

Non-Functional Requirements

Lawrence Chung

Department of Computer Science

The University of Texas at Dallas

Non-Functional Requirements

Practices and Recommendations:
A Brief Synopsis

- Why
- What
- Some Classification Schemes
- NFRs and RE Processes
- Product-Oriented Approach: Some Individual NFRs
- The NFR Framework
- Appendix
 - With Rational Unified Process and UML
 - With Volere Requirements Specification Templates
 - Others

Why **Non-Functional Requirements (NFRs)**?

- Consider a brochure from an automobile manufacturer:
 - When you buy our car, you can now drive to a store...



- Consider a brochure from a cellular phone manufacturer:
 - When you buy our cellular phone, you can now call your friend.
 - Well, ...

Non-Functional Concerns drive just about everything

➤ With cellular phones:

“... enhancements enable the *best possible* operation of your mobile ... in *various conditions*. ...

The earpiece fits in either ear allowing for *convenient* and *reliable* access to *all basic call controls*. ... To *maximize call security*, the headset also supports ... the wireless connection for *compatible* ... models.”



➤ With home networking:

“... is the total home networking solution ... linking variety of digital home appliances as one.

It enables you to enjoy *convenient, pleasant, comfortable*, and *reliable* living environment *at any time and any place*.



Why NFRs?

- **With CASE tool software:**

- The basic function is provision of some services
- "... is a *powerful, easy-to-use* application definition platform used by business experts to *quickly* assemble functionally *rich* simulations of Web-based applications *in a matter of hours*. ... Using the *easy to learn*, drag-and-drop paradigm ..., business people can *quickly* lay out the page flow of simulations and create *high fidelity* pages that *precisely* mimic not only the look and feel of the final application, ..."

But

Software is harder than hardware

Soft is harder to deal with than hard

NF

F

What are Non-Functional Requirements?

- **-ilities**: understandability, usability, modifiability, inter-operability, reliability, portability, dependability, flexibility, availability, maintainability, scalability, (re-)configurability, customizability, adaptability, stability, variability, volatility, traceability, verifiability, predictability, compatibility, survivability, accessibility, ...
- **-ities**: security, simplicity, clarity, ubiquity, integrity, safety, modularity, nomadicity, ...
- **-ness**: user-friendliness, robustness, timeliness, responsiveness, correctness, completeness, conciseness, cohesiveness, ...
- **...and there are many more**: convenience, comfort, performance, efficiency, accuracy, precision, slim, esthetics, consistency, coherence, fault tolerance, self-healing capability, autonomy, cost, development time, time-to-market, long-lasting battery, low coupling, ...

soft

subjective, ambiguous, conflicting, global

NFRs: IEEE definition

“**non functional requirement** – in software system engineering, a software requirement that describes *not what* the software will do, *but how* the software will do it, for example, software performance requirements, software external interface requirements, design constraints, and software quality attributes. Nonfunctional requirements are difficult to test; therefore, they are usually evaluated subjectively.”

General Observations

“**non functional requirement** – generally informally stated, often contradictory, difficult to enforce during development and evaluate for the customer prior to delivery”

subjective in both definitions & solutions

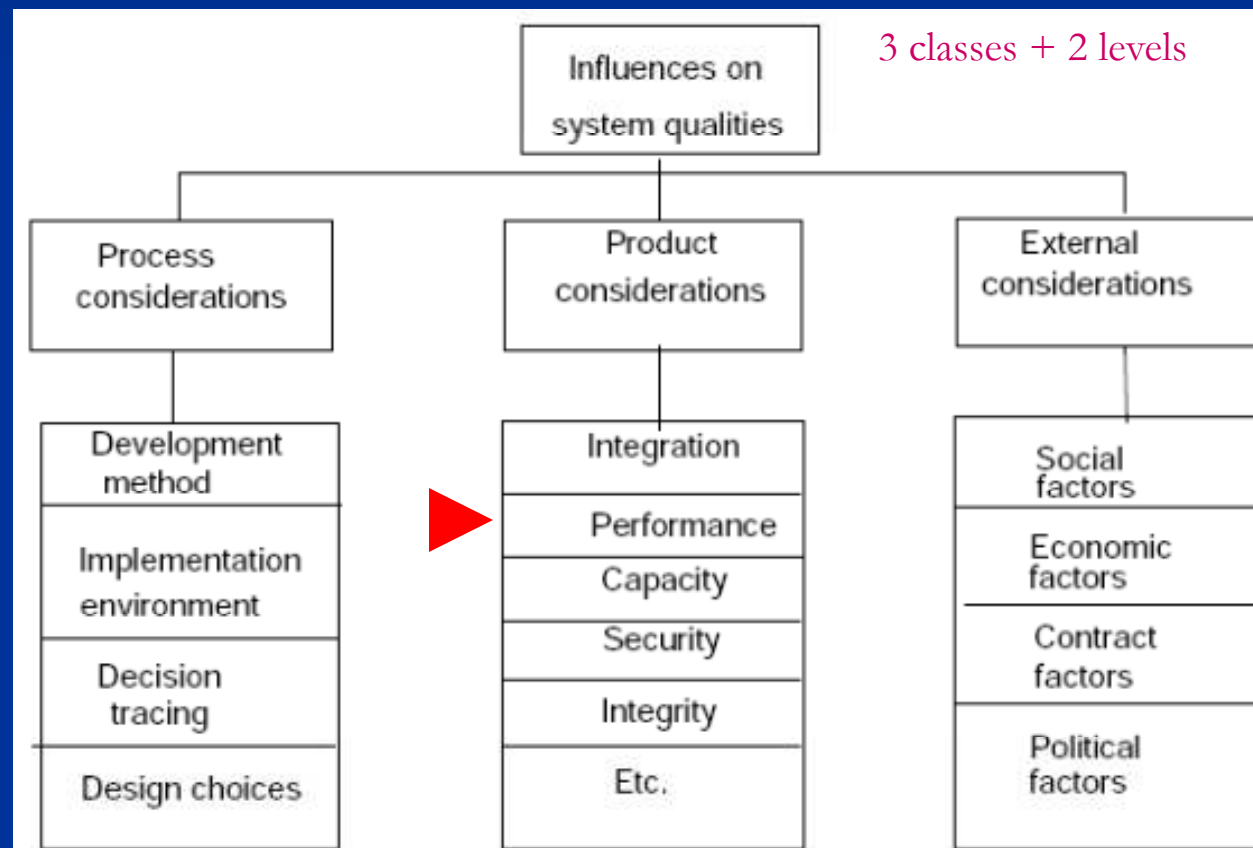
Classification is supposed to bring order into chaos, but ...

Classification 1 (Roman, IEEE Computer 1985) 6 classes + 2-3 levels

- **Interface requirements:** describe how the system is to interface with its environment, users and other systems. E.g., user interfaces and their qualities (e.g., user-friendliness)
- ▶ **Performance requirements:** describe performance constraints involving
 - time/space bounds, such as workloads, response time, throughput and available storage space. E.g., “system must handle 100 transactions/second”
 - ▶ reliability involving the availability of components and integrity of information maintained and supplied to the system. E.g., “system must have less than 1hr downtime/3 months”
 - security, such as permissible information flows
 - survivability, such as system endurance under fire, natural catastrophies
- **Operating requirements:** include physical constraints (size, weight), personnel availability, skill level considerations, system accessibility for maintenance, etc.
- **Lifecycle requirements:** can be classified under two subcategories:
 - quality of the design: measured in terms such as maintainability, enhanceability, portability.
 - limits on development, such as development time limitations, resource availability, methodological standards, etc.
- **Economic requirements:** immediate and/or long-term costs
- **Political requirements**

subjective in both definitions & solutions

Classification 2 - Process, Product and External considerations [Bonomerille 1992]

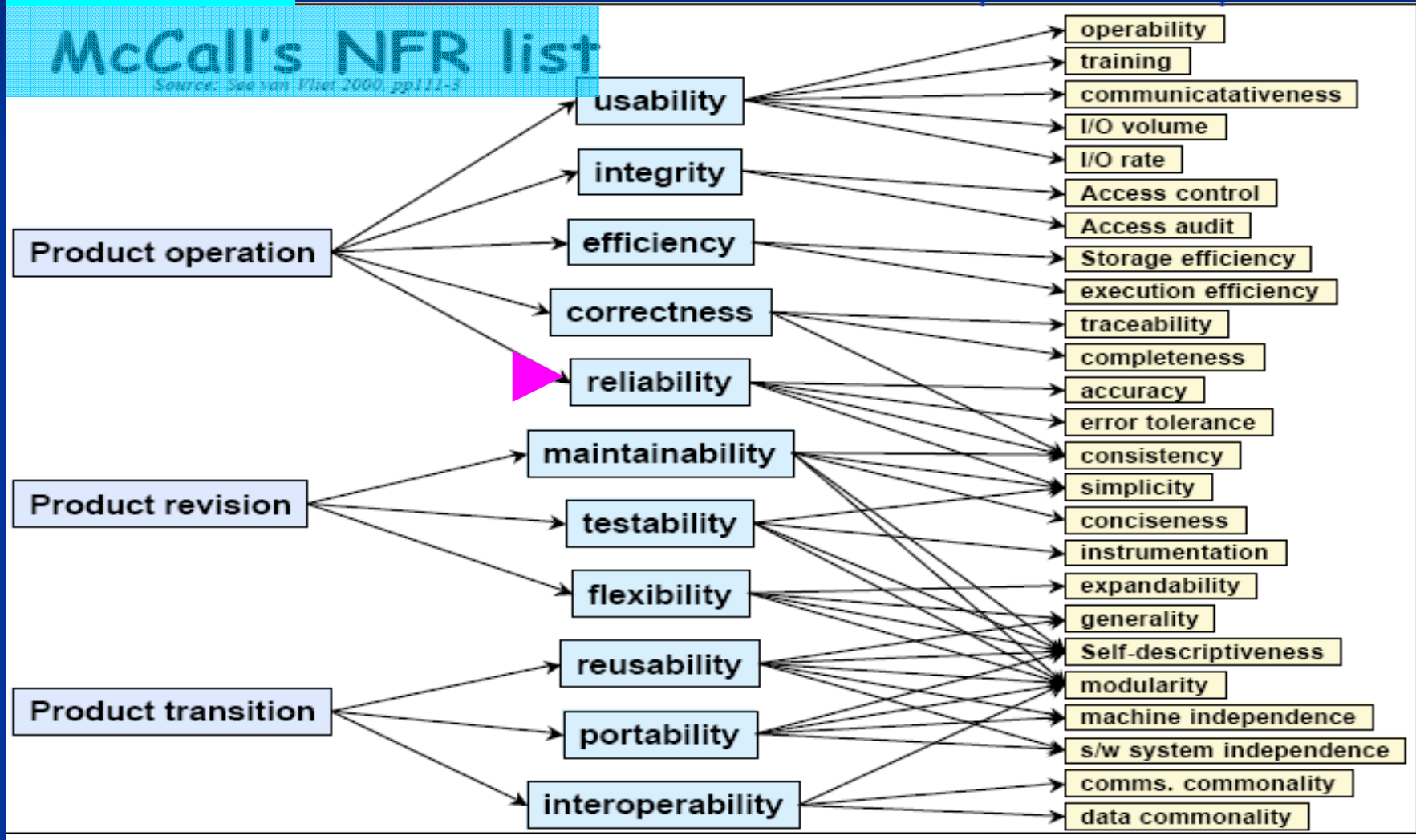


NFRs:

subjective in both definitions & solutions

Classification 3 3 classes + 3 levels

Note correlations



NFRs:

subjective in both definitions & solutions

Classification 4 4+ classes + 2 levels

Dimensions of Quality –Components of FURP+

[Grady1992]

<p>Functionality</p>	<p>Feature set capabilities, security, generality</p>	
<p>Usability</p>	<p>Human factors aesthetics, consistency, documentation</p>	
<p>▶ Reliability</p>	<p>Frequency/severity of failure, recoverability, predictability, accuracy, MTBF</p>	
<p>▶ Performance</p>	<p>Speed efficiency, resource usage, throughput, response time</p>	
<p>Supportability</p>	<p>Testability Adaptability Compatibility Serviceability Localizability</p>	<p>Extensibility Maintainability Configurability Installability Robustness</p>

NFRs:

