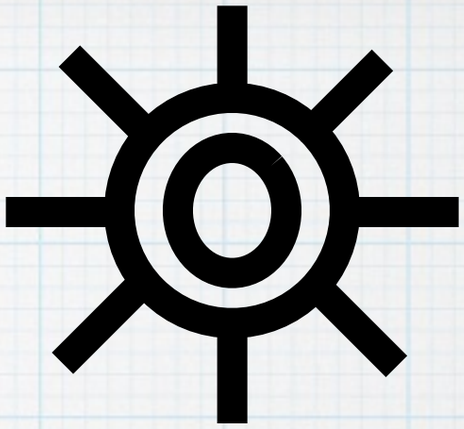
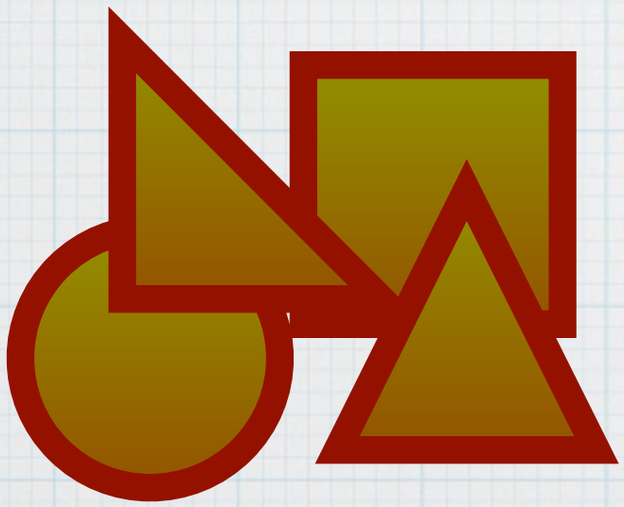


Requirements Engineering
Process

Les Waguespack, Ph.D.



Slides Four

Copyright and References

* Requirements Engineering, Kotonya & Sommerville, Wiley, Chichester, West Sussex, England, ISBN 0-471-97208-8
* Software Requirements Engineering, Second Edition, Richard H. Thayer and Merlin Dorfman, eds., pp. 7-22. Los Alamitos, Calif.: IEEE Computer Society Press, 1997.
* Use Case Modeling, Bittner & Spence, Addison-Wesley / Pearson Education, Inc., Boston, MA, ISBN 0-201-70913-9
* Writing Effective Use Cases, Cockburn, Addison-Wesley, Boston, MA, ISBN 0-201-70225-8
* UML and the Unified Process - Practical Object-Oriented Analysis and Design, Arlow & Neustadt, Addison-Wesley / Pearson Education, Inc., Boston, MA, ISBN 0-201-77060-1
* Business Modeling With UML, Eriksson & Penker, Wiley, Indianapolis, IN, ISBN 0-471-29551-5
* UML 2 Toolkit, Eriksson, Penker, Lyons & Fado, Wiley, Indianapolis, IN, ISBN 0-471-46361-2
* Enterprise Modeling With UML Designing Successful Software Through Business Analysis, Addison-Wesley, Reading, MA, ISBN 0-201-43313-3
* Object Oriented Systems Engineering, Waguespack, course notes CS390, CS460, CS630, CS771, Computer Information Systems Department, Bentley College, Waltham, MA.

Outline

1. Requirements Engineering Project Management Tools
2. Requirement Analysis
3. Negotiation
4. Validation
5. Managing Requirements

1. RE Project Management Tools

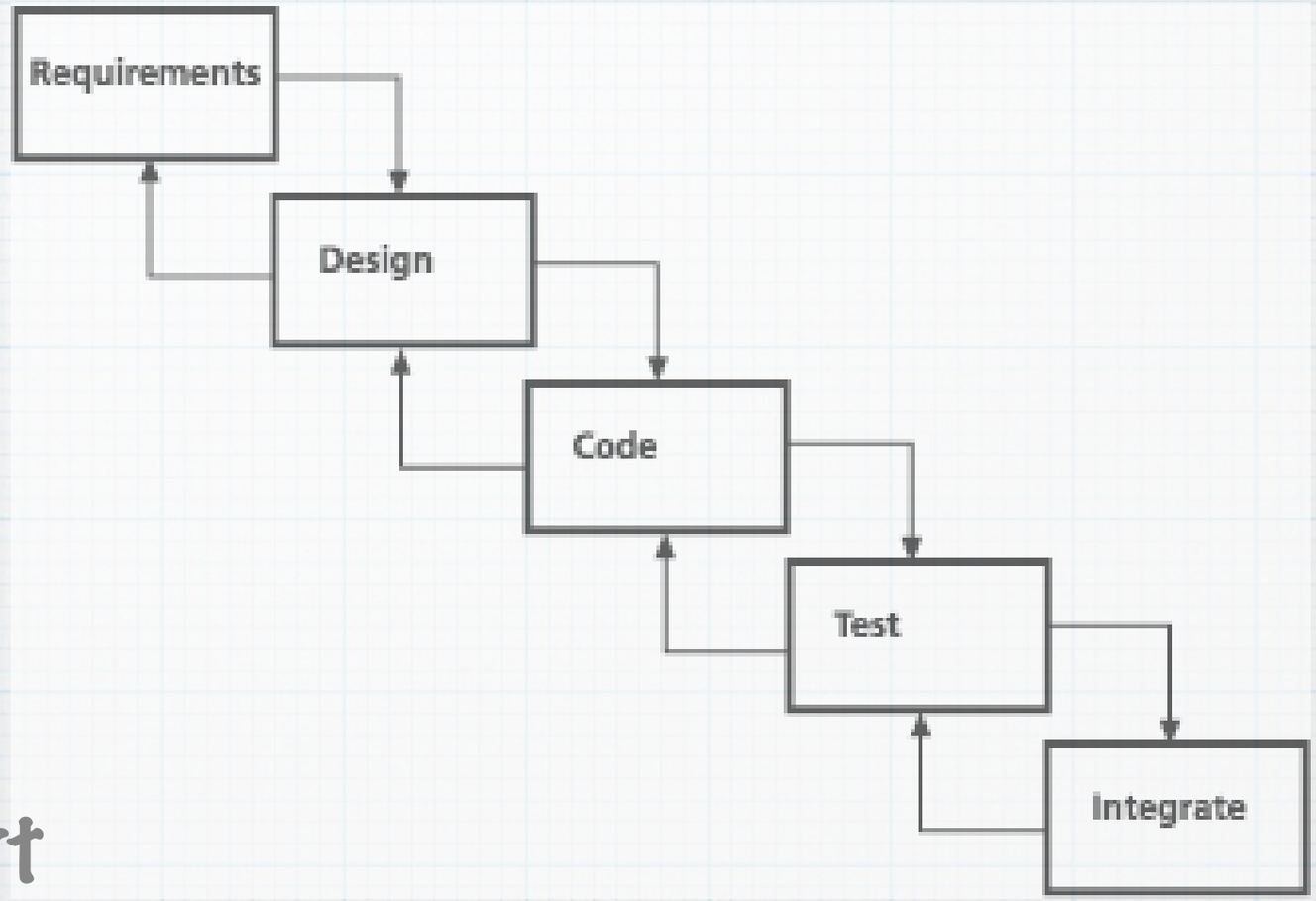
* project life cycle integration
* methodological
* technological

