REQUIREMENTS ENGINEERING

CSC 510
North Carolina State University

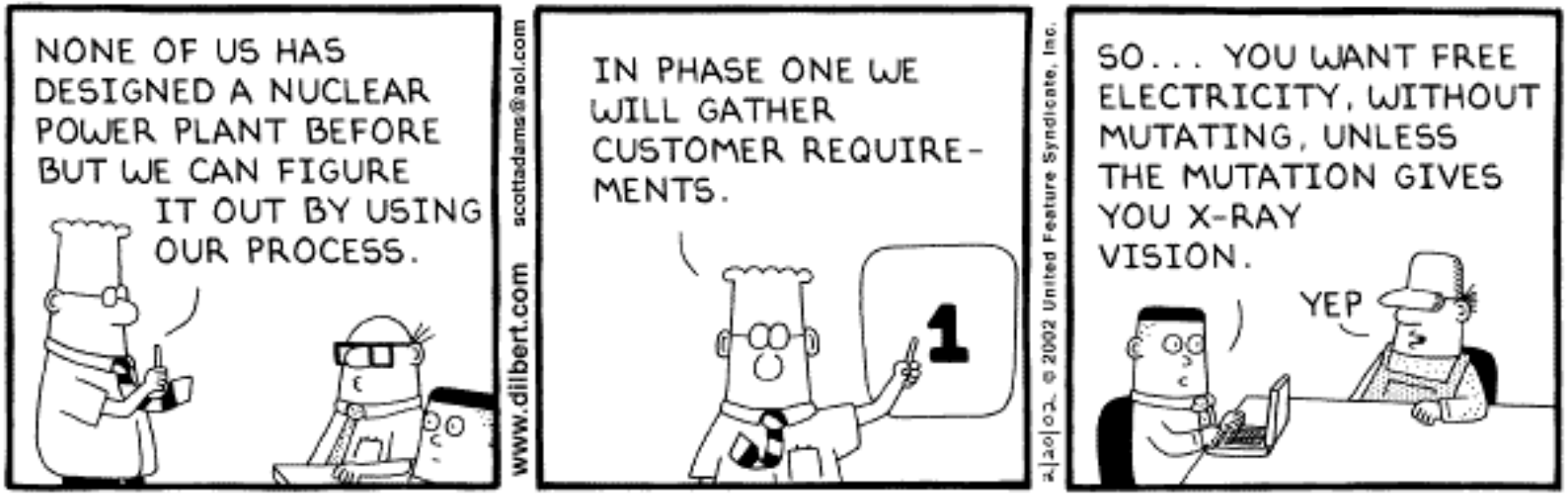
tim.menzies@gmail.com

March, 2016

Requirements Engineering Defined

"The development and use of cost-effective technology for the elicitation, specification and analysis of the stakeholder requirements which are to be met by software intensive systems."

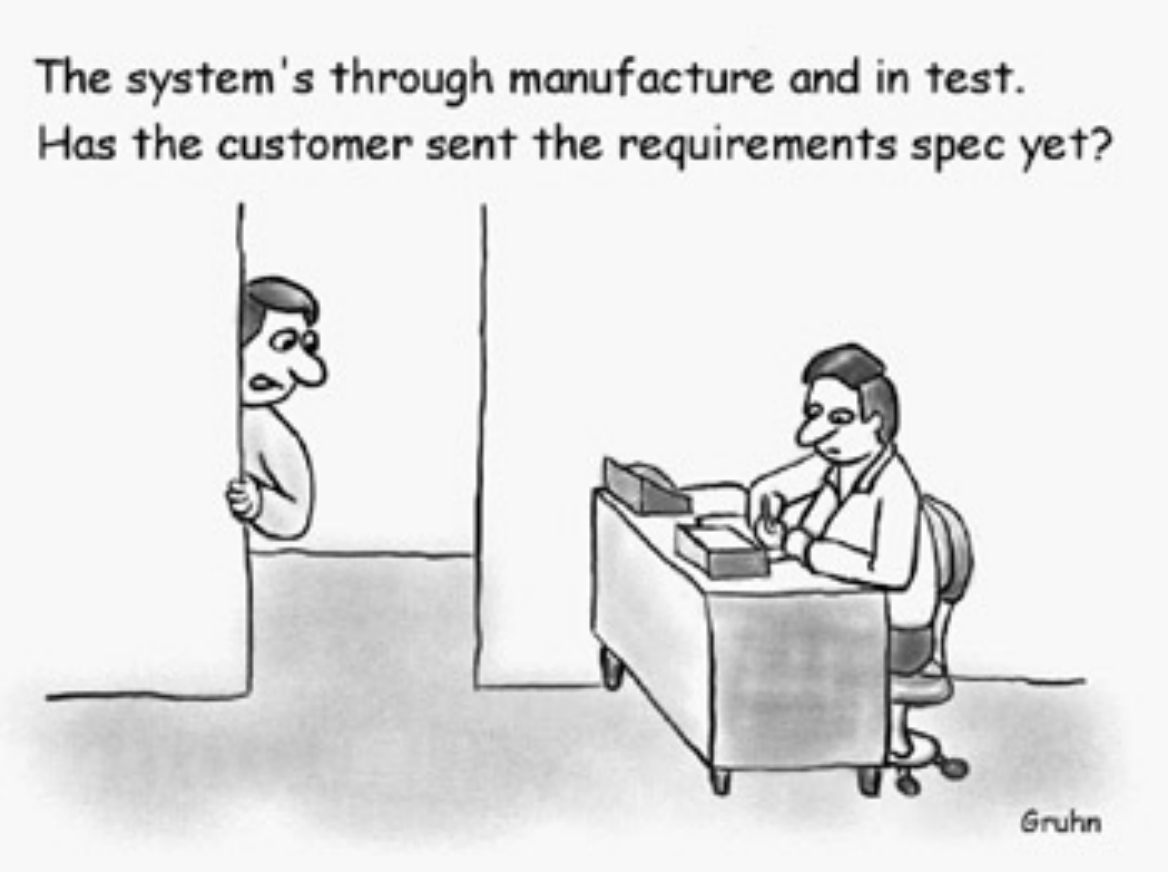
[RENOIR]