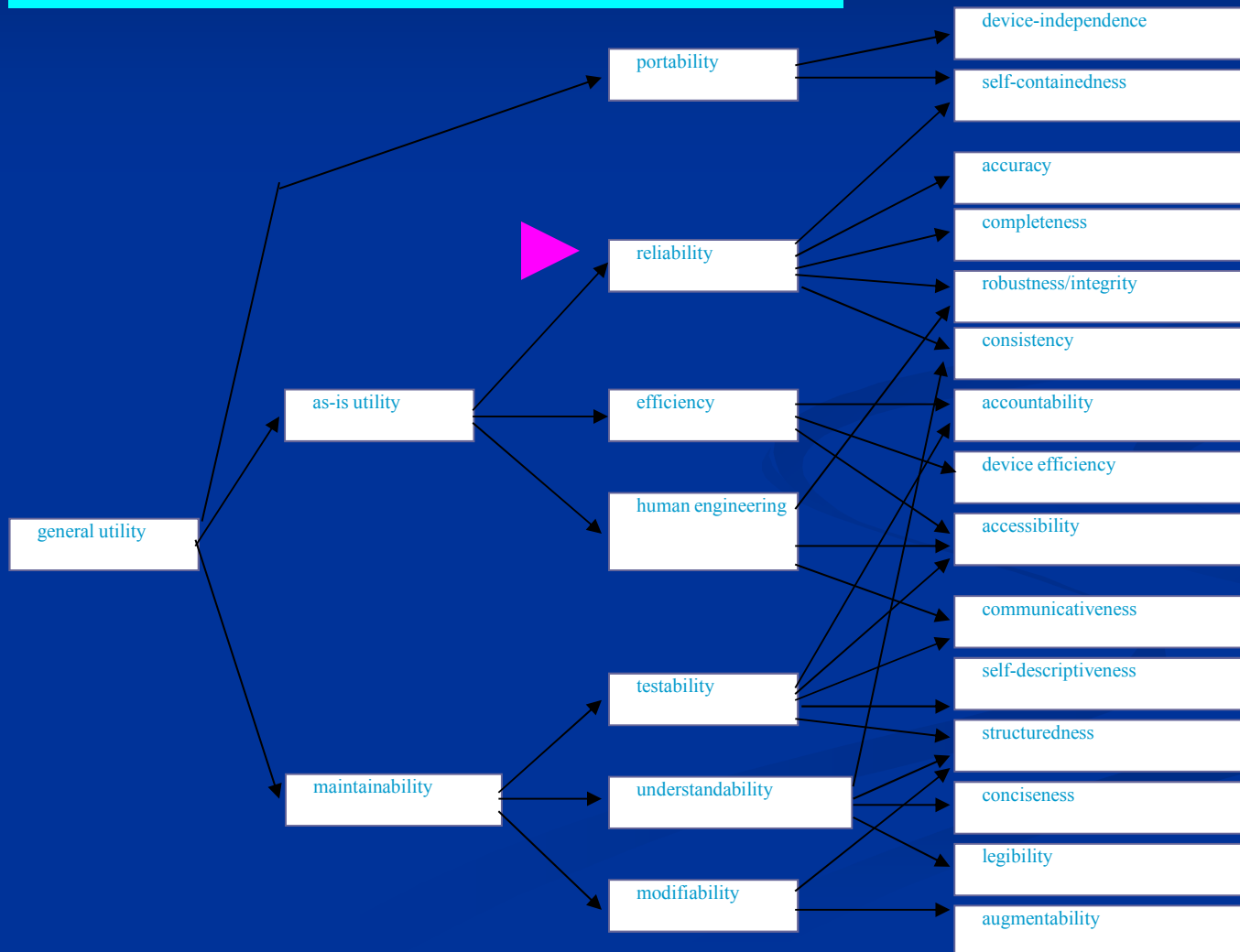
subjective in both definitions & solutions

Classification 5

2 classes + 3 levels

Software Quality Tree [Boehm 1976]

Note correlations



NFRs:

subjective in both definitions & solutions

Relationship Between Design Goals

Adapted from [Brugge]

Client (Customer, Sponsor)



- Low cost
- Increased Productivity
- Backward-Compatibility
- Traceability of requirements
- Rapid development
- Flexibility

Runtime Efficiency

Reliability

Portability
Good Documentation

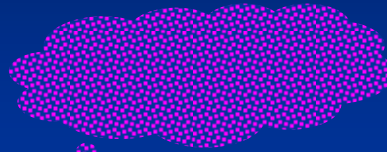
- Minimum # of errors
- Modifiability, Readability
- Reusability, Adaptability
- Well-defined interfaces

- Functionality
- User**-friendliness
- Ease of **Use**
- Ease of learning
- Fault tolerant
- Robustness

Developer/ Maintainer

NFRs & RE Processes: Why?

Quality of product ← Quality of Process



☐ Garbage in garbage out,
so get the right requirements



☐ Garbage thru garbage out,
so get the right process

Product

§ Evolution is inevitable – traceability is a virtue
Lawrence Chung

Approaches to NFRs

Measurement of products or systems is absolutely fundamental to the engineering process. I am convinced that measurement as practised in other engineering disciplines is *IMPOSSIBLE* for software engineering [Sommerville; <http://www.utdallas.edu/~chung/SE3354Honors/IEEEInaugural.pdf>]

■ Product vs. Process?

The most important things can't be measured [Deming]

■ Product-oriented Approaches

- Focus on system (or software) quality
- Aim is to have a way of measuring the product once it's built – metrics

■ Process-oriented Approaches

- Focus on how NFRs can be used in the design process
- Aim is to have a way of making appropriate design decisions

■ Quantitative vs. Qualitative?

■ Quantitative Approaches

- Find measurable scales for the quality attributes
- Calculate degree to which a design meets the quality targets

■ Qualitative Approaches

- Study various relationships between quality goals
- Reason about trade-offs etc.

Not everything that can be counted counts, and not everything that counts can be counted.
[Albert Einstein]

NFRs & RE Processes:

So, where are NFRs in an RE Process?

- *Before FRs?*
- *After FRs?*
- *At the same time with FRs?*
- *...and what about Business objectives/goals, system architectures, system models, SS, SRS, ...?*

But, should we perhaps better know about the various relationships between NFRS and such and such, before answering these questions, more clearly, understandably, concisely, precisely, agreeably, ...?



M, Prog \models S; G^s, S, D \models R; (G^s, R, D \models G) \vee (G^s, R, D \sim G); (G \models \neg P) \vee (G \sim \neg P)

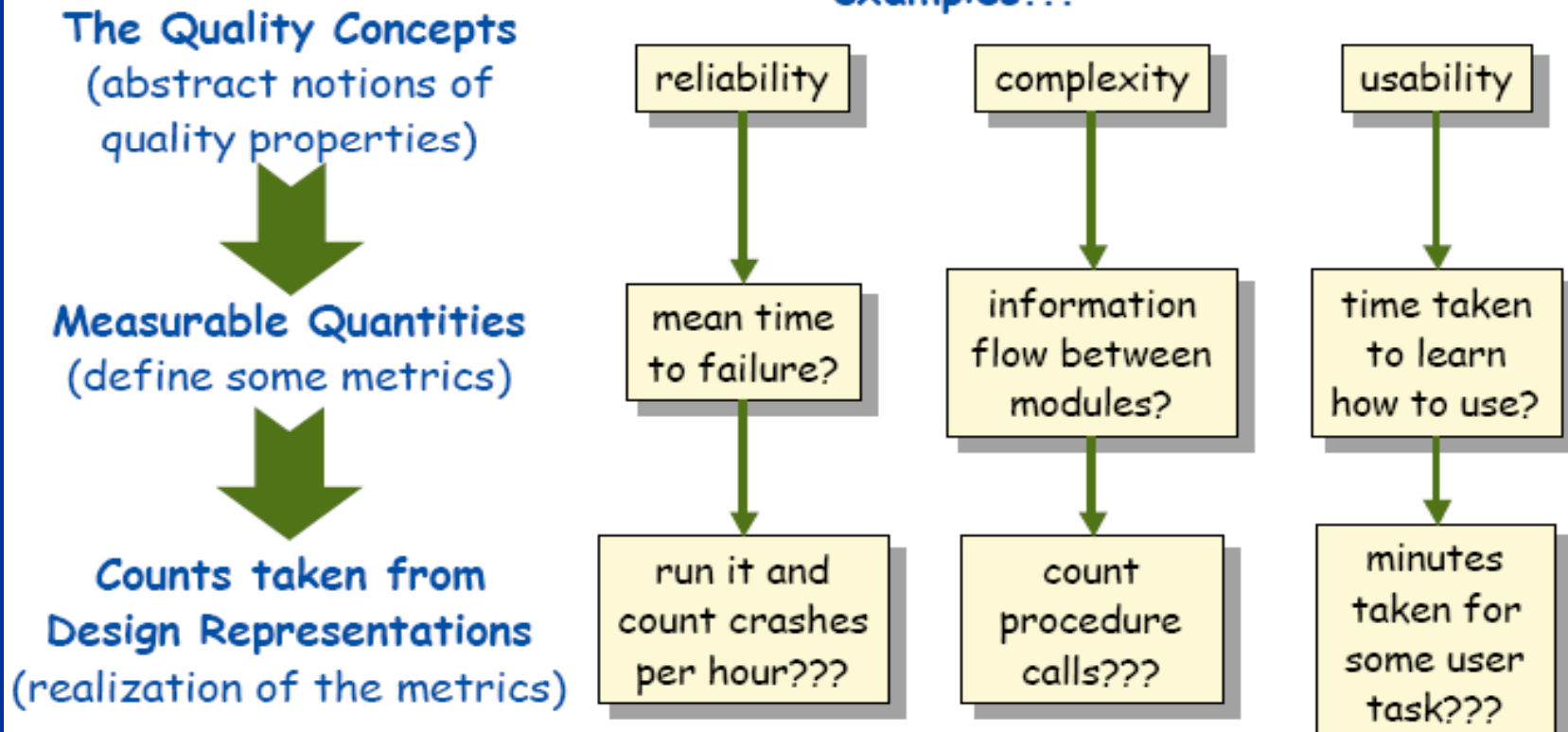
Product-oriented approaches

Making Requirements Measurable

Source: Budgen, 1994, pp60-1

We have to turn our vague ideas about quality into measurables

examples...



Product-oriented approaches

Quality Metrics:

Property	Metric
Speed	transactions/sec, response time, screen refresh time
Size	KBytes, LOCs, Function Points, Complexity measures
Ease of use	transactions/sec, response time, screen refresh time

- * usual metrification process:
 1. *determine a set of desirable attributes (i.e., ilities)*
 2. *determine relative importance / weight of such attributes*
 3. *evaluate the quality (rating) of each of the attributes*
 4. *compute weighted rating for each*
 5. *sum up all the weighted ratings*

Property	relative weight	rating	weighted rating
Speed	.3	6	1.8
Size	.6	5	3.0
Ease of use	.1	7	0.7
Overall Quality			5.5/10

- * an inexact science at this point
- * however, aids in understanding the factors that affect sw quality
 - a first-cut approximation
 - very poor quality factor

NFRs: Portability

- The degree to which software running on one platform can easily be converted to run on another platform
- E.g., number of target statements (e.g., from Unix to Windows)
- Hard to quantify, since it is hard to predict what a “next generation” platform might be like
- Can be enhanced by using languages, OSs and tools that are universally available and standardized.

E.g., C/C++/C#/Java

J2EE/J2ME/.NET

NFRs: Reliability

- the ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.
- theory and practice of hardware reliability are well established; some try to adopt them for software



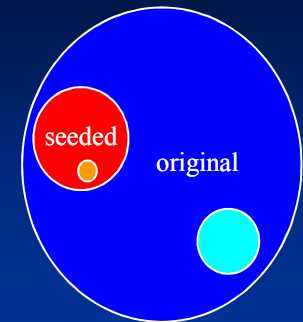
- one popular metric for hardware reliability is mean-time-to-failure (MTTF)
"Bathtub" curve characterizes MTTF:



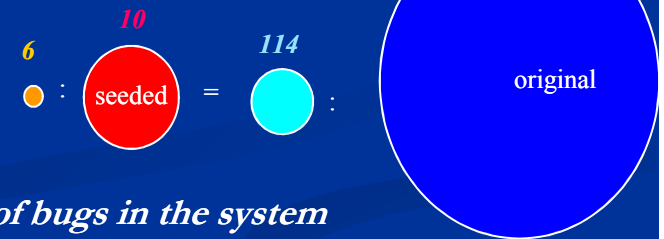
- Infant mortality:**
Given a large population of a particular component, many will fail soon after development due to inaccuracies in the manufacturing process;
- Issues:**
 - Do 2 different software copies have different characteristics?
 - Does software wear & tear by decomposition?
 - Does software obey physical laws?

NFRs: Reliability

- Sometimes reliability requirements take the form:
"The software shall have no more than X bugs/1K LOC"
 But how do we measure bugs at delivery time?



- Bebugging Process** - based on a Monte Carlo technique for statistical analysis of random events.
 - before testing, a known number of bugs (**seeded** bugs) are secretly inserted.
 - estimate the number of bugs in the system
 - remove (both known and new) bugs.



$$\begin{aligned} \# \text{ of detected seeded bugs} / \# \text{ of seeded bugs} &= \# \text{ of detected bugs} / \# \text{ of bugs in the system} \\ \# \text{ of bugs in the system} &= \# \text{ of seeded bugs} \times \# \text{ of detected bugs} / \# \text{ of detected seeded bugs} \end{aligned}$$

Example: secretly **seed 10** bugs (say, in 100 KLOC)
 an independent test team detects 120 bugs (6 for the seeded)
 $\# \text{ of bugs in the system} = 10 \times 120 / 6 = 200$
 $\# \text{ of bugs in the system after removal} = 200 - 120 - 4 = 76$

$$190 - 114; 100000 - (190-114)/100000$$

- But**, deadly bugs vs. insignificant ones; not all bugs are equally detectable; (Suggestion [Musa87]:
"No more than X bugs/1K LOC may be detected during testing"
"No more than X bugs/1K LOC may be remain after delivery,
as calculated by the Monte Carlo seeding technique"

NFRs:

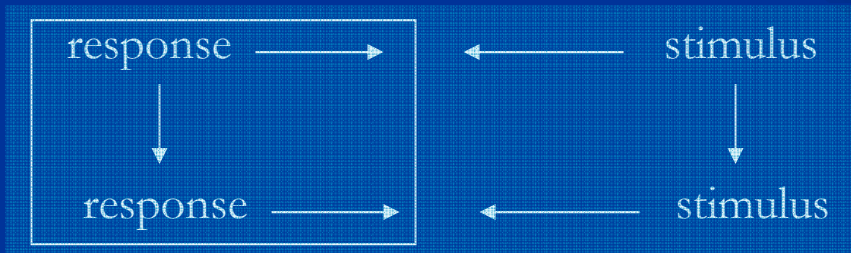
Efficiency

- refers to the level at which a software system uses scarce computational resources, such as CPU cycles, memory, disk space, buffers and communication channels
- can be characterized along a number of dimensions:
Capacity: maximum number of users/terminals/transactions ...