Project Life Cycle

* Baseline / Waterfall
* Prototyping
  * Incremental
  * Evolutionary
* Spiral
* Agile

Baseline / Waterfall

* Relies on primarily static requirement environment



* Vulnerable to environmental, technological, or policy evolution

* Most useful in short time frame situations

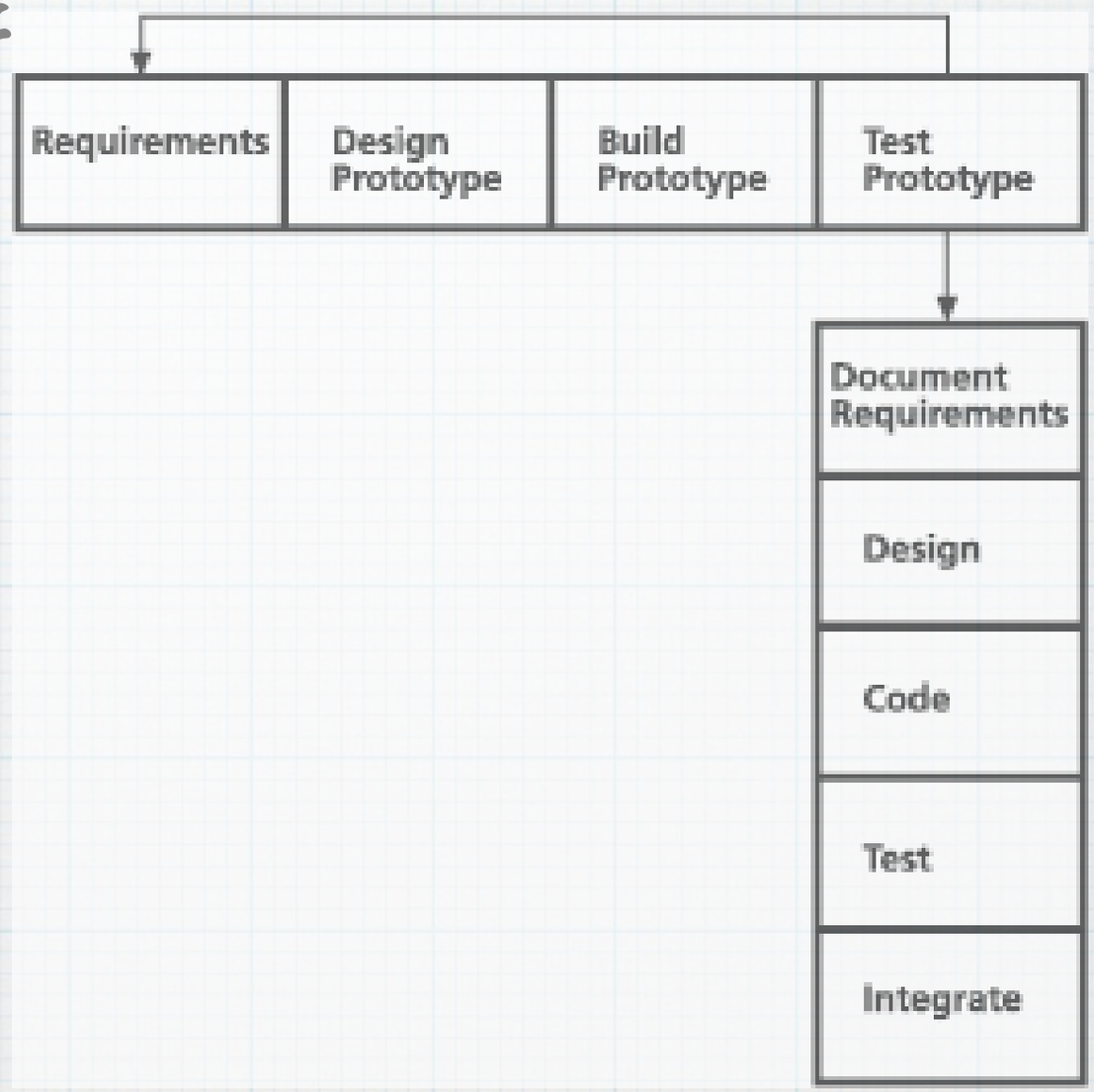
Thayer & Dorfman 1997

Prototyping

* Assumes "softness" of user expressed requirements

* Uncertainty, "risk" aversion intent

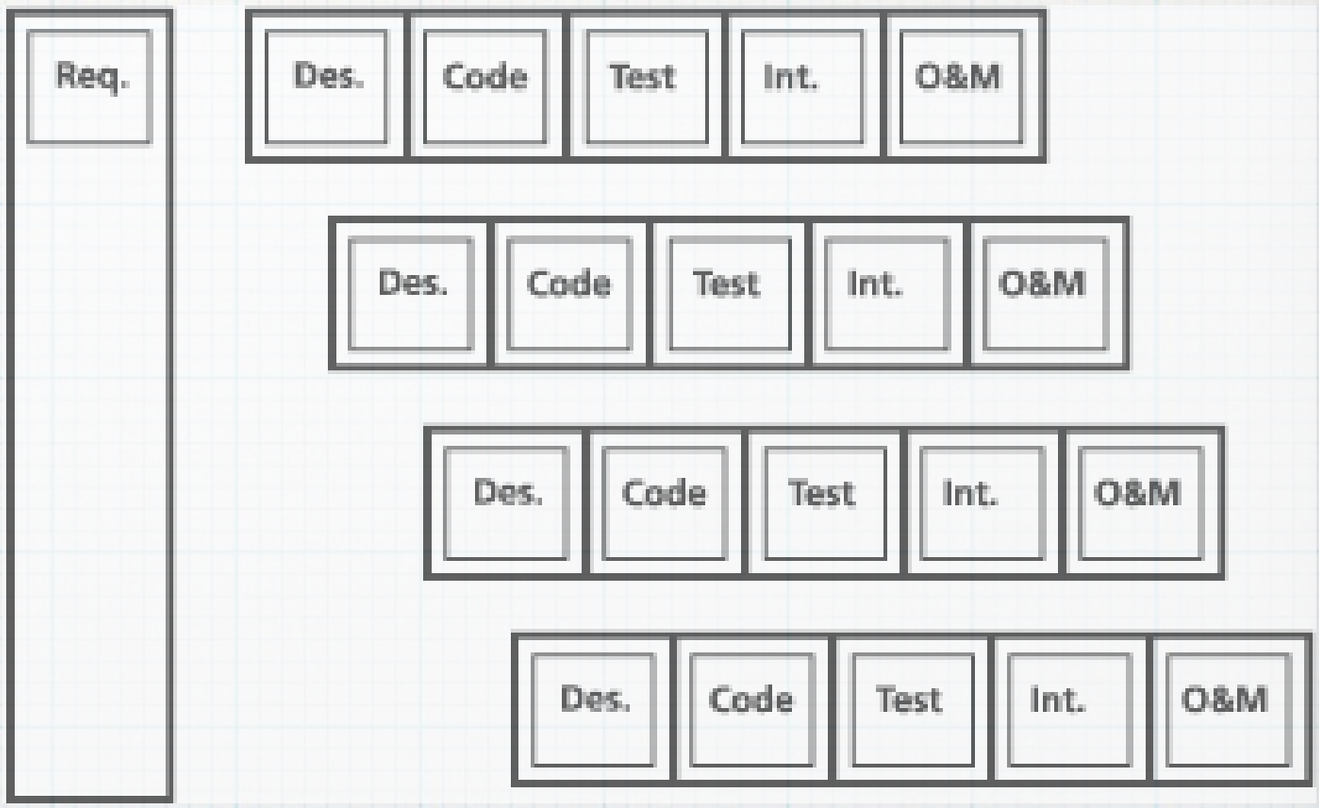
* Exploratory, Experimental, Evolutionary



Thayer & Dorfman 1997

Incremental

* Unitary requirements analysis allocated to a series of increments of system function
* Basically a "phased waterfall" approach
* Feedback from each increment informs the following phases