REQUIREMENTS: BAD IDEA?

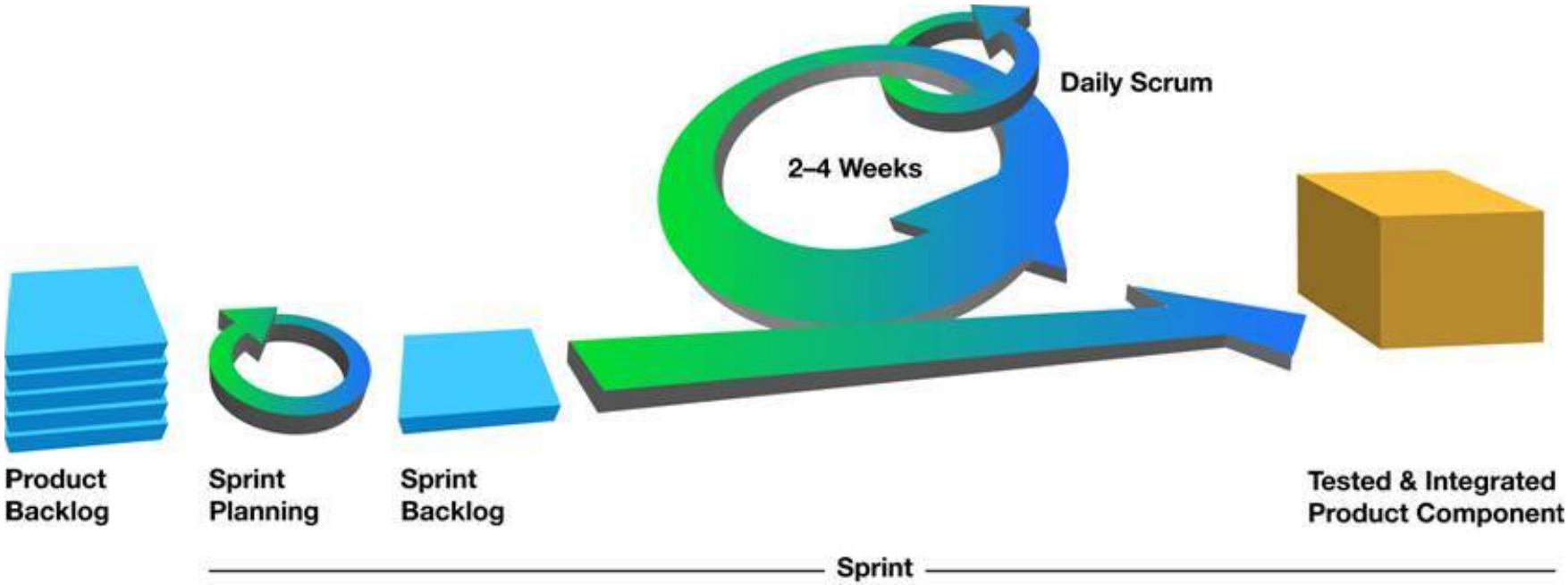
The dissenting view

Beware “analysis paralysis”



Fans of agile say…

- Experience tells you more than excessive initial analysis



Assessing any of the following in a rigorous manner is an open research issue

- Completeness
- Comprehensibility
- Testability
- Consistency
- Unambiguity
- Writeability
- Modifiability
- Implementability

As to the real value of requirements….