Degradation of service: what happens when a system with capacity X widgets per time unit receives $X+1$ widgets?

- Let the system handle the load, perhaps with degraded performance
- Let the system crash

Timing constraints: Let stimulus refer to an action performed by the user/environment, and response refer to an action generated by the system.



- **stimulus-response:** e.g., "the system will generate a dial tone within 10 secs from the time the phone is picked up"
- **response-response:** e.g., "the system will record that the phone is in use no later than 1 micro-second after it had generated a dial tone"
- **stimulus-stimulus:** e.g., "the user will type her password within 15 secs from typing her login name"
- **response-stimulus:** e.g., "the user will start dialing the phone number within 1 minute from getting the dial tone"

NFRs: Usability

- broadly – quality; fit to use
narrowly - good UI
- **Usability inspection:**
finding usability problems in UI design, making recommendations for fixing them, and improving UI design.
- **Heuristics:** a set of criteria against which usability of UI design is evaluated
- **"9 usability heuristics"** [Nielsen90]
 - *Promptness* no undue delay in accepting info items and responding to requests
 - *Tolerance* no hang-ups against errors, delays, unexpected behavior, etc.
 - *Guidance* providing guidance for correcting errors, generating reminders, etc.
 - *Coherence*
- **"10 usability heuristics"** [Molich and Nielsen90]
 - *Simple and natural dialogue; Speak the user's language*
 - *Minimize the user's memory; Consistency; Feedback*
 - *Clearly marked exits; Shortcuts*
 - *Precise and constructive error messages; Prevent errors*
 - *Help and documentation*

NFRs:

Usability

- All users will be satisfied with the usability of the product.
- 95% of all users will be satisfied with the usability of the product.
- 95% of the users will be able to complete representative tasks without requiring assistance (e.g., modifying exclusion date set)
- 95% of the users will be able to complete representative tasks by the third attempt without requiring assistance
- 95% of the users will be able to complete tasks X Y Z by the third attempt without requiring assistance
- 95% of the users will be able to complete tasks X Y Z in less than 10 minutes without requiring assistance
- 95% of the users will be able to complete task X in less than 10 minutes without requiring assistance
- 80% of the users will be able to complete task Y in less than 10 minutes
- 77% of the users will be able to complete task Z in less than 5 minutes

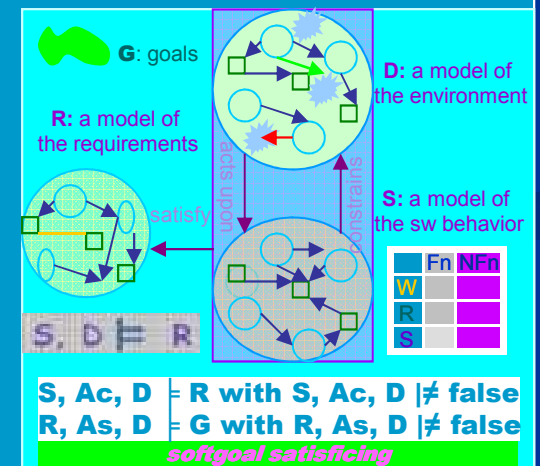
Non-Functional Requirements

Practices and Recommendations:
A Brief Synopsis

- Why
- What
- Some Classification Schemes
- NFRs and RE Processes
- Product-Oriented Approach: Some Individual NFRs
- The NFR Framework
- Appendix
 - With Rational Unified Process and UML
 - With Volere Requirements Specification Templates
 - Others

Non-Functional Requirements

What - Essential Concepts



$M, Prog \models S; G^s, S, D \models R; (G^s, R, D \models G) \vee (G^s, R, D \sim G); (G \models \neg P) \vee (G \sim \neg P)$

NFRs:

functional vs. **non-functional**: a mathematical perspective

- (mathematical) function:

$$f_1: I \rightarrow O$$

$$f_2: I_1 \times I_2 \rightarrow O$$

e.g.: sum: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

$$\mathbf{F: I \rightarrow O}$$

add: $2, 5 \rightarrow 7$

- **non-functional**:
 - How **fast** can it be done?
 - How **precise** is the answer?
 - How **easy** is it to figure out how to use it?
 - How **robust** is it concerning the 2nd input of f_2 ?
 - **Who** can use it?
 - Can it be **changed easily**?
 - How much would it **cost** to design and implement it?

NFRs:

functional vs. **non-functional**: a mathematical perspective

- (mathematical) function:

$$f(x, y) = f_1(f_2(x), f_3(y))$$

- non-functional:

$$nf(x, y) = nf_1(nf_2(x), nf_3(y))$$

$$nf(x, y) = nf_1(nf_2(n(x)), nf_3(n(y)))$$

Global nature

NFRs: subjective, graded, interacting

- Subjective vs. objective:



- Graded:

worse

expensive

slower

better

cheaper

faster



- Interacting:

- Conflicting: the whole is less than the sum of its parts
- Synergistic: the whole is more than the sum of its parts

NFRs: subjective in both definitions & solutions

Classification 1 [Roman, IEEE Computer 1985]

- **Interface requirements:** describe how the system is to interface with its environment, users and other systems. E.g., user interfaces and their qualities (e.g., user-friendliness)

- **Performance requirements**

- time/space bounds, such as must handle 100 transactions per second
- reliability involving the availability of the system. E.g., "system must be available 99.999% of the time"
- security, such as permission requirements
- survivability, such as system recovery

- **Operating requirements:** in terms of hardware and software considerations, system access, etc.

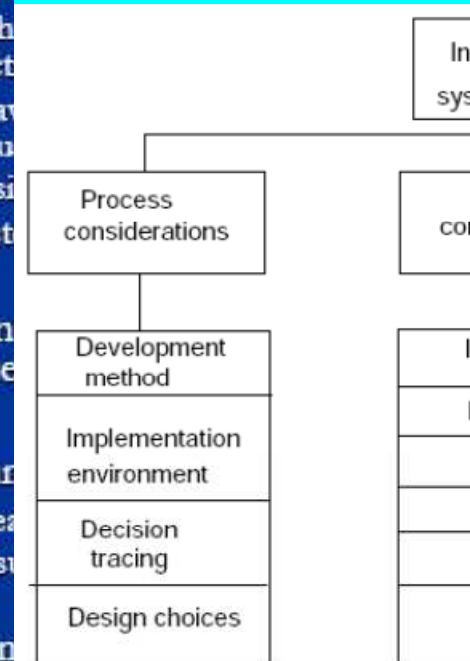
- **Lifecycle requirements:** can be further divided into

- quality of the design: measurement, etc.
- limits on development, support, etc.

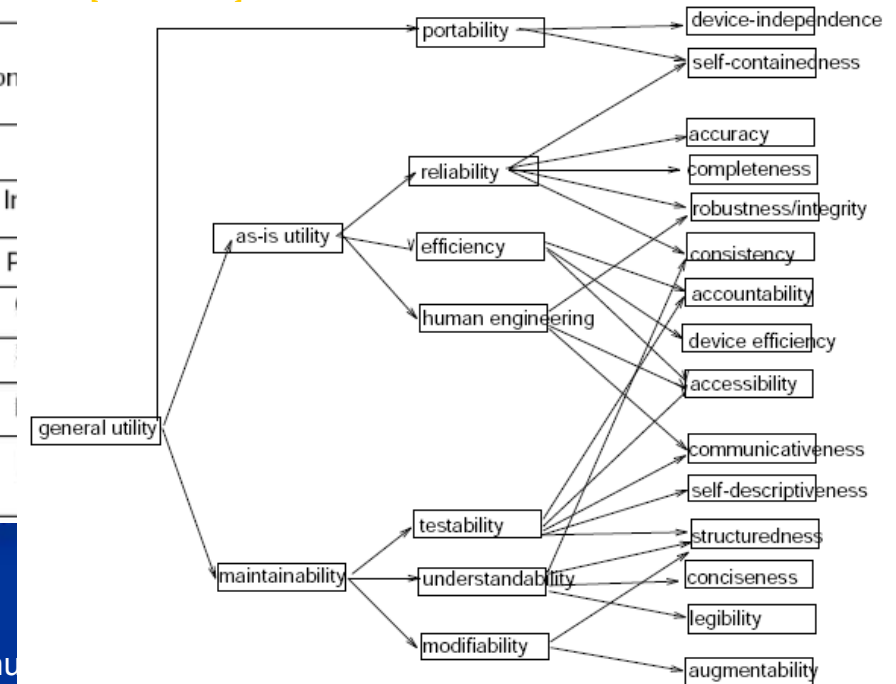
- **Economic requirements:** in terms of cost, etc.

- **Political requirements**

Classification 2 - Process, Product and External considerations [Somerville 1992]



Classification 5 - Software Quality Tree [Boehm 1976]



NFRs:

subjective in both definitions & solutions

➤ Consider “security” – problem is subjective

- Protection of data alone, fine with Chris
- Protection of data, and data availability, fine with Pat
- Protection of data, and data availability, and data accuracy, fine with Alex
- Protection of data, and data availability, and data accuracy, and filtering of viruses, fine with Neo
- Protection of data, and data availability, and data accuracy, and filtering of viruses, and blocking adware, fine with Gail

➤ Consider “security” – solutions are subjective

- A password authentication fine with Chris
- A password authentication, with periodic change, fine with Pat
- A password, together with a fingerprint verification, fine with Alex
- A password, with a fingerprint verification rechecked every hour, fine with Neo
- A password, with a fingerprint verification rechecked every hour, and co-presence of two people, fine with Gail

NFRs:

subjective – and also relative in priorities

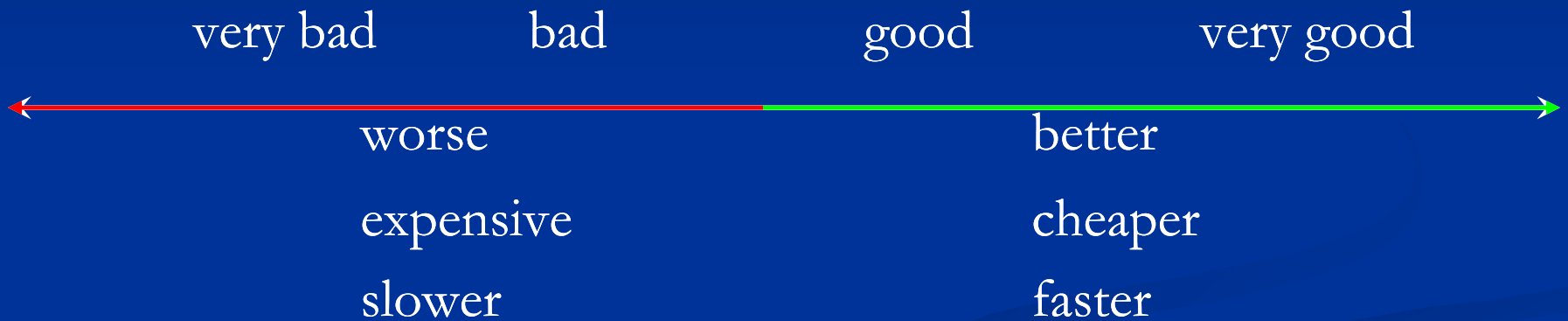
security  performance  reliability

usability  security  safety  reliability

reliability  security

NFRs:

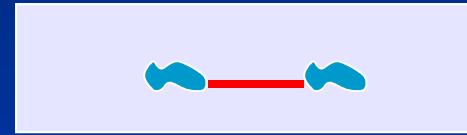
graded in both definitions and solutions – and relative



- ❑ Protection of data alone good
- ❑ A password authentication alone bad
- ❑ Protection of data alone << Protection of data, and data availability
- ❑ A password authentication << A password, together with a fingerprint verification

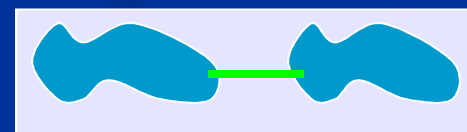
NFRs: interacting

- **Conflicting:** the whole is less than the sum of its parts



- ✓ A password, with a fingerprint verification rechecked every hour, fine for security
- ✓ Simplicity is the key for ease-of-use

- **Synergistic:** the whole is more than the sum of its parts



- ✓ A password, with a fingerprint verification rechecked every hour, fine for security
- ✓ Restricted access is good for data accuracy

Non-Functional Requirements

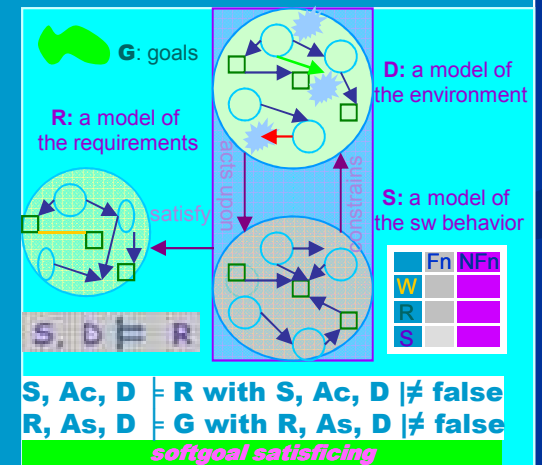
What - Essential Concepts

- non-functional,
 - subjective,
 - graded,
 - interacting
 - – and relative
- - in both definitions & solutions

Non-Functional Requirements

How 1 - Essential Tasks

Goal-oriented analysis focuses on the description and evaluation of alternatives and their relationship to the organizational objectives.



