one's control
  - Other management decisions
  - Uncertainty

Kinds of software development threats

- Technical
  - Poor design choices
  - Poor performance
  - Novel technologies
  - Poorly understood requirements
  - Errors
  - Usability problems
  - Change

Threats and process

- Consider the Waterfall model



Threats and process

- The Waterfall model:
  - Avoid technical problems by enforcing completion of one development phase before starting the next
  - Estimate schedules and costs
  - Document everything
  - Rigidly control change
- However:
  - Abstracts away the fact that these things may be impracticable
  - Assumes that scope is inflexible

Remember …

- Stakeholders
  - Customers
  - End-users
  - Managers
  - Development team

- Effective communication with the stakeholders is key to success

Goals of communication

- Explaining and understanding
  - what needs to be done
  - what is being done
  - what has already been done
- Social, managerial, and legal
  - what has been accomplished
  - who is responsible for what
  - what has each individual accomplished
  - why were certain decisions made

Forms of communication

- Informal verbal
  - hallway conversation, telephone call
- Formal verbal
  - presentation, meeting
- Informally written
  - email, electronic forum, scribbled note
- Formally written
  - letter, report, manual
  - source code, comments, README

Issues and tradeoffs

- Expense of creation
- Expense of reading or hearing
- Expense of modification
- Misunderstandings are likely
- Long distance communication
- Schedule coordination
- Preservation of information

formal
is worse than
informal,
written
is worse than
verbal

formal
is better than
informal,
written
is better than
verbal

Synchronous mechanisms

- When communication requires that sender and receiver must be available at the same time
  - hallway conversation, interview, meeting, chat software
- Useful for:
  - clarifying understanding, receiving summaries
- Not useful for:
  - preserving information, absorbing many details

Asynchronous mechanisms

- When communication DOES NOT require that sender and receiver must be available at the same time
  - reports, manuals, email, fax, discussion forum, message board, note, webpage
- Useful for:
  - disseminating details
- Not useful for:
  - ensuring that the "big picture" is understood
  - because people don't usually read

Documents

- Request for Proposal (RFP) document
- Requirements analysis document
- System design document, object design document
- Test plan document, test incident report document
- Software configuration management plan
- Software project management plan

- Installation manual, troubleshooting manual, FAQ, user manual
- Source code

How much communication?

- Balance risks of not communicating with risks of communicating too much
  - how serious would a misunderstanding be?
- The "sweet spot" is highly dependent on the nature of the project and the organization
- In general,
  - for small teams, informal communication is sufficient
  - for large teams, formal communication becomes increasingly important

What is a team?

- A group of people who work together in one unit for one purpose and one result, and who jointly follow a common guideline or process to achieve their common goal
- Key ideas:
  - one purpose, one process, one unit

What is a team?

- Members could be individual superstars, but if they don’t work by common principles, you don’t have a team!
- Members have a sense of collective ownership, and use the “we” word much more than the “I” word
- Members get rewards of success or failure as a group, not as individuals
- Team’s performance affects the images of members (positively or negatively)

What’s involved?

- Personalities
  - people are different; this ought to be a strength
- Politics
  - *poly*: many; *ticks*: blood-sucking insects
  - politics cannot be avoided in life, only coped with
- Motivation
  - the reasons for behaviours vary
  - when they conflict, politics comes into play
- Team life cycles
- Effective teams
- Teamwork inhibitors

Personality factors

- assertiveness
- introversion vs. extroversion
- internal vs. external sense of control
- anxiety
- motivation
- tolerance for ambiguity
- compulsive precision
- humility
- tolerance for stress

Personality types

- Falls into four dimensions, each of which involves a spectrum of characteristics
- Expressiveness/reservation
  - introvert versus extrovert
- Discovery
  - intuitive versus sensing
- Analysis
  - feeling versus thinking
- Decision-making
  - perceptive versus judging

Personality types

- Jung/Meyers-Briggs Personality Type Test
  - www.humanmetrics.com/cgi-win/JTypes1.htm
- Alike personalities tend to attract
  - good for finding and keeping friends
- Unalike personalities conflict
  - strengths and weaknesses are complementary
  - good for building a strong team
- Conclusion:
  - a group of friends often makes for a weak team
  - conflicts must be managed

(Socio-)Politics

- Unavoidable, in any facet of life
- Everyone has a personal agenda (goals, dreams, …)
  - if these mesh, great; if not, conflict

- Who’s involved?
- What’s the nature of the project?
- What’s the level of commitment of each participant?
- What’re the key issues that could lead to political fights?