- Ideally, we look before we leap
  - Find best ways to do proceed
  - And we don't get it wrong
- In reality, our initial peeks are wrong
  - Need extensive modification
  - If you want God to laugh, tell her your plans

- Recent research says requirements are an illusion
  - Cognitive phenomena including anchoring bias, fixation and confirmation bias
  - Misrepresenting decisions as requirements in situations where no real requirements are evident.
  - Misrepresenting an incidental feature as a requirement will reduce exploration of the design space, curtailing innovation
- Nevertheless, sometime in your career,
  - you will be asked "how do we start?"
- Welcome to the black art of requirements engineering

GETTING IT WRONG

Examples to make you tremble

Must have mobility access ramps

Must have external staircase

The Geneis crash landing

- NASA sample return probe
  - collected a sample of solar wind
  - returned it to Earth for analysis
  - Then crash landed in Utah in 2004
  - Landing chute did not deploy
- Design error
  - Acceleration installed backwards

- Cost
  - $164 million for spacecraft development and science instruments
  - $45 million for operations and science data analysis.
- Mistake have been made worse by reuse
  - Same design in the (already launched) Stardust cometary sample return craft
  - Stardust landed successfully in 2006.

HMS Sheffield Exocet Missile Mis-classification	May 4, 1982	30 deaths 257 lives at risk £23,200,000	•failure to intercept incoming missile •ship loss	•inadequate requirements •unnecessary simplicity
Y2K (*)	Jan 1, 2000	$300,000,000,000	•large-scale effort •widespread problems	•short-sighted temporal requirements •premature optimization •arithmetic overflow
Virtual Case File	2000-2005	$170,000,000	entire program scrapped	•shifting requirements •scope creep •mismanagement
Vancouver Stock Exchange Rounding Error	1984	unspecified	index off by 50%(!)	•inadequate research •inadequate testing •rounding error
NASA Mariner 1.	1962		A booster went off course during launch, resulting in the destruction of	• Failure of a transcriber to notice an overbar in a written specification for the guidance program, resulting in the coding of an incorrect formula in its FORTRAN software

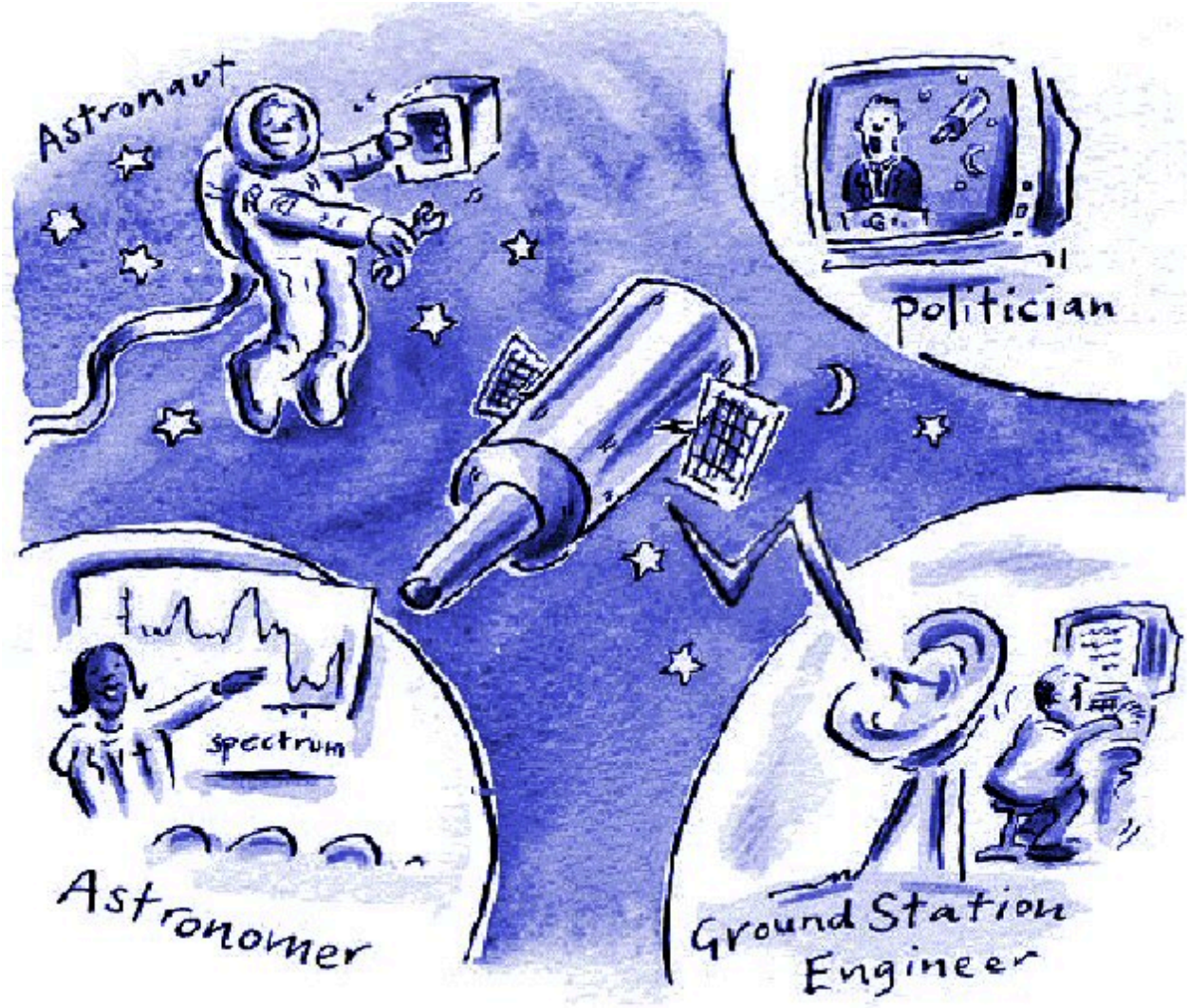
(*) A similar problem will occur in 2038 (the year 2038 problem), as many Unix-like systems calculate the time in seconds since 1 January 1970, and store this number as a 32-bit signed integer, for which the maximum possible value is $2^{31} - 1$ (2,147,483,647) seconds.[19]