$M, Prog \models S; G^s, S, D \models R; (G^s, R, D \models G) \vee (G^s, R, D \sim G); (G \models \neg P) \vee (G \sim \neg P)$

NFRs:

functional vs. non-functional: a mathematical perspective

- (mathematical) function:

$$f_1: I \rightarrow O$$

$$f_2: I_1 \times I_2 \rightarrow O$$

$$\text{e.g.: sum: } \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

$$f(x, y) = f_1(f_2(x), f_3(y))$$

$$nf(x, y) = nf_1(nf_2(x), nf_3(y))$$

$$= nf_1(nf_2(n(x)), nf_3(n(y)))$$

- non-functional:

- How fast can it be done? *Fast, Fast(f), Fast(f₂)*

- How precise is the answer? *Precise, Precise(f), Precise(O)*

- How easy is it to figure out how to use it?

Easy-to-learn, Easy-to-learn(f), Easy-to-learn(f₂), Easy-to-learn(x)

- How robust is the input? *Robust, Robust(I₁), Robust(I₂)*

- Who can use it? *Security, Security(f), Security(I), Security(O), Security(f₂), Accessibility, Accessibility(f), Accessibility(O)*

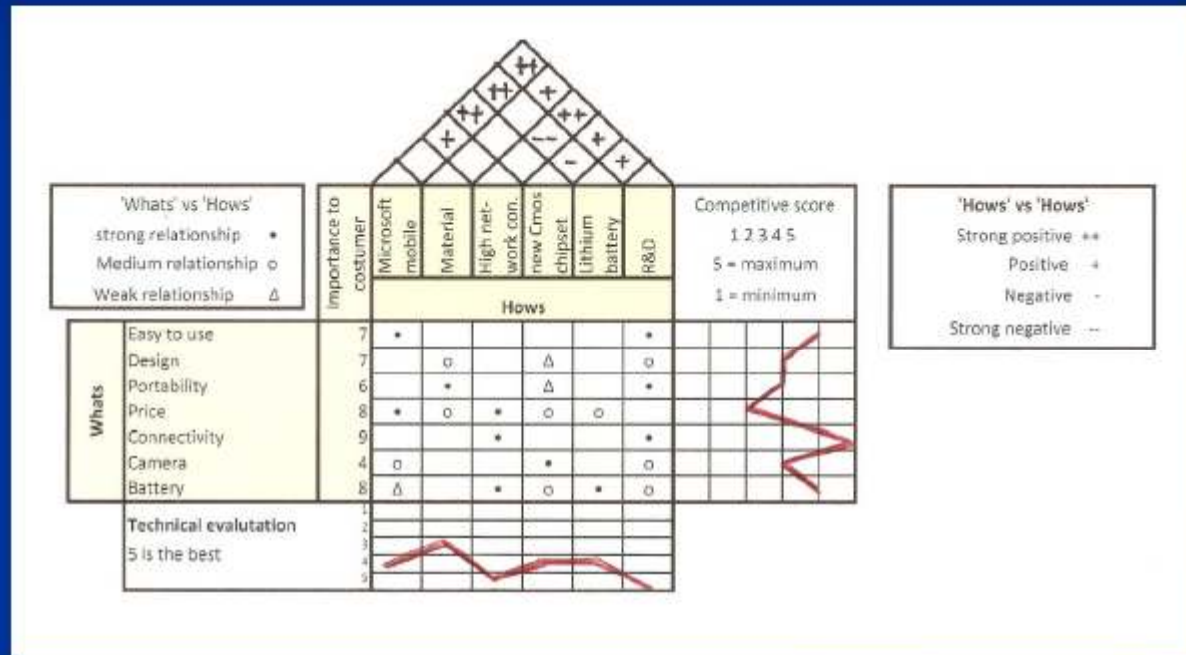
- Can it be changed easily?

Changeability, Changeability(f), Changeability(f₂)

- How much would it cost?

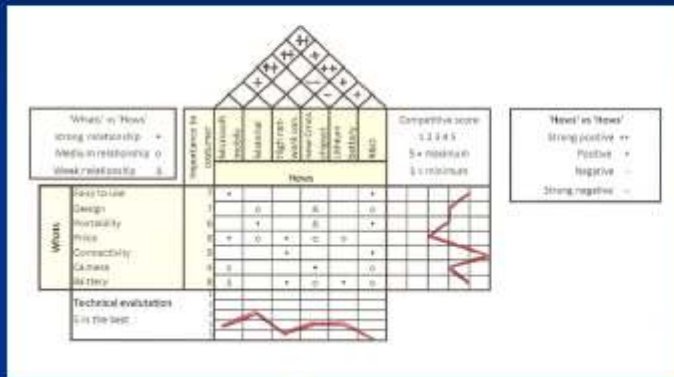
Cost, Design-cost(f), Implementation-cost(f), Testing-cost(f₂)

Quality Function Deployment (QFD) aka House of quality



http://blogs.warwick.ac.uk/images/pic/2007/10/22/qfd_v2.jpg

Quality Function Deployment (QFD) aka House of quality



- Intuitive
- Rich with relationships
- No relationship with problems
- NFRs not structured
- Solutions not structured

↕ ???



Goals



Problems

The NFR Framework

Chung, Nixon, Yu and Mylopoulos, Non-functional Requirements in Software Engineering

- **Softgoal**
 - No clear-cut definition and or criteria as to whether it is satisfied or not; NFRs are subjective, relative, and interdependent
 - No optimal solution
- **satisficing**
 - One softgoal can **contribute positively or negatively, fully or partially**, towards other softgoals (i.e., achieved not absolutely but within acceptable limits).
- **Softgoal Interdependency Graphs (SIGs)**
 - For modeling non-functional requirements and interdependencies between them
- **Labeling (evaluation) procedure**
 - To determine the degree to which softgoals/contributions are satisfied
- **Catalogues**
 - for knowledge of NFR satisficing and correlations, much like patterns for design

NFRs act as the basis for exploring alternatives and as the criteria for selecting among alternatives, hence for rationalization

Qualitative in nature, Process oriented

The NFR Framework



Softgoal Interdependency Graph (SIG): Three types of refinements

Softgoal types:

- NFR 
- Operationalizing 
- Claim (supporting/explaining a choice) 

Formal Softgoal := Priority Type [Topic]





contribution types:

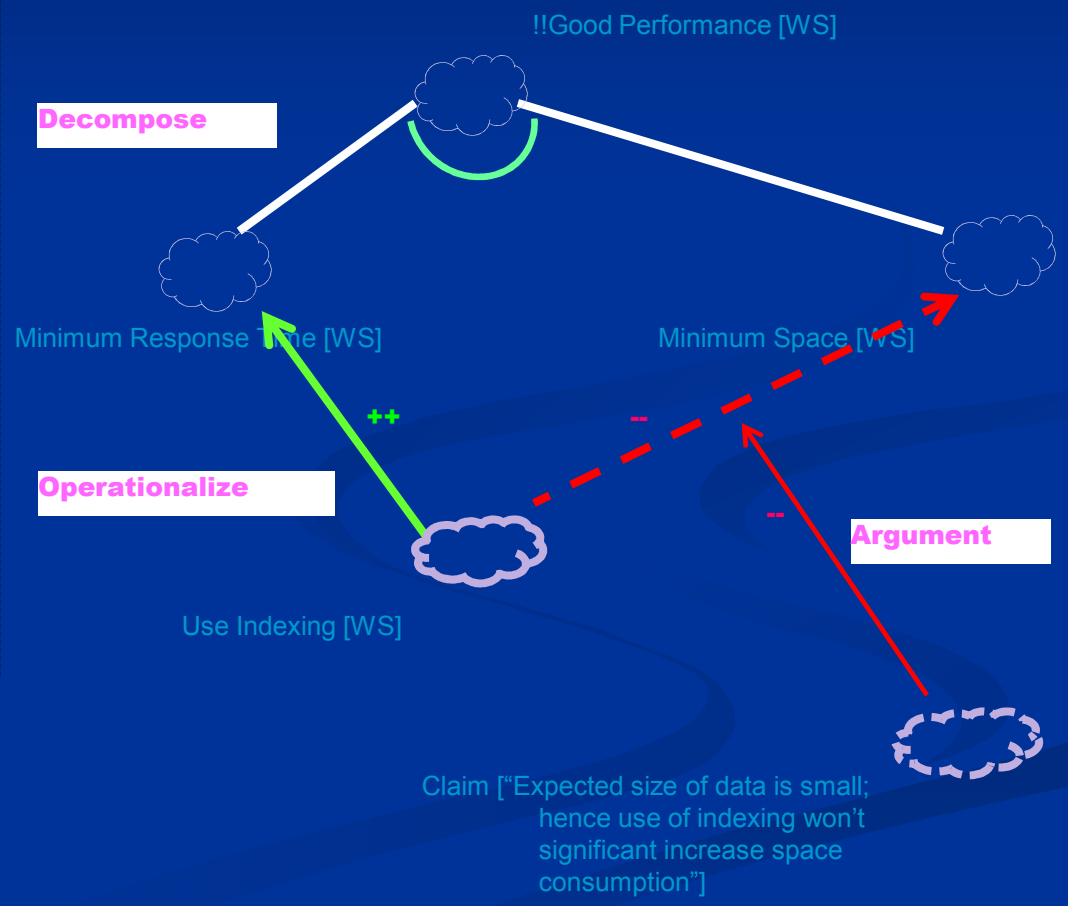
- AND (decomposition) 
- OR (alternatives) 

Make >> Help >> Hurt >> Break



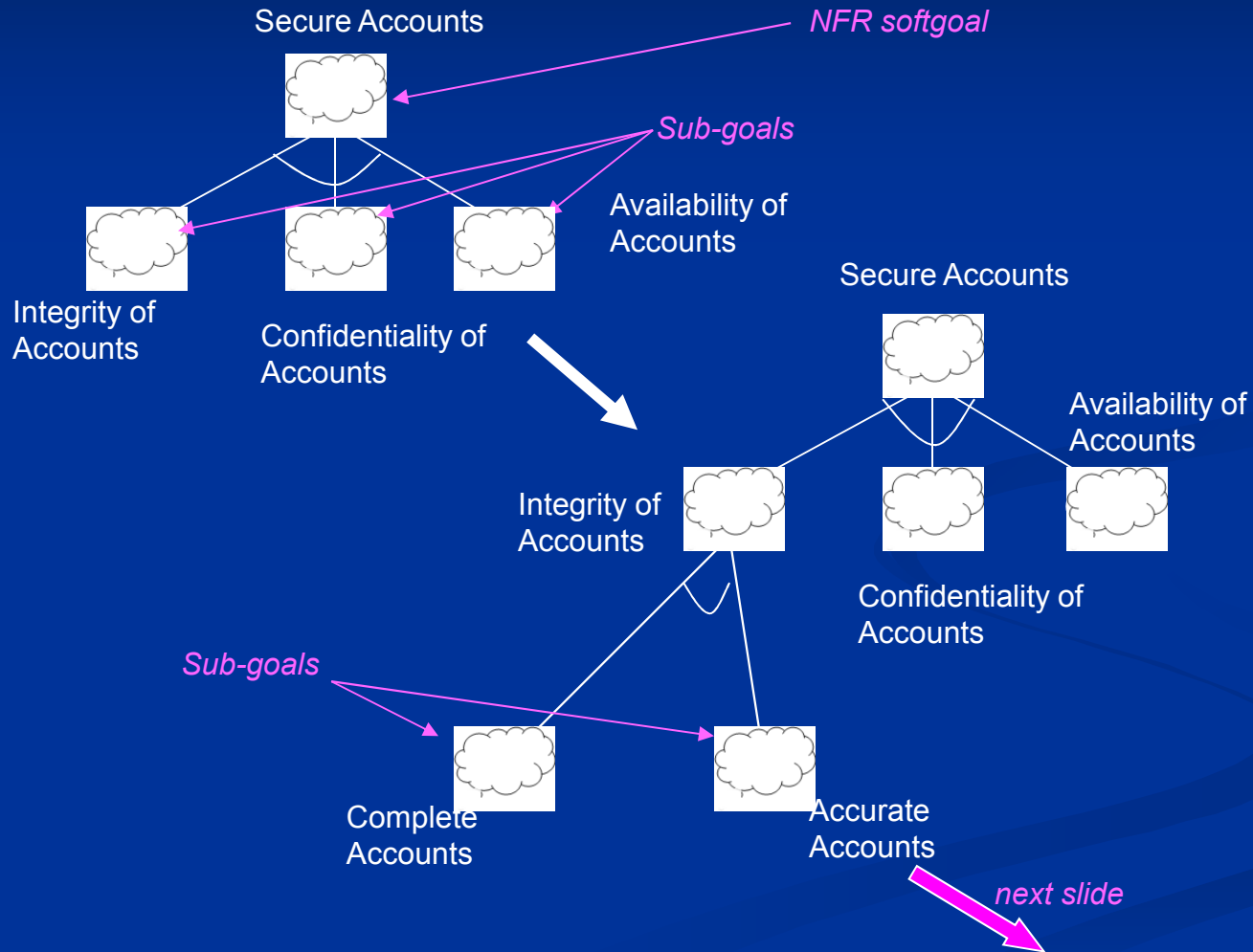
Label types

- satisfied 
- denied 
- conflicting 
- undetermined 



The NFR Framework

Qualitative in nature, Process oriented



The NFR Framework

Softgoal Interdependency Graph (SIG)

Softgoal types:

- NFR
- Operationalizing (satisficing technique)
- Claim (supporting/explaining a choice)

Softgoal := Informal Sg | Formal Sg

Formal Sg := Type [Topic]

Contribution types:

- AND (decomposition)
- OR (alternatives)



Make >> Help >> Hurt >> Break

Labels (evaluation of softgoals/contributions)