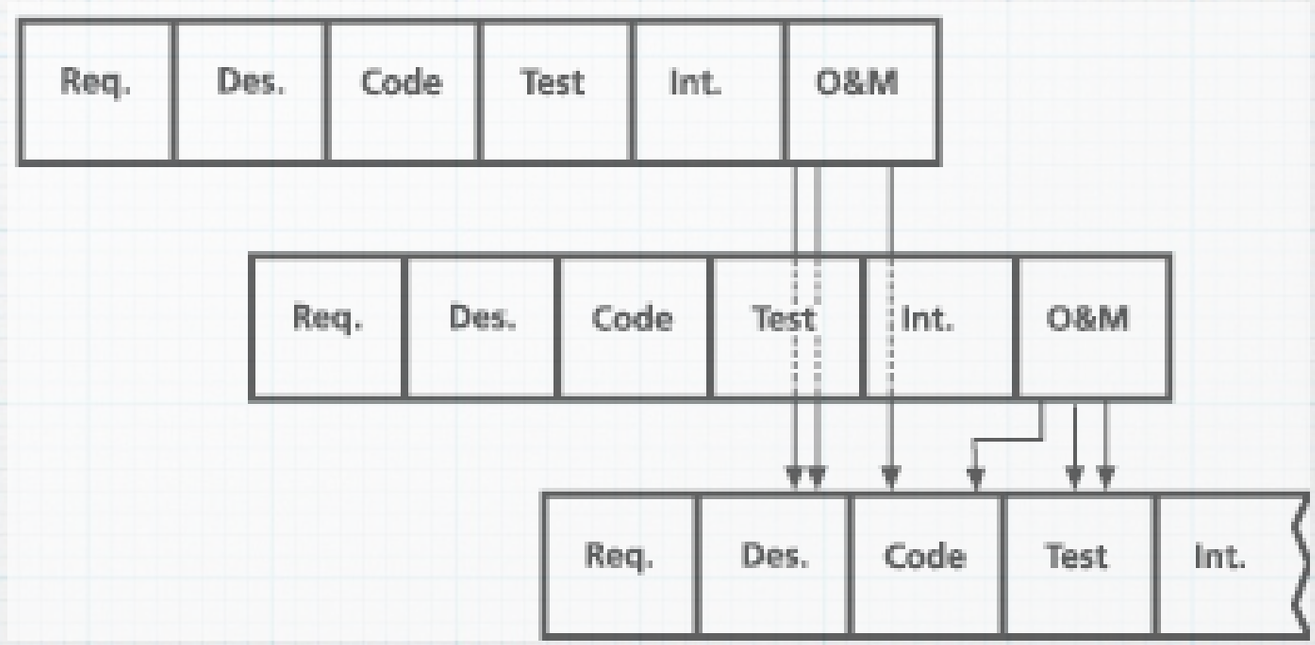
Thayer & Dorfman 1997

Evolutionary

* Like increments the prototypes address phases of the whole system development

* However, each increment is put into production

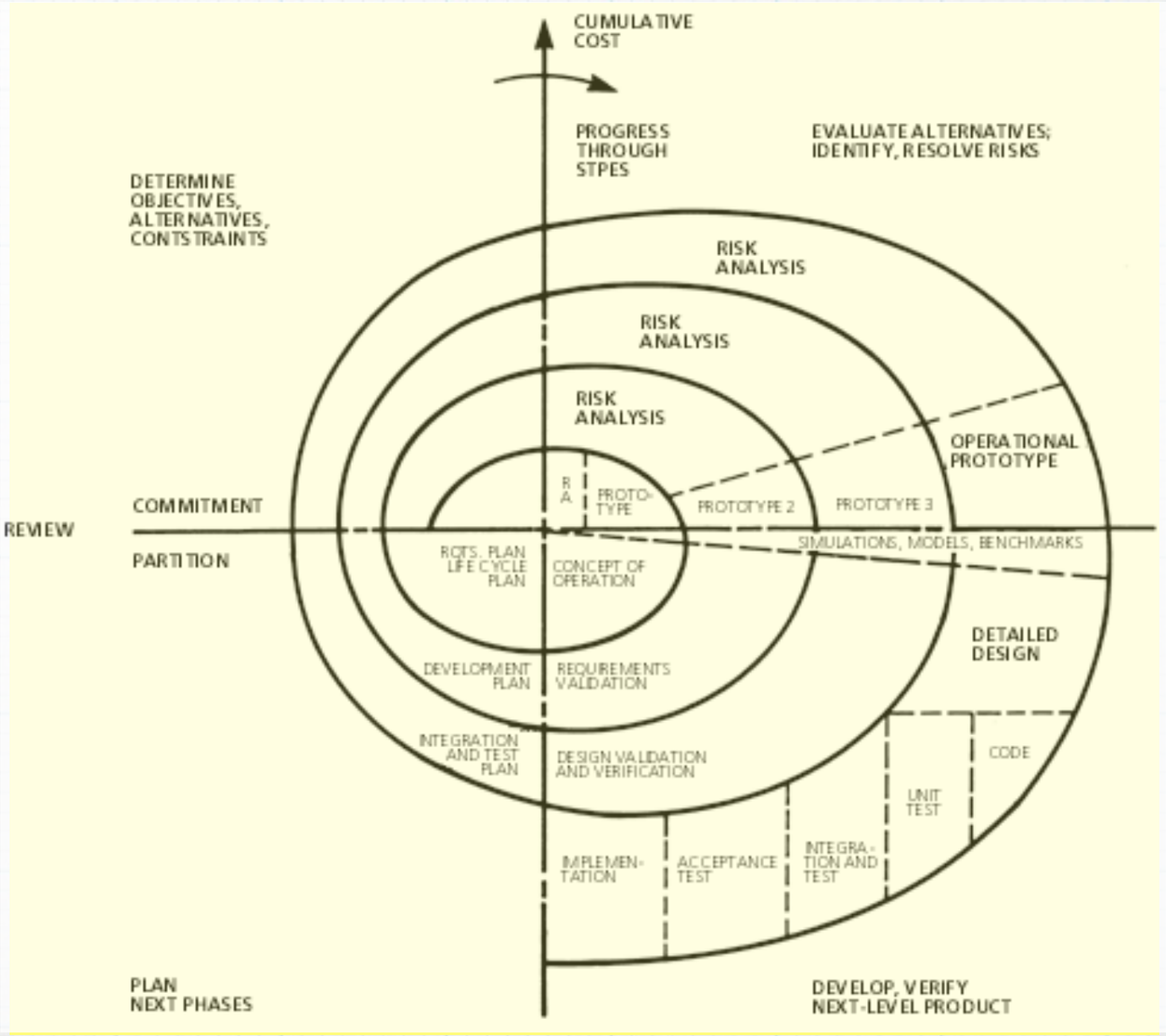
* Feedback follows extensive experience



Thayer & Dorfman 1997

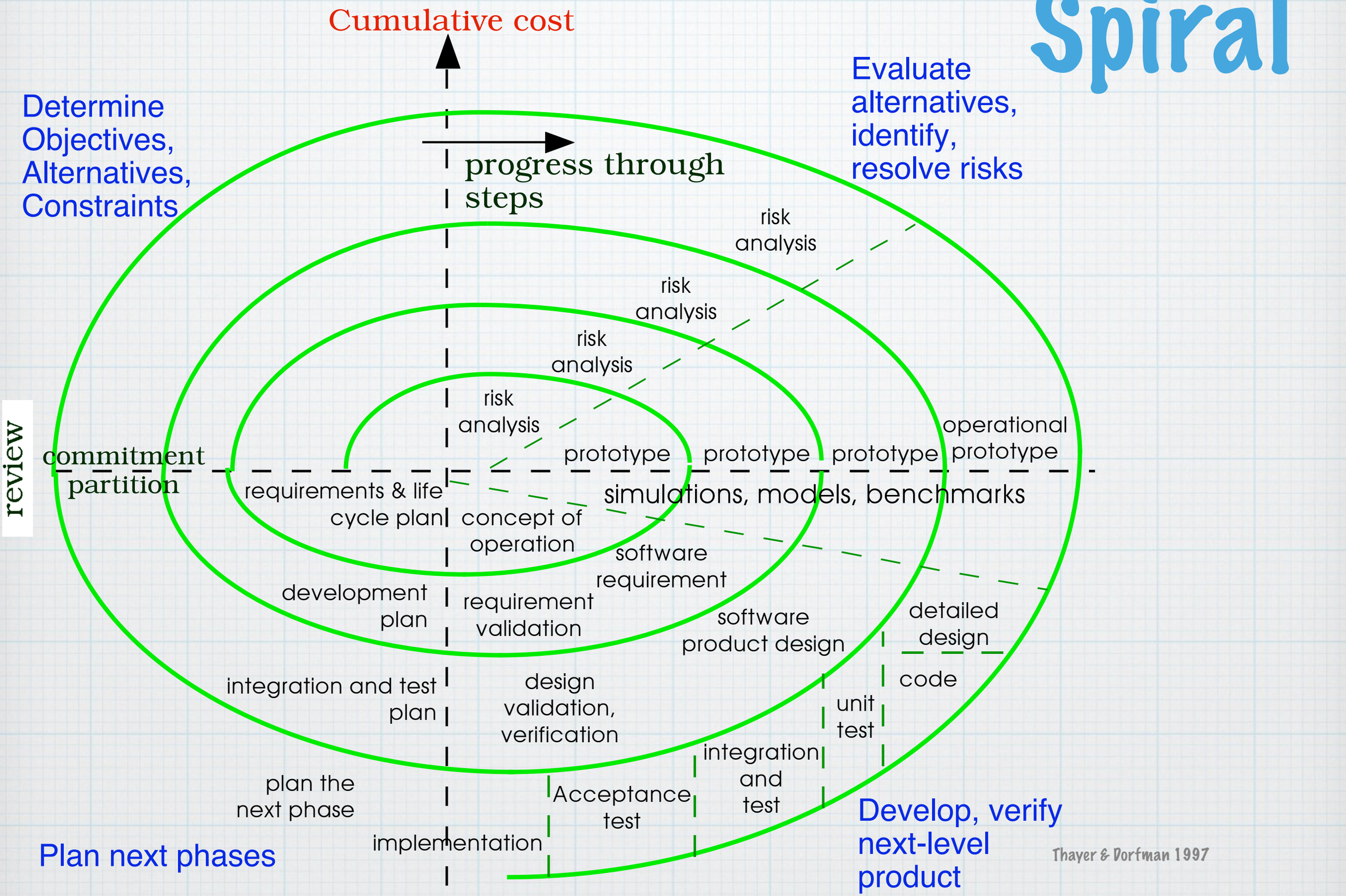
Spiral

* expands the scope of cycle focus to process decisions as well as product decisions
* focuses on risk analysis to guide process
  * revisits objectives, alternatives, constraints frequently
  * shapes subsequent cycle phases as part of the life cycle process
* It redefines the life cycle question
  * by subsuming the life cycle as a product in itself
  * allows other life cycle models to be special cases