- Can someone benefit from your project’s failure?
- Can someone benefit from your personal failure?
- Will someone perceive your success as detrimental?

Motivation

- Money goes only so far
  - not of equal importance to everyone
    - sufficient pay: money rarely is the issue
    - low pay: money is the only issue
  - dangerous:
    - perceptions of inequity on the team
- Job satisfaction
  - usefulness of the results, acknowledgment, fun
- Time off
- Perks

Team Life Cycle

- B. W. Tuckman’s model of team-development
- 4 phases
  - Forming
  - Storming
  - Norming
  - Performing
- This is not a linear progression, but teams actually go back and forth before they stabilize in higher-level phases

Teamwork inhibitors

- Defensive management
  - no autonomy to the team; team feels it's not trusted
- Bureaucracy
  - perceived as waste of time
- Physical separation
  - casual interaction is important; separation from other teams enhances group perception
- Fragmentation
  - doing too many other things
- Quality reduction of the product
  - self-esteem and enjoyment are reduced
- Phony deadlines
  - sense of manipulation, lack of respect

Effective teams

- Clear goals
- Clear roles
- Internal respect
- Open communication
- Consistent and fair leadership
- Sense of ownership, responsibility, accomplishment
- Avoid putting detrimental people on team
  - isolate them if unavoidable
  - encourage them to do something else
  - you’re not a therapist, so be careful

Death March projects

- A death march project is one whose “project parameters” exceed the norm by at least 50% through:
  - Schedule compression
  - Staff compression
  - Budget compression
  - Functionality inflation
- Alternatively, the probability of failure is >50%

Why should you care?

- Death March projects seem to be commonplace, and maybe even the norm
- They lead to constant overwork for prolonged periods
  - serious health impacts
  - serious personal life impacts
- Unless the high probability of failure is acknowledged up-front, **<u>you</u>** may be blamed for the failure

Why are we discussing this here?

- Not everyone in the course will go on to SENG 403
- This is practical advice on how to deal with real world situations that you may very likely encounter

Why do Death Marches happen?

- Politics
- Naïve/devious promises made by marketing, senior executives, inexperienced project manager, …
- Naïve optimism of youth
- The “Marine Corps” mentality
- Intense competition from new markets/new technologies
- Intense pressure caused by unexpected regulations
- Unexpected or unplanned crises

Why do people participate in Death Marches?

- The alternative is unemployment
- Required to be considered for future advancement
- The alternative is bankruptcy or similar calamity
- An opportunity to escape the “normal” bureaucracy
- Revenge
- The naivete and optimism of youth
- Risks are high, but so are rewards
- The “Mt. Everest” syndrome
- The “buzz” of working intensely with other committed people

Typical responses to Death Marches

- For team members:
  - depression
  - exhaustion
- For management:
  - denial
  - accusations
  - poorly considered "fixes"
    - Putting more people onto a project is a guarantee to make it even later and more likely to fail
  - demands for grueling workloads

Kinds of Death Marches

- "Mission impossible"
  - high risk, but doable
  - everyone dreams of fame, glory, and riches from the success
- Ugly
  - manager expects that team members will be sacrificed to achieve (his) success
  - individual problems attributed to character flaws
- Suicide
  - everyone miserable; project failure almost certain
  - alternative is to be fired
- "Kamikaze"
  - a glorious death: opportunity to work with cutting-edge technology, etc.

What can you do?

- Be aware of whether the project has the hallmarks of a Death March
- If you are a volunteer, enjoy
  - You have no one to blame but yourself!
- If you are a draftee:
  - refuse to participate (may mean losing your job)
  - do your best to survive
    - do the things that will be examined
    - protect your teammates
    - lie to your superiors if you must
  - or negotiate

Negotiation

- If (upper) management/customer is sometimes somewhat flexible, negotiate to minimize damage
  - extra time off for the entire team
  - assignment to sabbatical-like projects for extended periods
  - flexibility in the scope (80/20 rule)
  - flexibility in the deadlines (if it's one day late, will the company be bankrupt?)
- Unwillingness to discuss alternative scenarios can reflect badly on management
  - can be used to deflect blame when the project fails

Negotiation

- Avoid the trap of the "instant estimate"
- Always give your estimates with confidence levels (plus-or-minus ranges)
  - when they are repeated back to you without the confidence levels, doggedly add them back in
- Threaten to give your services to the competition, especially once the project is well-started
- Resign for real (there are probably better jobs out there)