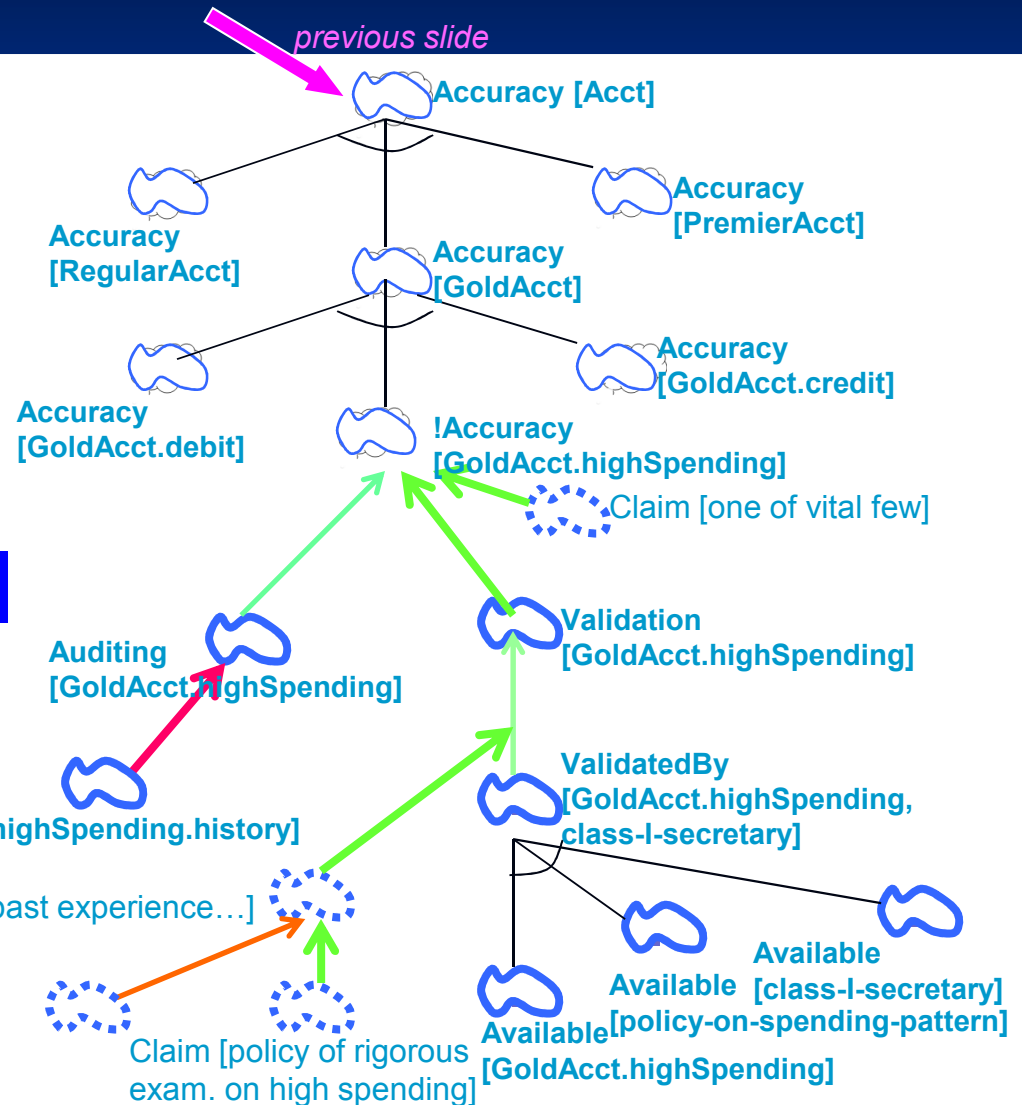
- satisfied
- denied
- conflicting
- undetermined



Destroy
[GoldAcct.highSpending.history]

Claim [past experience...]

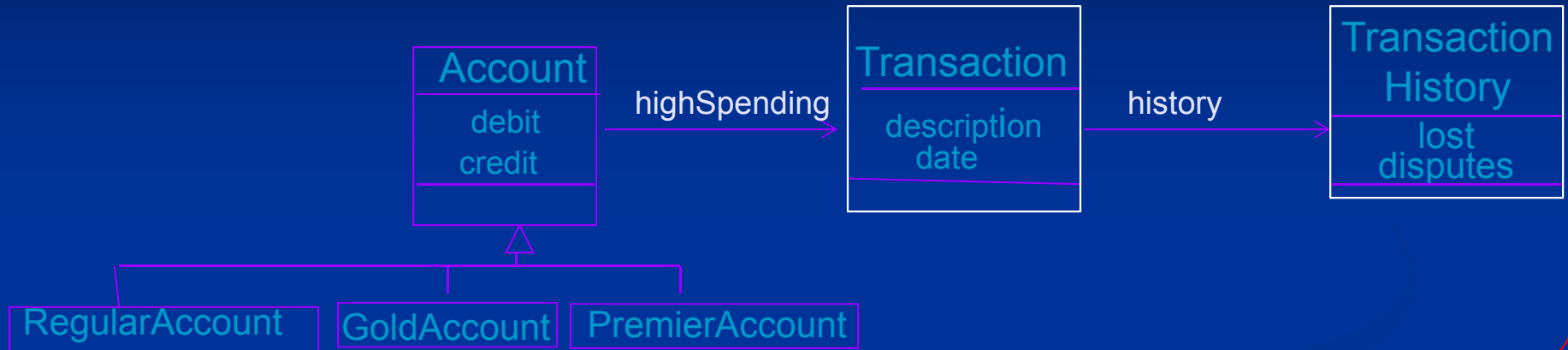
Claim [policy of rigorous exam. on high spending]



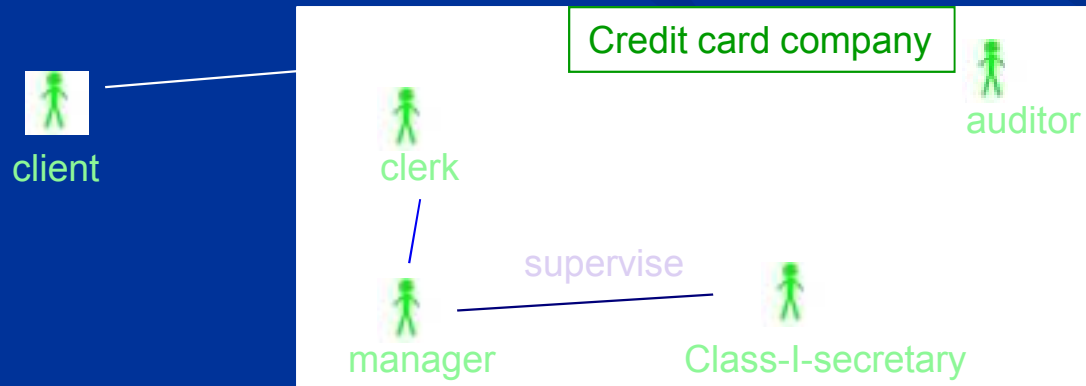
Domain, Agent, User, Platform, ...

as topic (parameters)

Domain



Agent Interaction

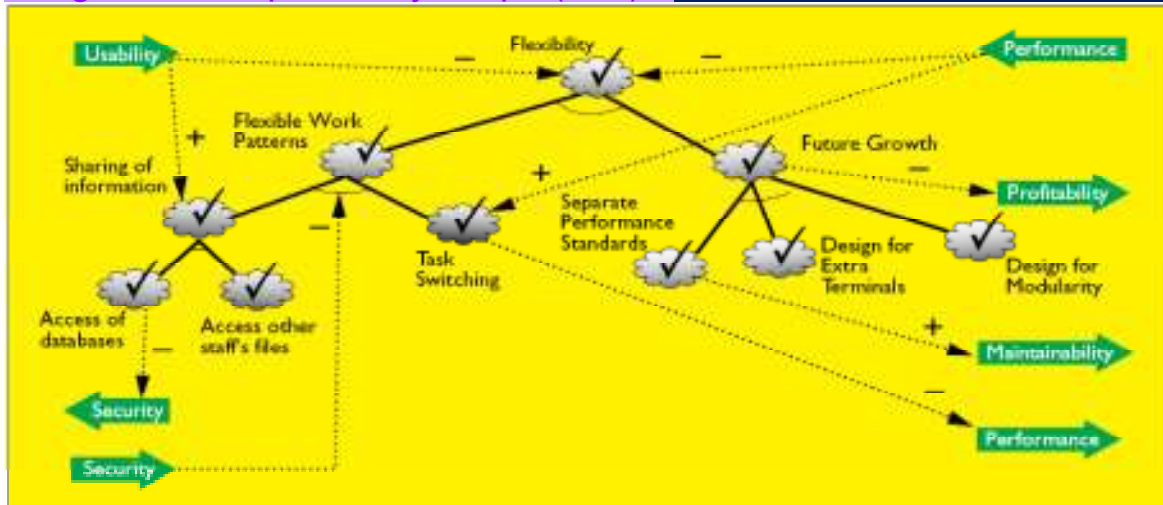


The NFR Framework

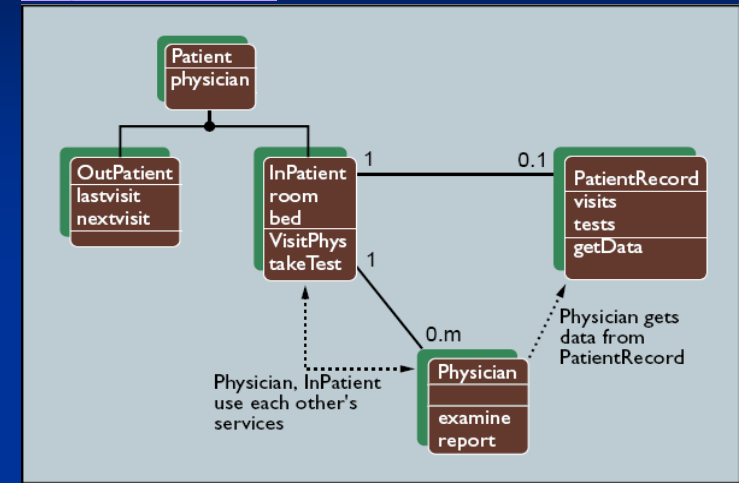
[J. Mylopoulos, L. Chung, E. Yu, "From object-oriented to goal-oriented requirements analysis", CACM, pp31-37. ACM Press]

Example: A small portion of a hospital model for requirements analysis

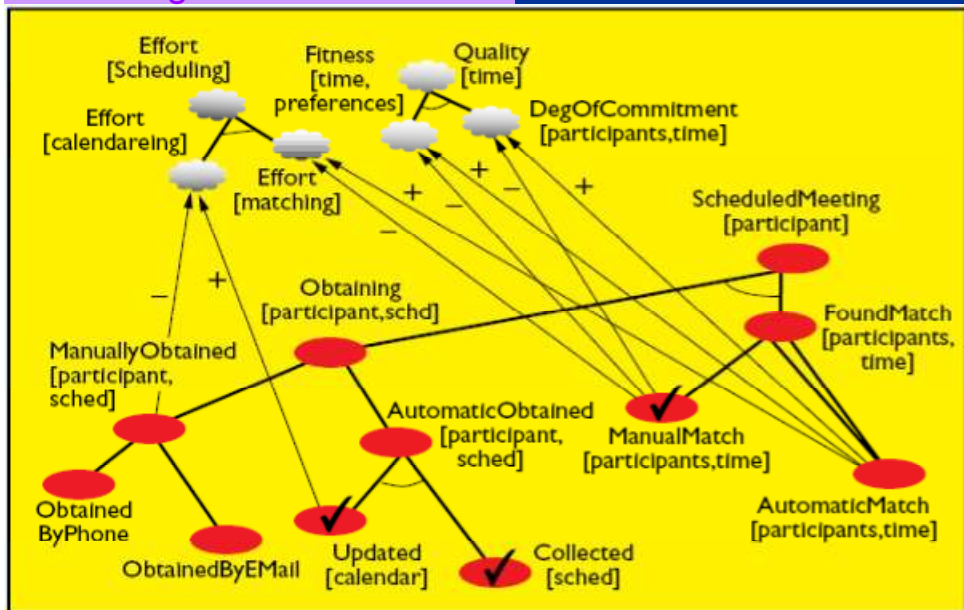
Softgoal Interdependency Graph (SIG)



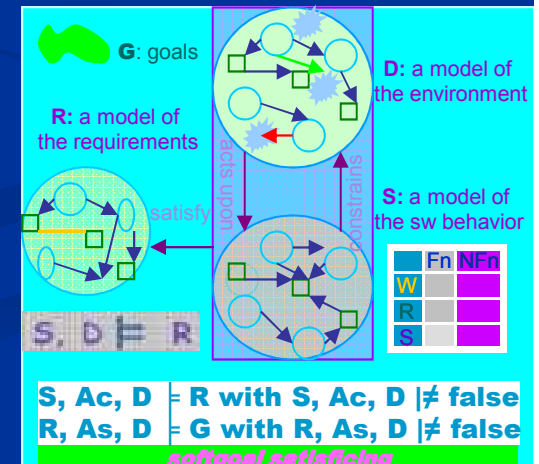
Object Model



From Softgoals to Use Cases



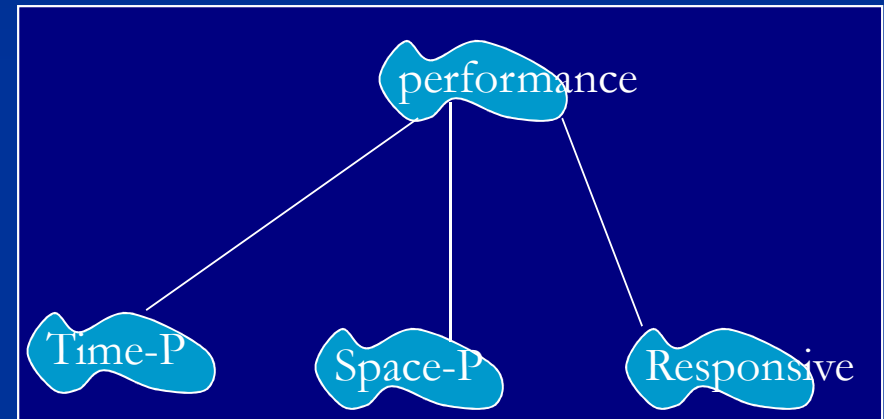
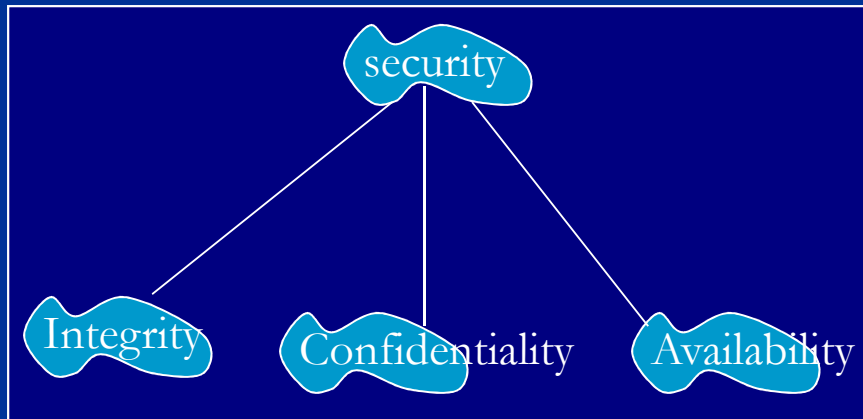
Chung