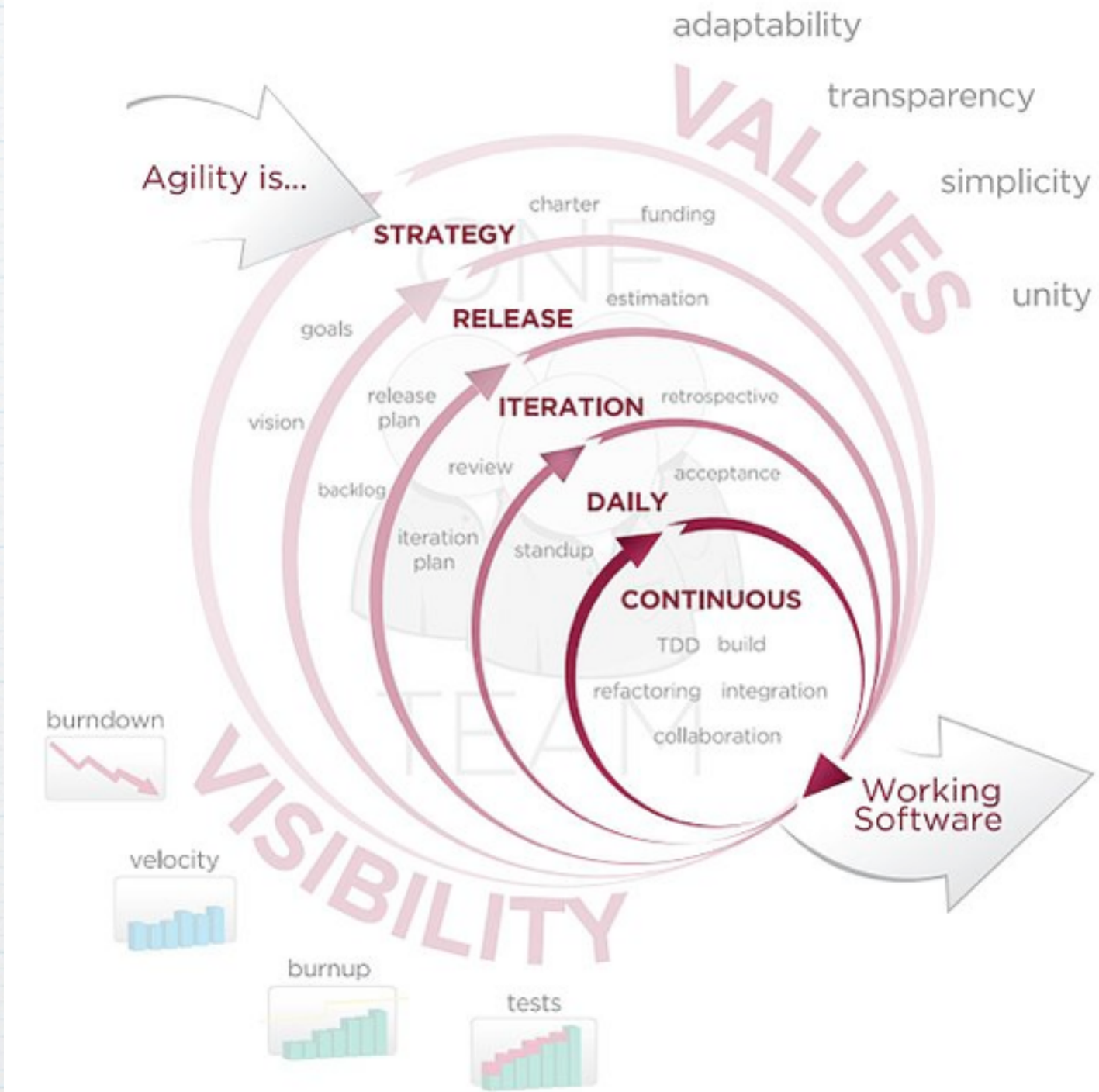
Thayer & Dorfman 1997

Spiral



Thayer & Dorfman 1997

AGILE DEVELOPMENT



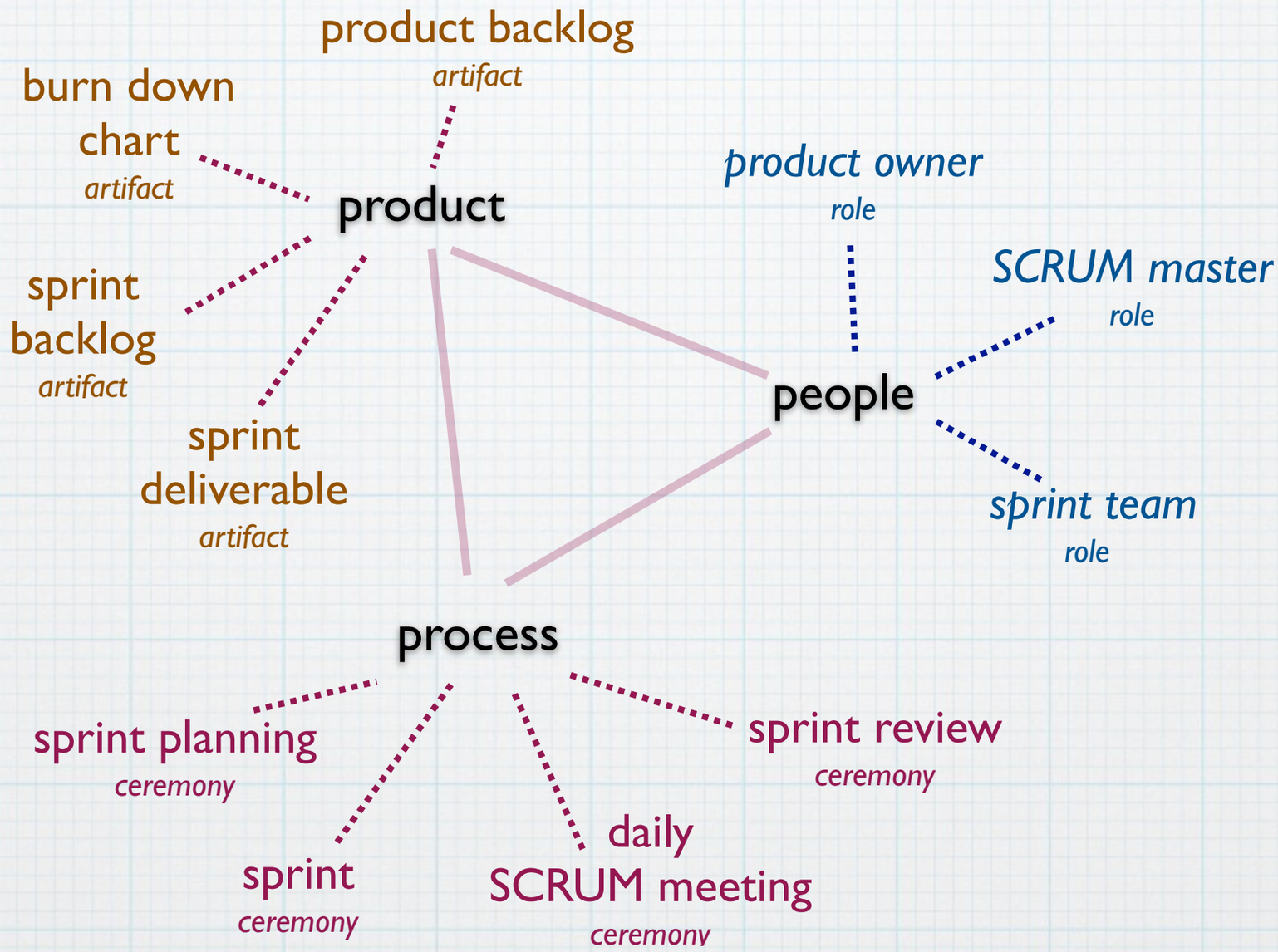
ACCELERATE DELIVERY

Agile Manifesto

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Working software is the principal measure of progress
5. Sustainable development, able to maintain a constant pace
6. Close, daily co-operation between business people and developers
7. Face-to-face conversation is the best form of communication (co-location)
8. Projects are built around motivated individuals, who should be trusted
9. Continuous attention to technical excellence and good design
10. Simplicity
11. Self-organizing teams
12. Regular adaptation to changing circumstances

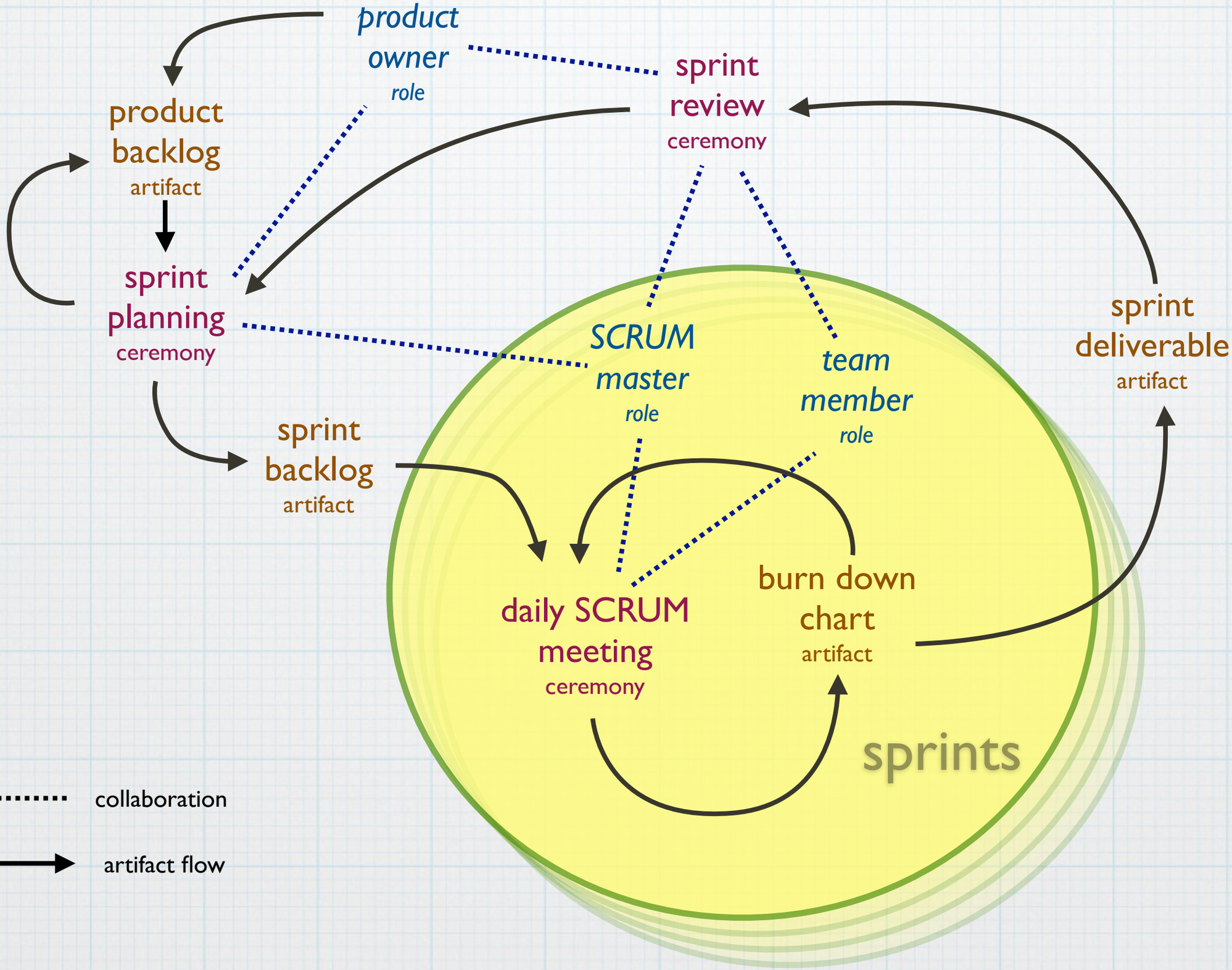
Beck, Kent; et al. (2001). "Principles behind the Agile Manifesto". Agile Alliance. Retrieved 6 June 2010.

SCRUM Ontology



Sutherland, J. and Schwaber, K., The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process, http://assets.SCRUMfoundation.com/ down- loads/2/SCRUMpapers.pdf?1285932052, Retrieved May 29, 2011.

SCRUM Architecture



* Methodology Support

* "decomposition-driven"
  * process oriented - Input/Process/Output -ex: Structured Analysis and Design (SADT), Vienna Development Methodology (VDM), "Z" (A formal specification model)
  * data oriented - ex: Jackson Systems Development (JSD), Entity Relationship (E-R)
  * control oriented - synchronization, deadlock, exclusion, concurrency, process activation/deactivation - ex: Real-Time SADT, Flowcharting
  * object-oriented - classes of objects, behavior, interaction - ex: Unified Process (UP)
* Agile methodologies:
  * SCRUM, Agile unified process (AUP), Dynamic Systems Development Method (DSDM), ... Extreme Programming (XP) ??