Scope of Software Project Failures

WHY PROJECTS FAIL	%
1. Incomplete Requirements	13.1
2. Lack of user involvement	12.4
3. Lack of resources	10.6
4. Unrealistic Expectations	9.9
5. Lack of executive support	9.3
6. Changing requirements	8.7
7. Lack of planning	8.1
8. Didn't need it any longer	7.5
9. Lack of IT management	6.2
10. Technology illiteracy	4.3

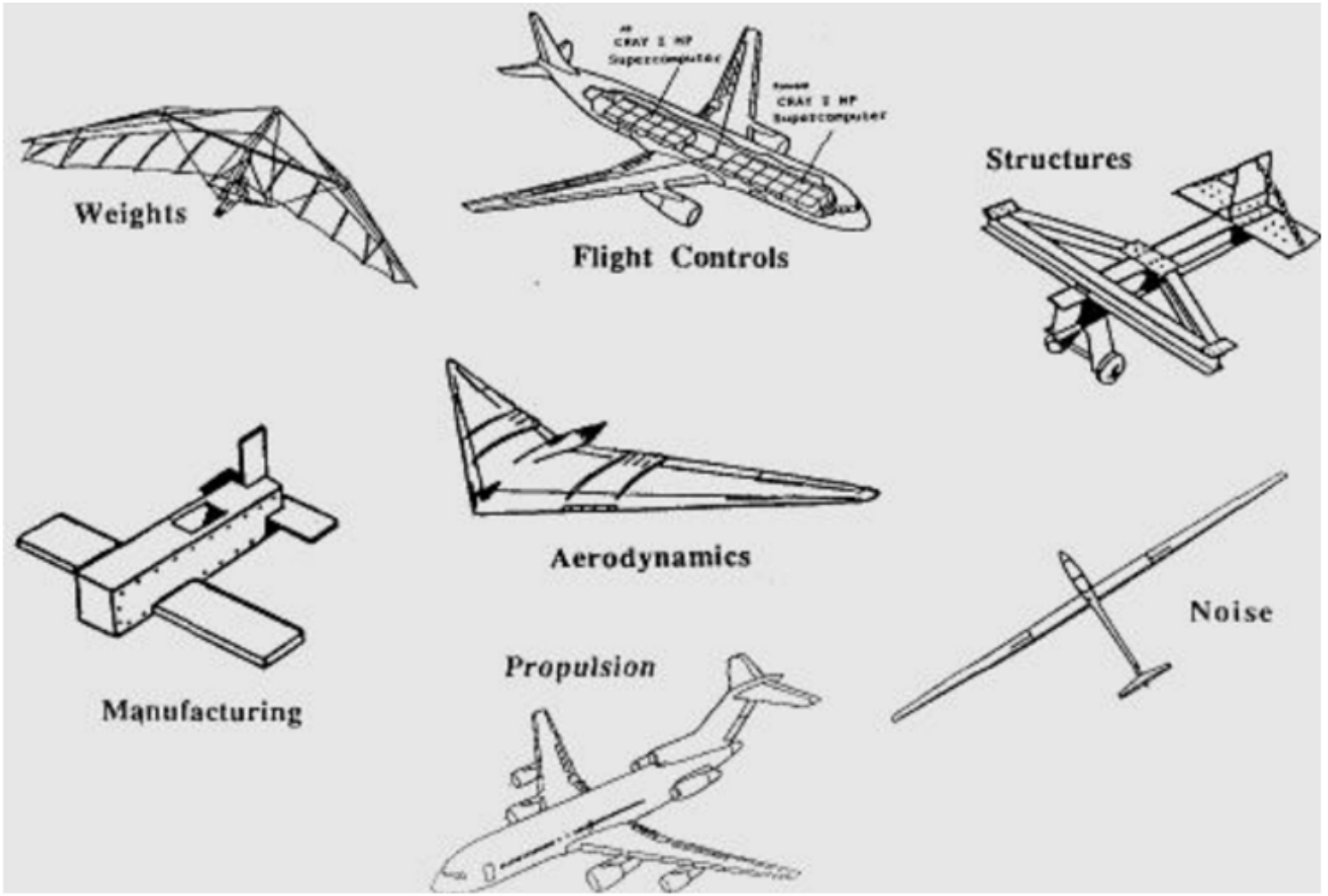
Jim Johnson, The Standish Group International Project Leadership Conference, May 1995, Chicago

Stakeholders:
Do all these folks want the same thing?