NFRs:

subjective in both *definitions* & solutions

- Know at least what you mean - *decompose*

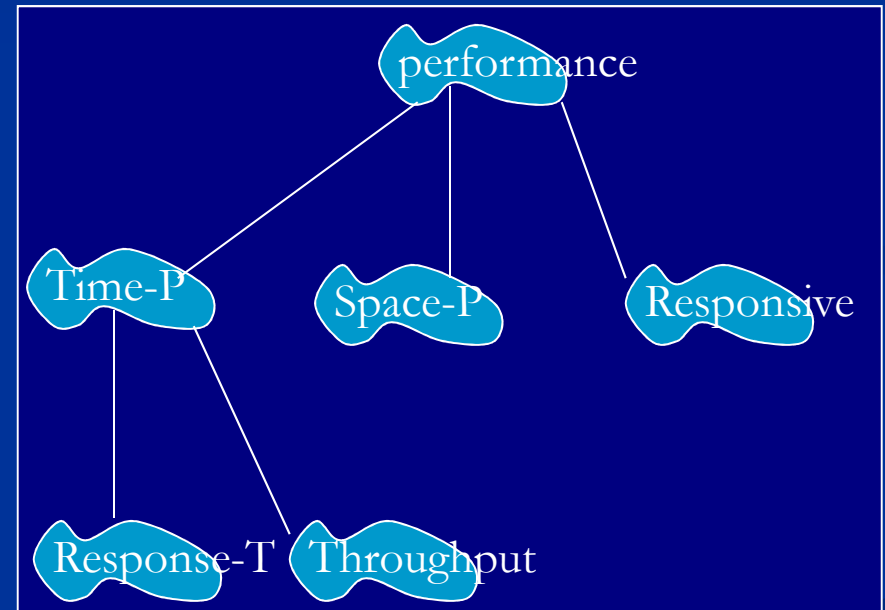
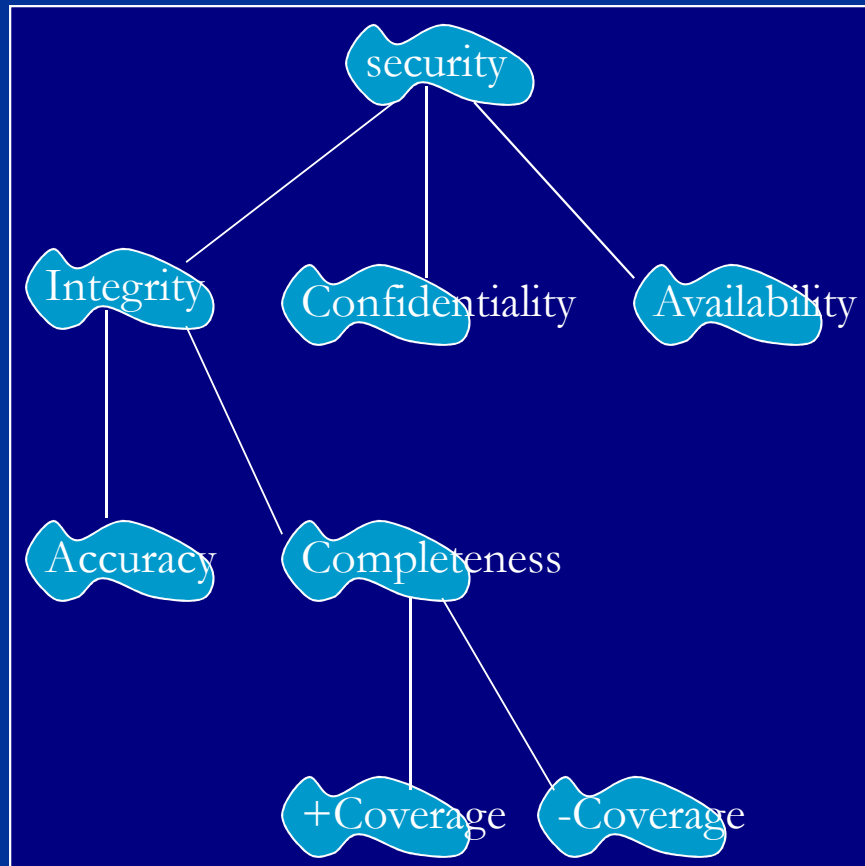


NFRs:

subjective in both *definitions* & solutions

➤ Know at least what you mean *as precisely as possible*

- *as many decompositions as needed*

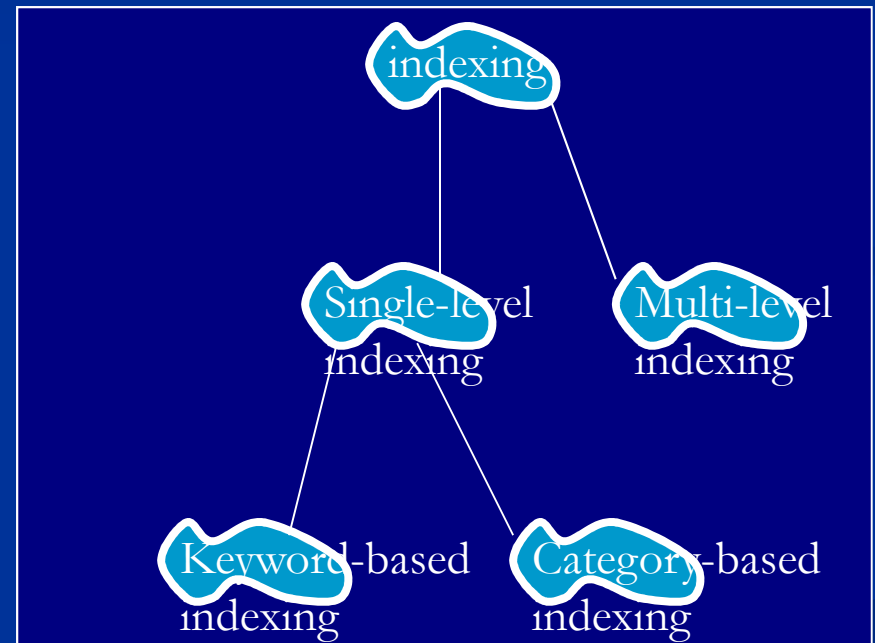
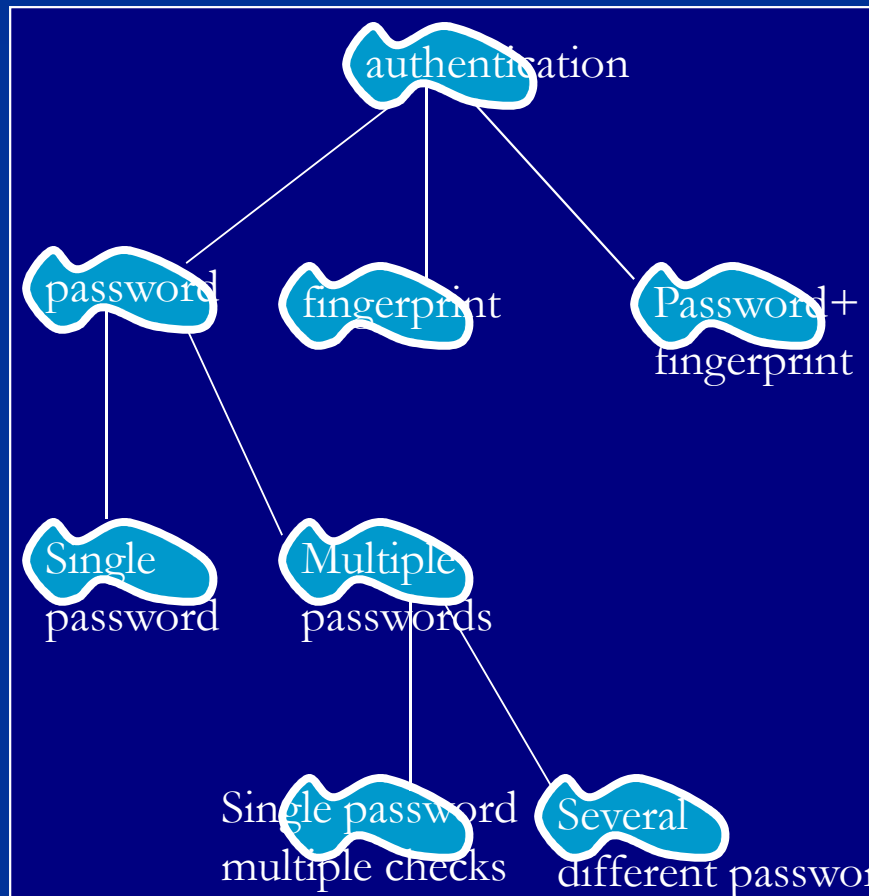


NFRs:

subjective in both definitions & *solutions*

➤ Know at least what you mean *as precisely as possible*

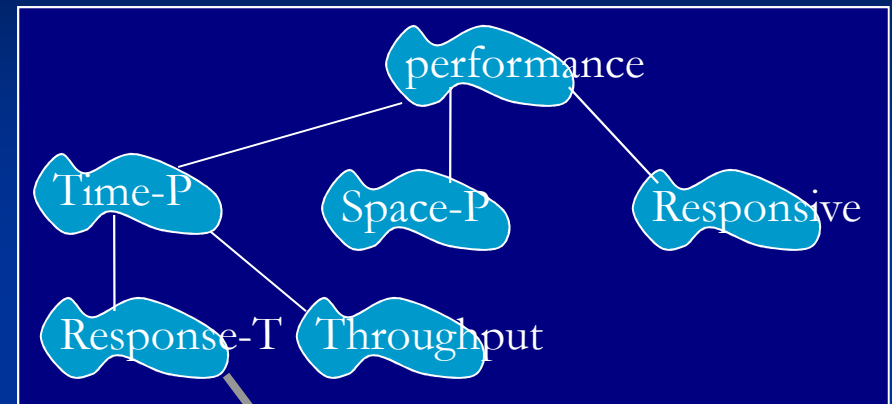
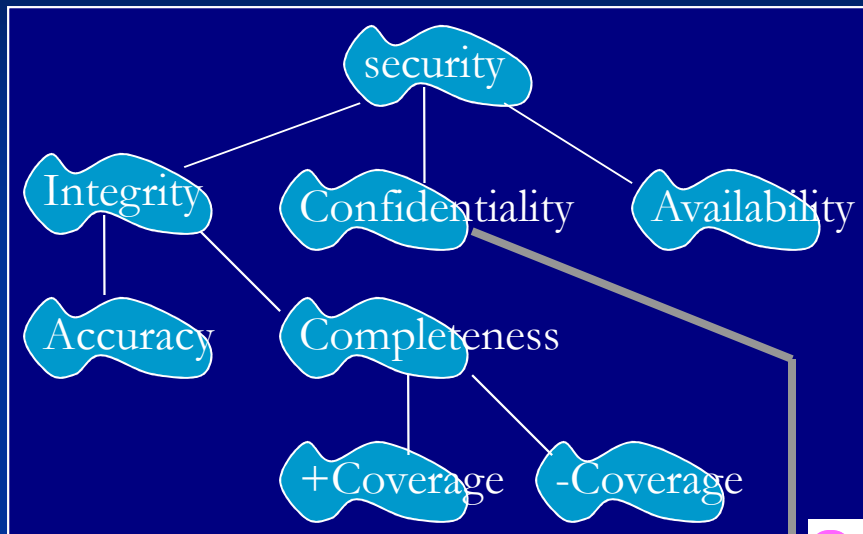
- *as many decompositions as needed*