* Technology Support - (CASE)

* production technology
  * representation -
    * to enable the user to define, describe or change a definition or description of an object, relationship or process
  * analysis -
    * that enables the user to explore, simulate, or evaluate alternate representations or models of objects relationships or processes
  * transformation -
    * functionality that executes a significant planning or design task, thereby replacing or substituting for a human designer/planner
* coordination technology
  * control
    * functionality that enables the user to plan for and enforce rules, policies or priorities that will govern or restrict the activities of team members during the planning or design process
  * cooperative functionality
    * enables the user to exchange information with another individual(s) for the purpose of influencing (affecting) the concept, process or product of the requirements team

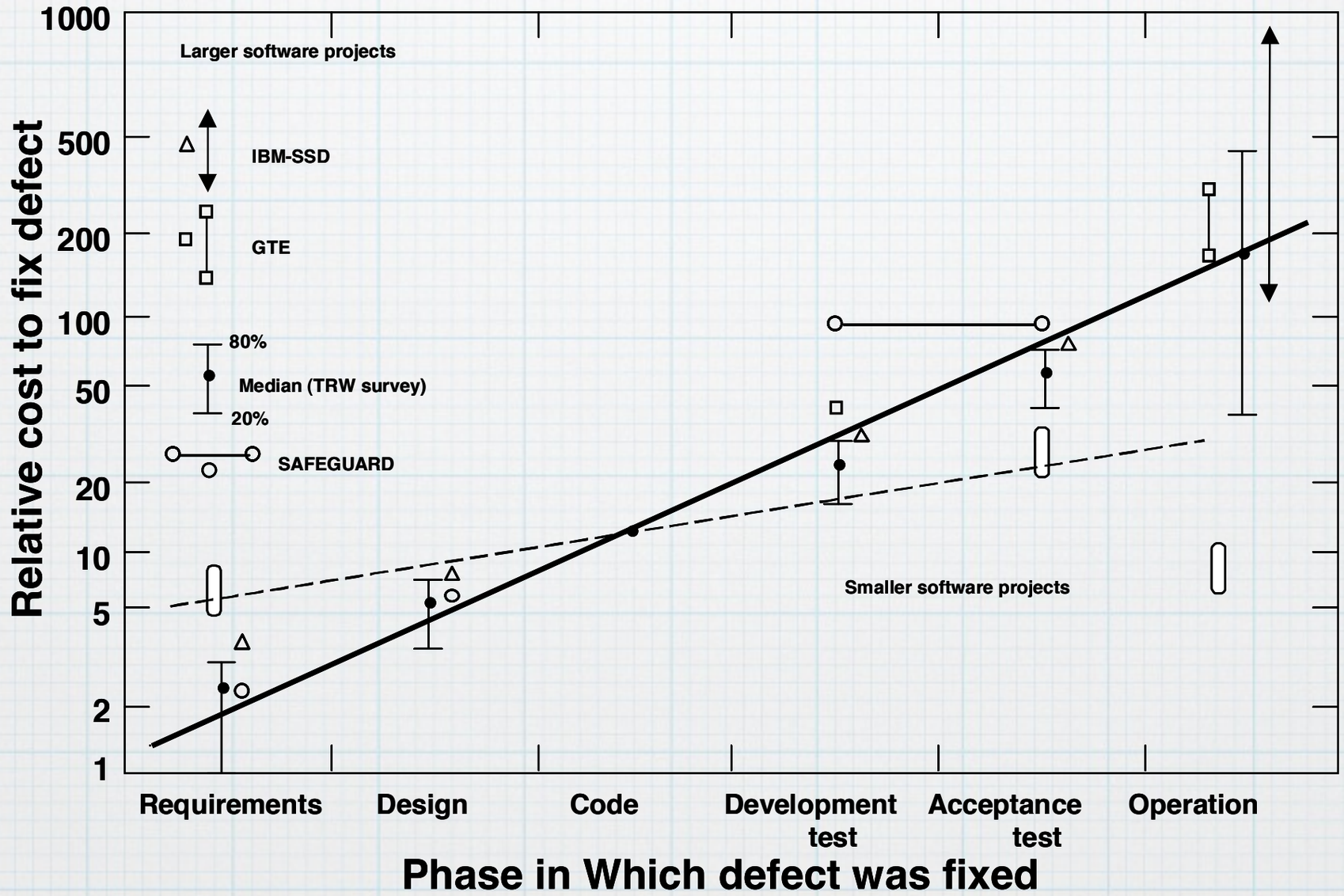
Requirements Process Improvement

* Improve what?
  * quality, time to market, cost
* Opportunity identification -
  * current process problems, improvement goals, process changes, process control
* Typical obstacles -
  * stakeholder involvement, missed business needs, management discipline, vague responsibilities, weak communication
* Process approaches -
  * Six Sigma, Capability Maturity Model (CMM)

2. Requirement Analysis

* "Have we got the right requirements?"
  * Early identification of anomalies, inconsistencies, or ambiguities is critical
  * The longer a deficiency survives in the system development time line - the more it costs to fix
  * Budget (time, cost, personnel) estimation depends on reliable work definition (requirements)