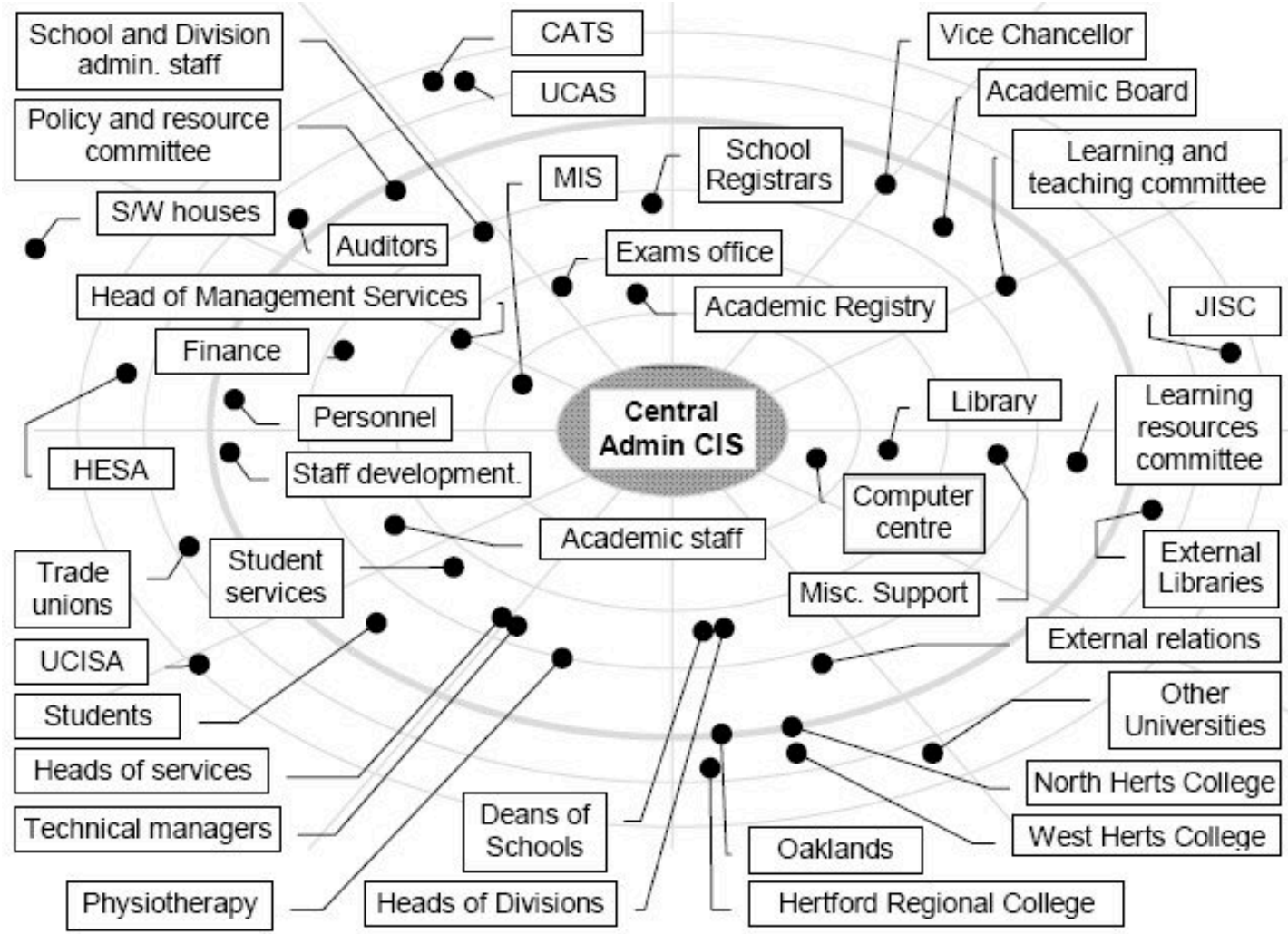


Stakeholders:

"one" thing is "many" things, depending on stakeholder perspective.



Stakeholders, examples:
Do all these folks want the same thing?