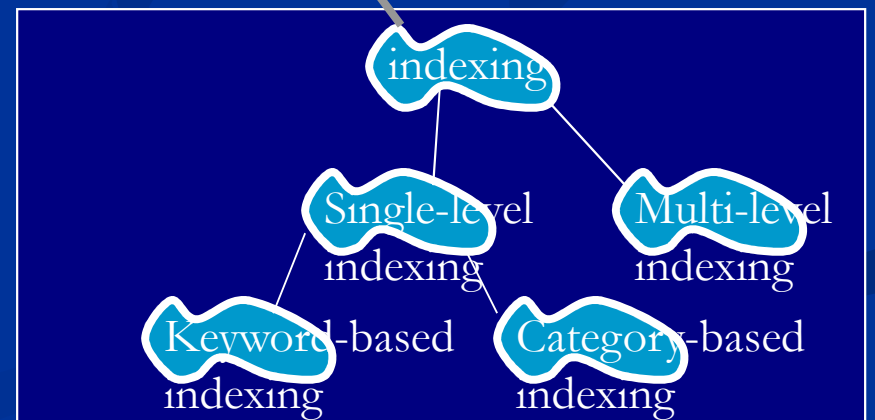
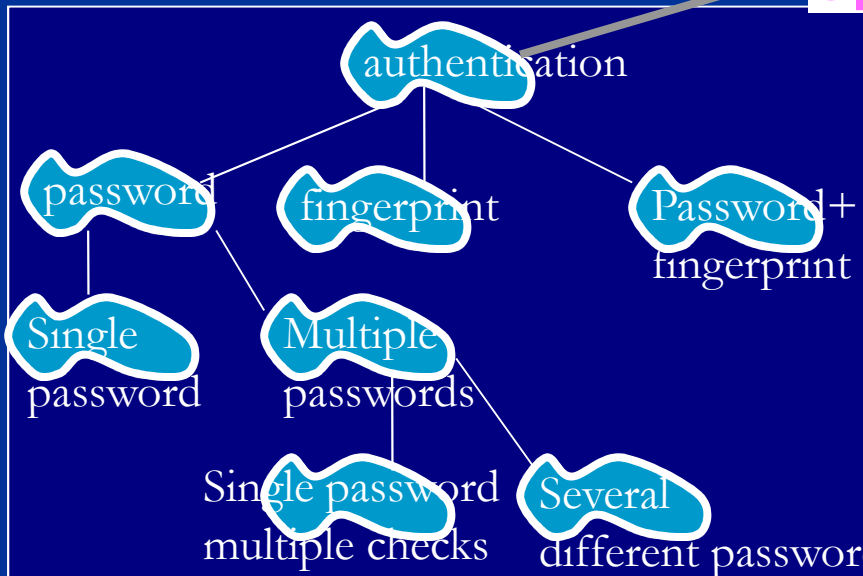
NFRs:

subjective in both *definitions* & *solutions*

- Know at least what you mean *as precisely as possible* - *as many decompositions as needed*

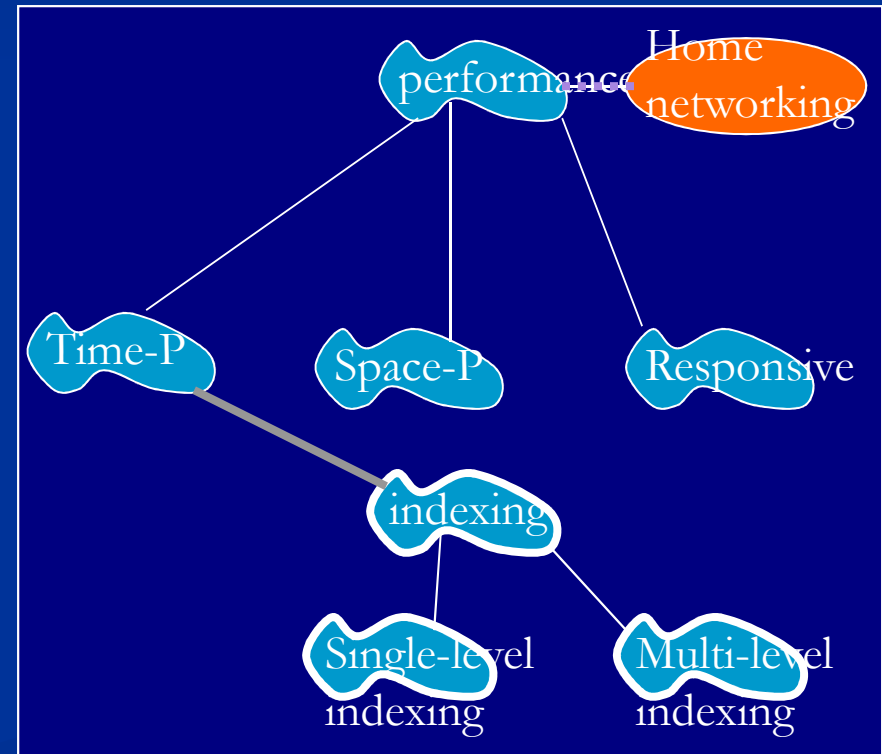
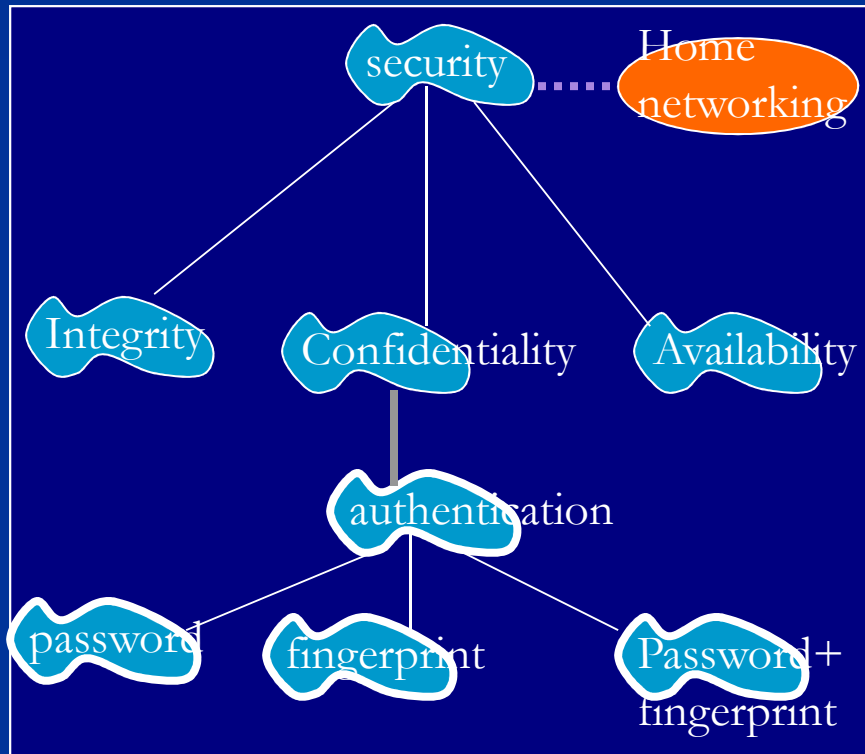


Operationalize



NFRs: non-functional ...and...functional

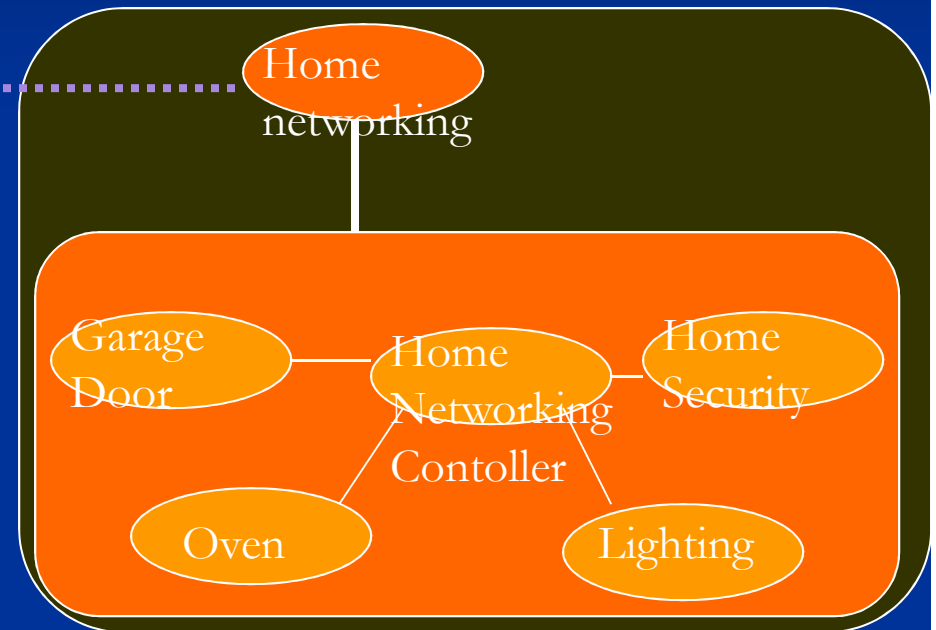
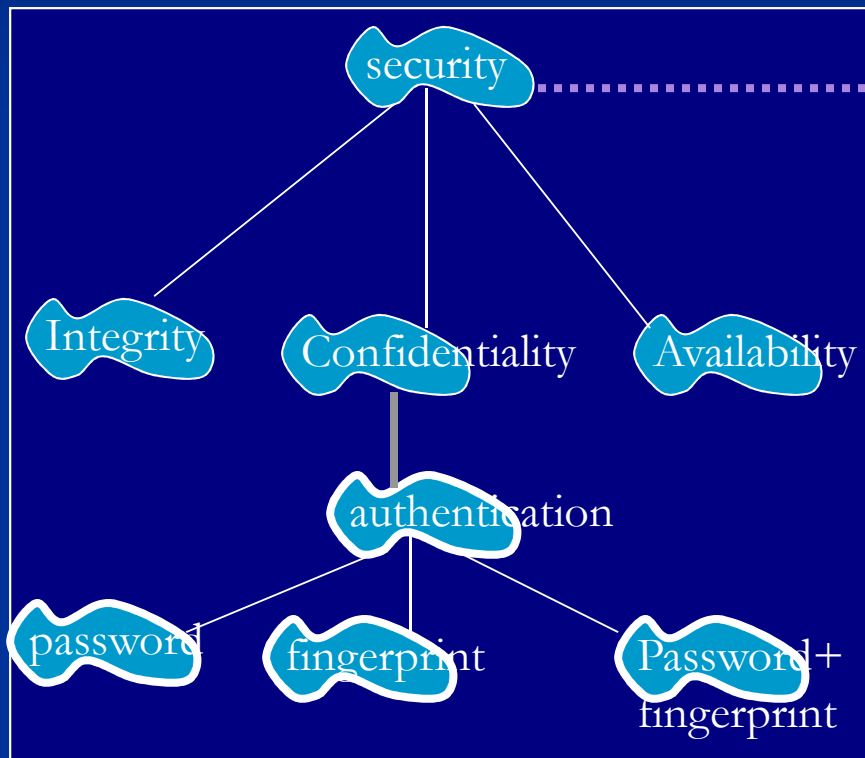
- Know at least what you mean – decompose
- Relate **Functional** and **Non-functional** sides



NFRs:

non-functional ...and...functional

- Know at least what you mean – decompose
- Relate Functional and Non-functional sides
- **Be as specific about the scope/topic/parameter: from global to local**



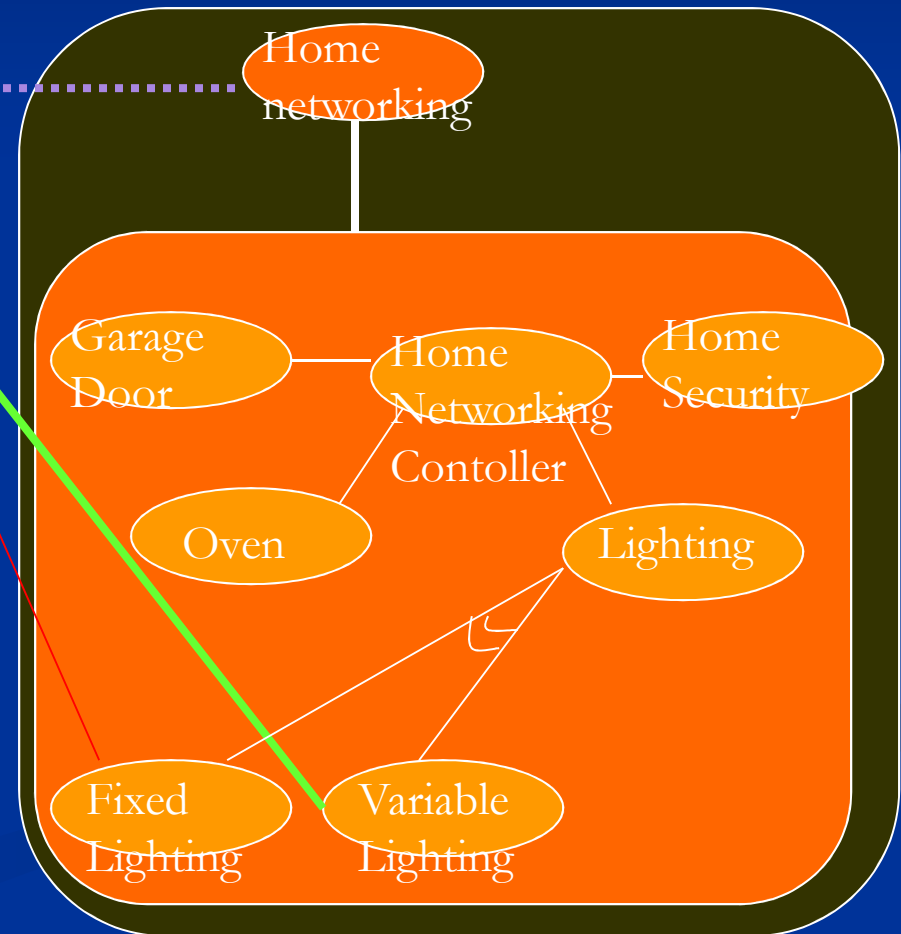
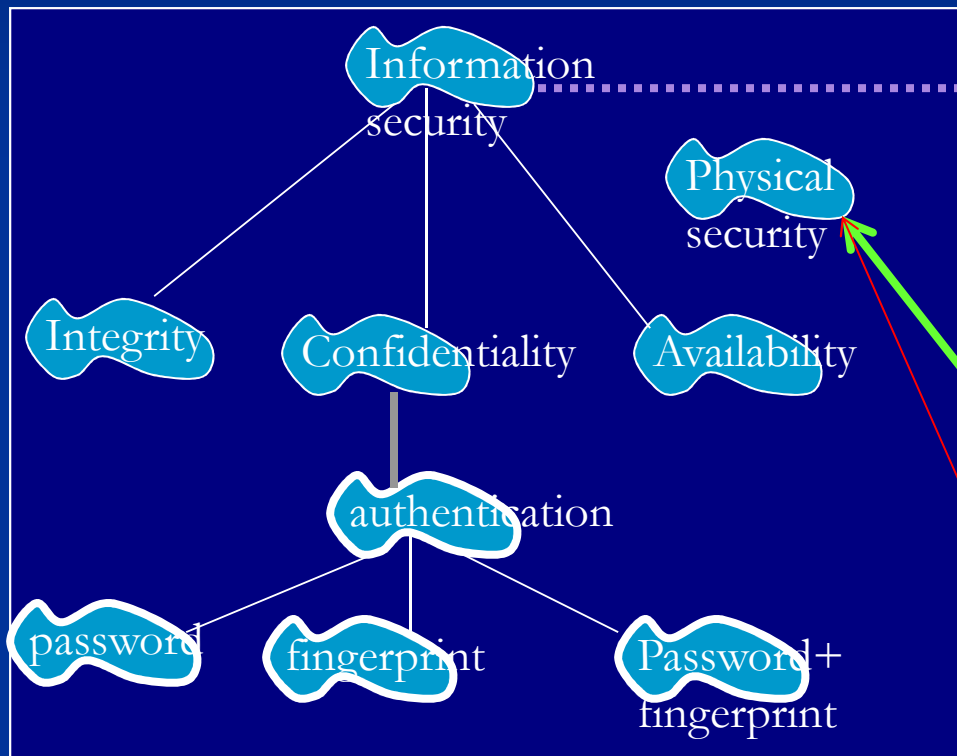
- Security \Rightarrow Security [Home Networking] \Rightarrow Security [Garage Door, Home Networking]
- Authentication \Rightarrow authentication [Home Networking] \Rightarrow authentication [Garage Door, Home Networking]