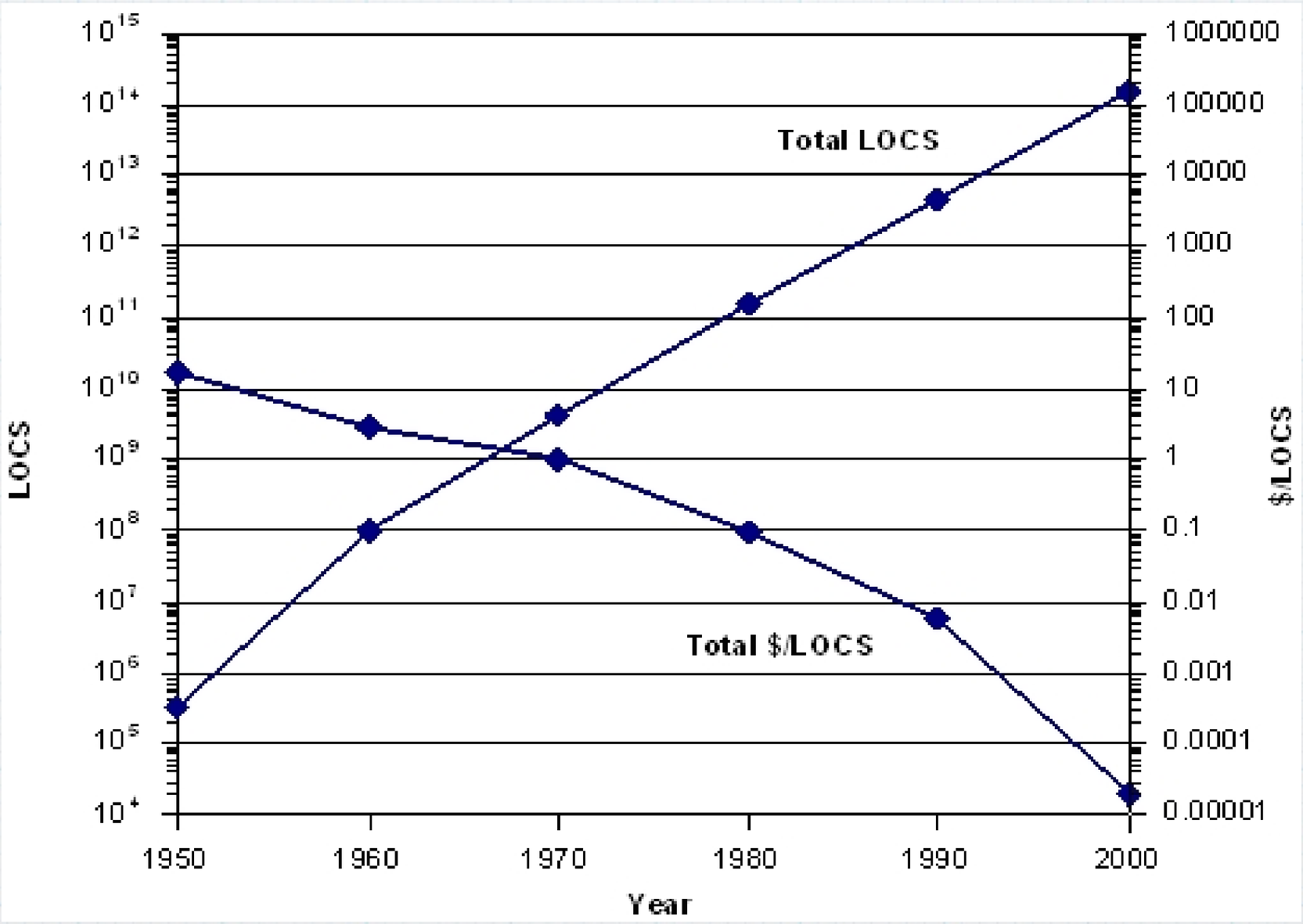
Cost to Fix



Boehm, B., Software engineering. IEEE Trans. Computers, 100(25):1226-1241, 1976.

Where the cost lies . . .

LOCS -Lines of code in service



U.S. DoD Lines of Code in Service and Cost/LOCS

Boehm, B., A Spiral Model of Software Development and Enhancement, *Computer*, May 1988, pp. 61-72.

Requirement Relationships

* checklists are useful in normalizing lists of requirements

* interaction matrices uncover -
  * overlaps - may indicate redundancy
  * conflicts - may indicate inconsistency

	R1	R2	R3	R4	R5	R6	R7
R1							
R2	O			C			
R3						C	
R4		O					
R5			O				C
R6				O			
R7		O					

O - overlap C - conflict

"Ironing out the Wrinkles"

* Potential requirement deficiencies
  * premature design - confusing requirement with solution
  * multi-issue requirement - convolution
  * questionable necessity - "dream vs need?"
  * business goal / process inconsistency
  * ambiguity
  * reality check concrete testability

3. Negotiation

* Requirements analysis usually raises "wrinkles" to be ironed out
  * differences in stakeholder understanding / realization of the business model / process
  * differences in stakeholder held priorities
  * need for added clarity in requirement specification
  * need to revisit scope of project with client

Whose requirements are these?

* Until the client's authority "signs off" on the requirements document all you have is a "draft" that may be the client's requirements.
* The requirements document is an AGREEMENT that all parties understand and describes the same system.

4. Validation

* "Have we got the requirement right?"
  * review
  * prototype testing
  * model validation
  * requirements testing

5. Managing Requirements

* Convergence
* Stability Analysis
* Equilibrium
* Identification, Storage and Reuse
* Change management
* Traceability

Convergence

* Requirement elicitation and documentation is a process of discovery and refinement - a progressive approximation

* Change is inevitable due to policy, market, government, culture, etc.

Stability Analysis

* Changes occur for many reasons
  * errors, conflicts, inconsistencies
  * customer / user "epiphany"
  * technical, schedule, cost issues
  * customer priorities
  * environment, domain changes
  * organizational changes

Equilibrium

* The volume and rate of change in the description can indicate requirements in flux which require additional attention
* Descriptions that maintain limited change can be said to be in "equilibrium"
* Project experience can be used to set these stability thresholds

Identification, Storage and Reuse

* **Document**: "If it's not recorded, it doesn't exist!"
* **Catalog**: "If you can't find it, it doesn't exist!"
* **Index**: "If it's too much work to look for it, it doesn't exist!"
* **Cross-Reference**: "If you don't know what it relates to, you won't think to look for it!"

* Typical Requirements Tracking Data
  * identification
  * description / explanation
  * entry date
  * change history
  * change source
  * reason / rationale for change
  * status: proposed, under review, accepted, rejected
  * precedent and antecedent requirements / changes
  * analyst comments to the community
  * author

Change Management

* Requirements knowledge is a valuable asset
  * change can help its value accrue
  * haphazard change can erode its value
* As the requirement resource builds (matures) change should be treated with growing care and diligence retaining the whole stakeholder community's concurrence

* Each Change is its own Project
  * verify change request validity / authority
  * identify affected system components
  * draft changes due to coordinated dependencies
  * propose change specifics
  * accept / reject the change with clear documentation

Change Information Management

- the volume, complexity and volatility of evolving requirements information can tax the most well organized team
- repository tools can mean the difference between a well structured resource and a "house of cards"
- repository tools will also include "practice" standards for the team and stakeholders to normalize the quality across the board (more to come)

Traceability

* What is supposed to be done?
* Who told us to do it?
* When did we know we would do it?
* Why did we choose to (or not to) do it?
* What other things are affected by it?
* How will we know these things in the future?

All is Cost/Benefit

* An unstructured collection of documents, contacts, interviews, requirements specifications, change requests, change decisions and supporting commentary quickly becomes a "haystack" - virtually unsearchable.

* The database, indexing, cross-references, and model diagrams all contribute visibility and connectivity to the requirement resource.

* The value of the requirement resource (size, longevity, quality, user community, problem complexity, post-implementation customer commitment) influences the investment decision in the automation and stewardship of the resource.

Wrap Up

1. Requirements Engineering Project Management Tools
    1. project life cycle integration
    2. methodological
    3. technological
2. Requirement Analysis
3. Negotiation
4. Validation
5. Managing Requirements
    1. Convergence
    2. Stability Analysis
    3. Equilibrium
    4. Identification, Storage and Reuse
    5. Change management
    6. Traceability