WRITING REQUIREMENTS

The Requirements Document

- The official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document
- As far as possible, it should set of WHAT the system should do rather than HOW it should do it
- Also, it should have tests that can be applied incrementally
  - See “commit partition”, below

Requirements Document Structure (1)

- **Introduction**
  - Describe need for the system and how it fits with business objectives
- **Glossary**
  - Define technical terms used
- **System models**
  - Define models showing system components and relationships
- **Functional requirements definition**
  - Describe the services to be provided
  - User stories go here
  - Add in notes for the commit partition here

Requirements Document Structure (2)

- **Non-functional requirements definition**
  - Define constraints on the system and the development process
  - Add in notes for the commit partition here
- **Constraints**
  - Add in notes for the commit partition here
- **System evolution**
  - Define fundamental assumptions on which the system is based and anticipated changes
- **Requirements specification**
  - Detailed specification of functional requirements
- **Appendices**
  - System hardware platform description
  - Database requirements (as an ER model perhaps)
- **Index**

Parts of a Req'ts Document

- Product Constraints
- Functional Requirements
- Non-functional Requirements
- Project Issues
- User Stories

Product Constraints

restrictions & limitations that apply to the product & problem

1. **Purpose of Product** - the reason for building it and the business advantage if we do so
2. **Stakeholders** - the people with an interest in the product
3. **Users** - the intended end-users, & how they affect the product's usability
4. **Requirements Constraints** - limitations on the project & restrictions on product's design
5. **Naming Conventions & Definitions** - vocabulary of the product
6. **Relevant Facts** - outside influences that make some difference to this product
7. **Assumptions** - that the developers are making

Constraints Are:

- global requirements that shape the requirements
- apply to the entire product
- The users of a product are a constraint since they dictate usability of the product:

> *Passengers on board an aircraft will use the product.*

- Constraints may also be placed on the eventual design and construction of the product:

> *The product shall run on the company's existing UNIX machines.*

Functional Requirements

the functionality of the product

8. Scope of the Product - defines the product boundaries and its connections to adjacent systems

9. Functional & Data Requirements - things the product must do and the data manipulated by the functions

Functional Requirements Are:

- Things the product must do
- An action that the product must take to provide functionality for its user
- Arise from the fundamental reason for the product's existence

> *The product shall produce an amended de-icing schedule when a change to a truck status means that previously scheduled work cannot be carried out as planned.*

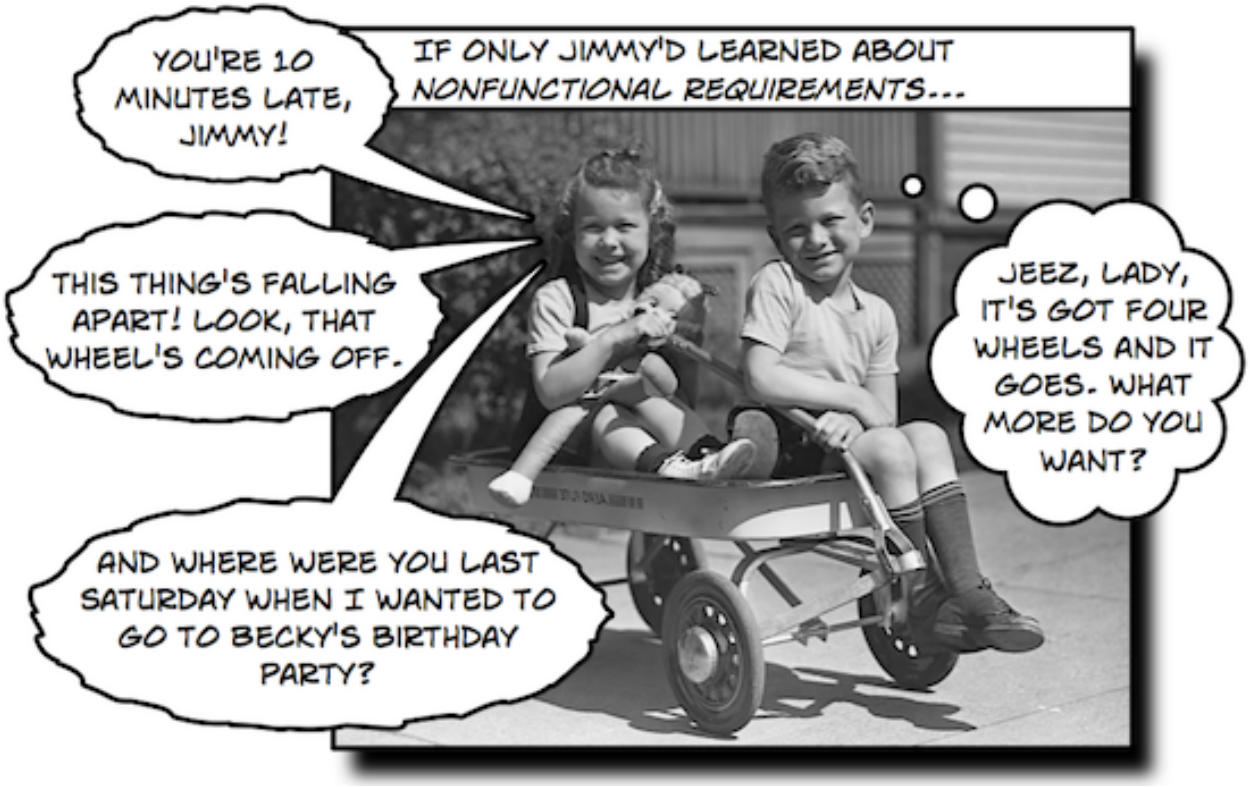
-> Something systems must do if it is to be useful within the context of the customer's business needs.

Non-Functional Requirements

the products qualities

10. **Look & Feel Reqt's** - the intended appearance
11. **Usability Reqt's** - based on the intended users
12. **Performance Reqt's** - how fast, big, accurate, safe reliable, etc.
13. **Operational Req'ts** - the product's intended operating envt.
14. **Maintainability & Portability Reqt's** - how changeable the product must be
15. **Security Reqt's** - the security, confidentiality & integrity of the product
16. **Cultural & Political Reqt's** - human factors
17. **Legal Reqt's** - conformance to applicable laws

Stakeholders have different "non-functional requirements"



Time
- Transactions / sec
- Response time
- Time to complete an operation

Space
- Main memory
- Auxiliary memory
- (Cache)

Usability
- Training time
- Number of choices
- Mouse clicks

Reliability
- Mean time to failure
- Downtime probability
- Failure rate
- Availability

Robustness
- Time to recovery
- % of incidents leading to catastrophic failures
- Odds data corruption after failure

Portability
- % of non-portable code
- Runs on N operating systems
- Runs on desktop, tablet, mobile

Etc

Non-Functional / Quality Requirements Are:

- properties, or qualities, that the product must have
- may be critical to the product's success

> *The product shall determine 'friend or foe' in less than 0.25 seconds.*

- some NFRs may simply enhance the product:

> *The product shall use company colors, standard company logos and standard company typefaces.*

- NFRs are usually attached to the product's functionality
  - E.g., how it will behave, qualities it is to have, how big or fast it should be

Project Issues

apply to the project that builds the product

18. **Open Issues** - as yet unresolved issues w/ a possible bearing on the product's success
19. **Off-the-Shelf Solutions** - components that may be used instead of building something
20. **New Problems** - caused by the introduction of new product
21. **Tasks** - things to be done to bring the product into production
22. **Cutover** - tasks to convert from existing systems
23. **Risks** - the risks the project is most likely to face
24. **Costs** - early estimates of cost or effort needed to build it
25. **User Documentation** - plan for building user documentation
26. **Waiting Room** - req'ts to be included in future releases

REVIEWING REQUIREMENTS

"Testing" things that do not yet execute

Actual log of topics addressed in requirements meeting

Not everything is a requirement

Non-functional requirements

Definition of user-level demos

Actual system requirements

Things talked about, but need not be in this system (maybe ideas for other development, another time?)

Is there more than one design?

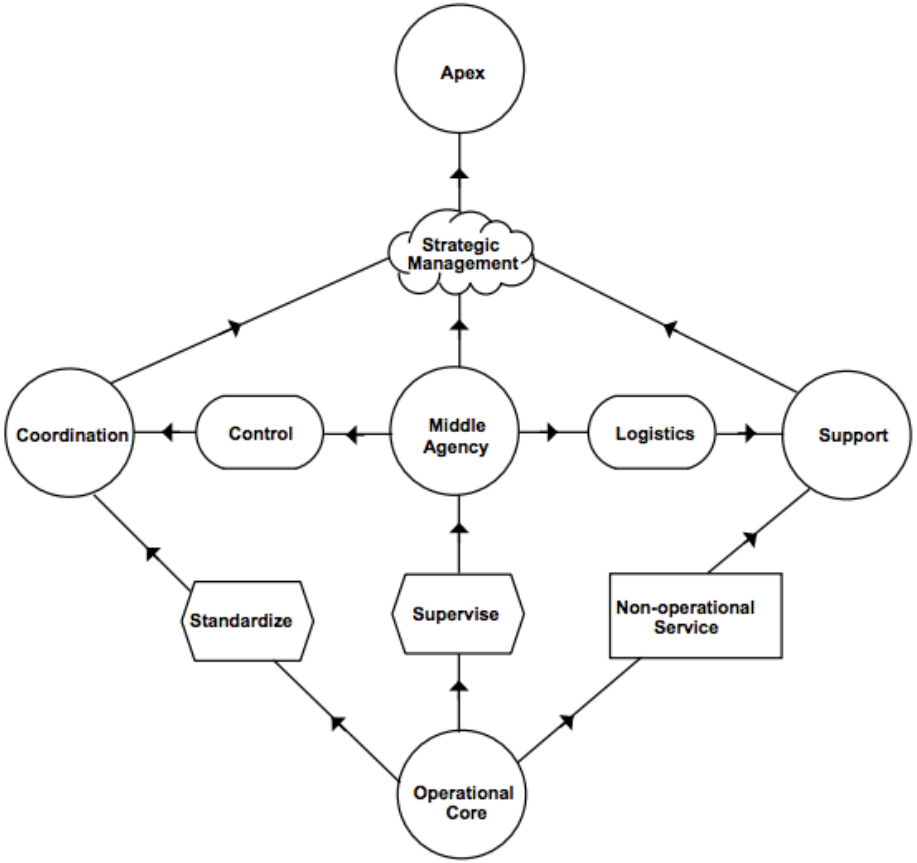


Fig. 6. Structure-in-5.

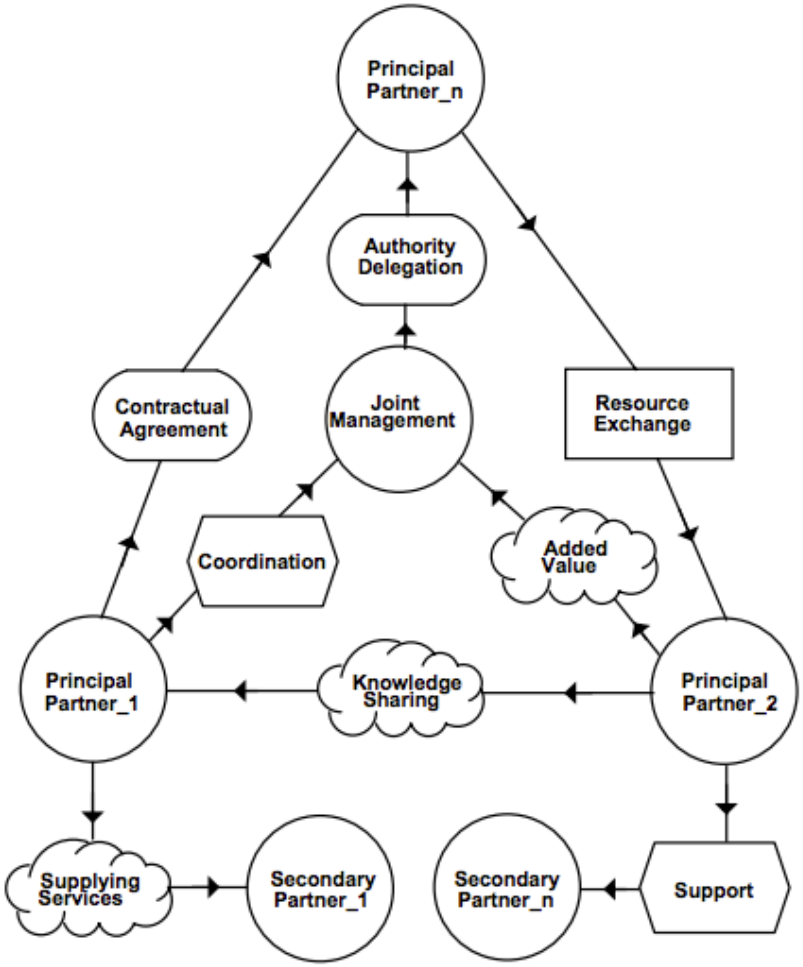


Fig. 7. Joint venture.

Is there more than one design?

- Assessing options of criteria
  - predictability (1), security (2), adaptability (3), coordinability (4), cooperativity (5), availability (6), integrity (7), modularity (8), or aggregability (9)

Table 1
Correlation catalogue

	1	2	3	4	5	6	7	8	9
Flat structure	− −	− −	−			+	+	++	−
Structure-in-5	+	+		+	−	+	++	++	++
Pyramid	++	++	+	++	−	+	− −	−	
Joint-venture	+	+	++	+	−	++		+	++
Bidding	− −	− −	++	−	++	−	− −	++	
Takeover	++	++	−	++	− −	+		+	+
Arm's-length	−	− −	+	−	++	− −	++	+	
Hierchical contracting			+	+	+	+		+	+
Vertical integration	+	+	− −	+	− −	+	− −	− −	− −
Cooperation	−	−	++	++	+	− −	−	− −	

- Which is best? Dunno. Ask your stakeholders!
  - But add value to their discussions
  - Give them considered conclusions to aid their decisions

Requirements Checking

- Validity
  - Does the system provide the functions which best support the customer's needs?
- Consistency
  - Are there any requirements conflicts?
- Completeness
  - Are all functions required by the customer included?
- Realism
  - Can the requirements be implemented given available budget and technology

Requirements 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?

Check for ambiguity in Stating Requirements

- Missing requirements
  - (e.g. structure, functions, physical env't.)
- Ambiguous words
  - E.g., *small* does not adequately specify the *size of the group*
  - *E.g., group* implies that the people will interact, but it's not clear *how*
- Introduced elements
  - E.g., *Structure* - "create a means" or "design a structure"

[Gause and Weinberg 1989]

Exploring to Remove Ambiguity (What explorers do)

- Move in some direction
- Look at what they find there
- Record what they find
- Analyze their findings in terms of where they want to be
- Use their analysis and recordings of what they find to choose the next direction
- Go back to step 1 and continue exploring

Identify the Requirements "Classes"

- Enduring requirements
  - Stable requirements derived from the core activity of the customer organization.
    - E.g. a hospital will always have doctors, nurses, etc.
  - May be derived from domain models
- Volatile requirements
  - Requirements which change during development or when the system is in use.
    - In a hospital, requirements derived from health-care policy