NFRs:

non-functional ...and...

subjective in both definitions & solutions

- Know at least what you mean – decompose
- Relate Functional and Non-functional sides
- **Different functional operationalizations contribute differently**



Make >> Help >> Hurt >> Break

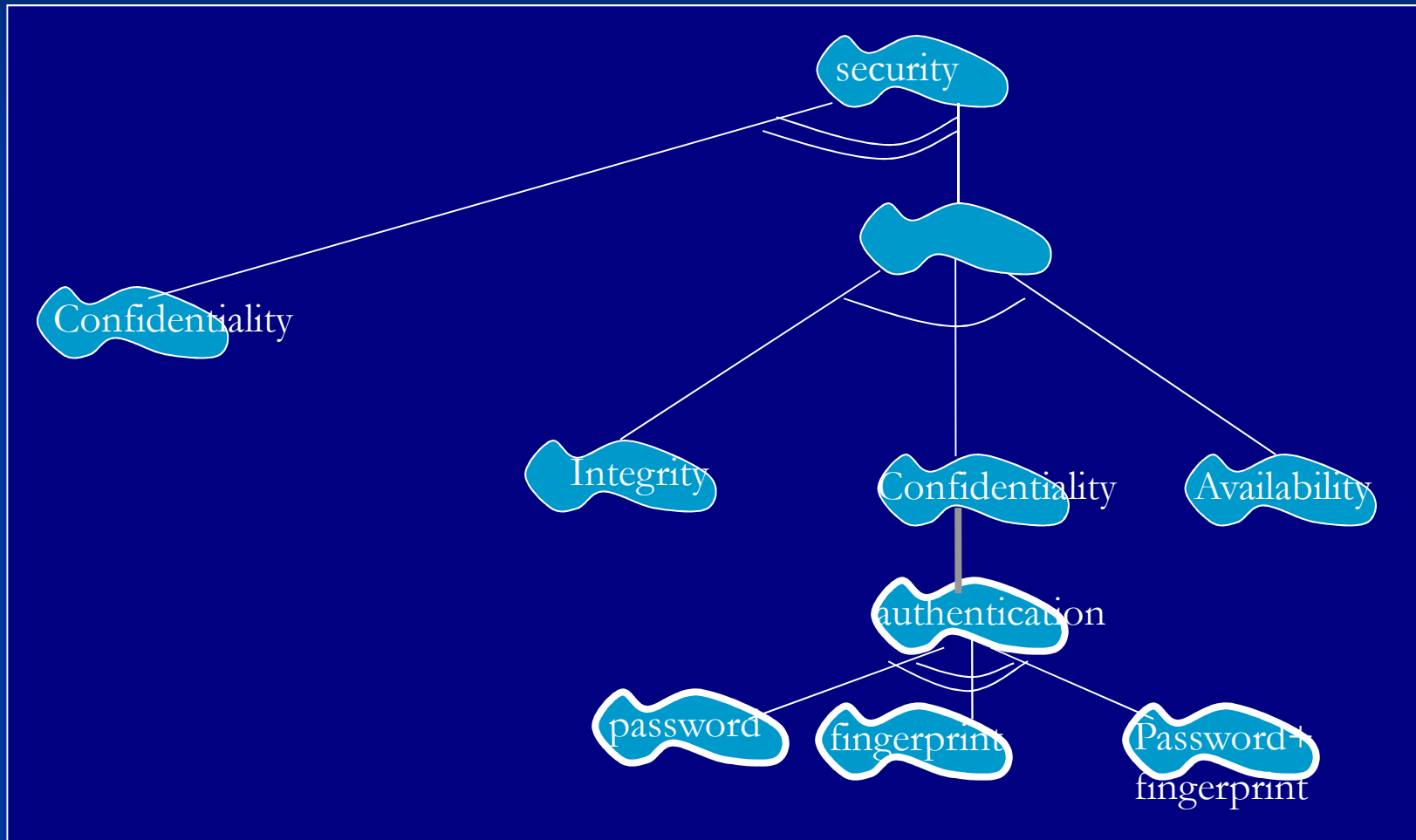
"Satisficing" (cf. Nilsson's)

Lawrence Chung

NFRs:

graded in both definitions and solutions – and relative

- **Explore alternatives** – some are better/worse than others



NFRs:

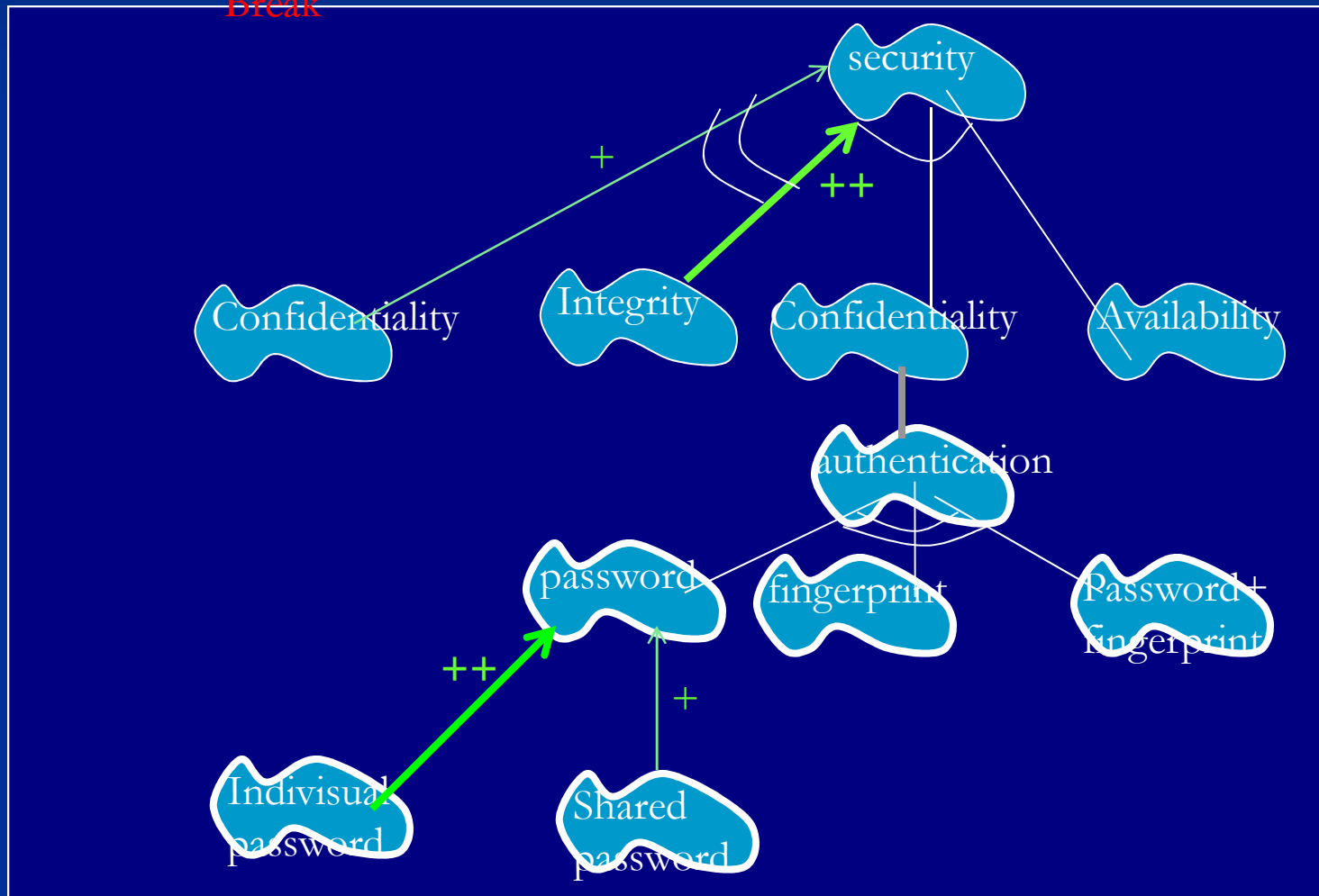
graded in both definitions and solutions – and relative

- Explore alternatives – some are better/worse than others
- **Different alternatives may have different degrees of contributions**

Make >> Help >> Hurt >>

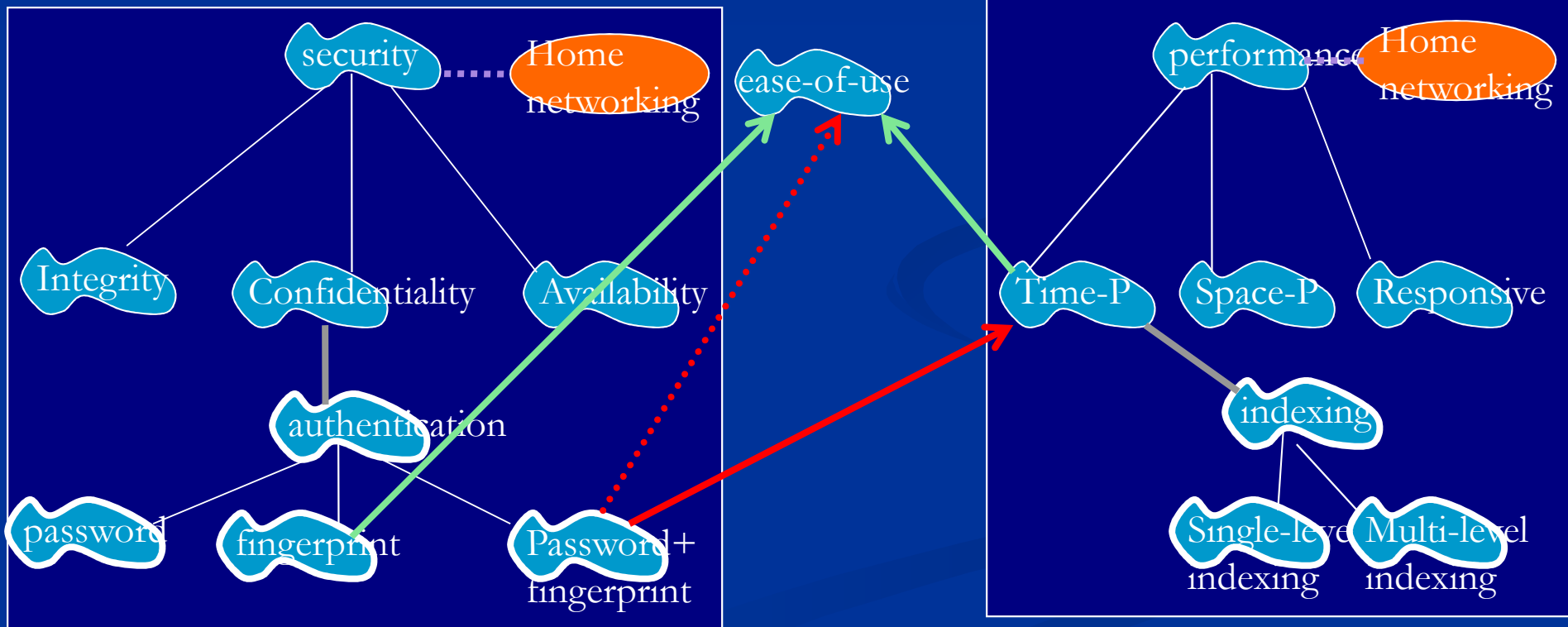
“Satisficing” (cf. Nilsson’s)

Break



NFRs: interacting

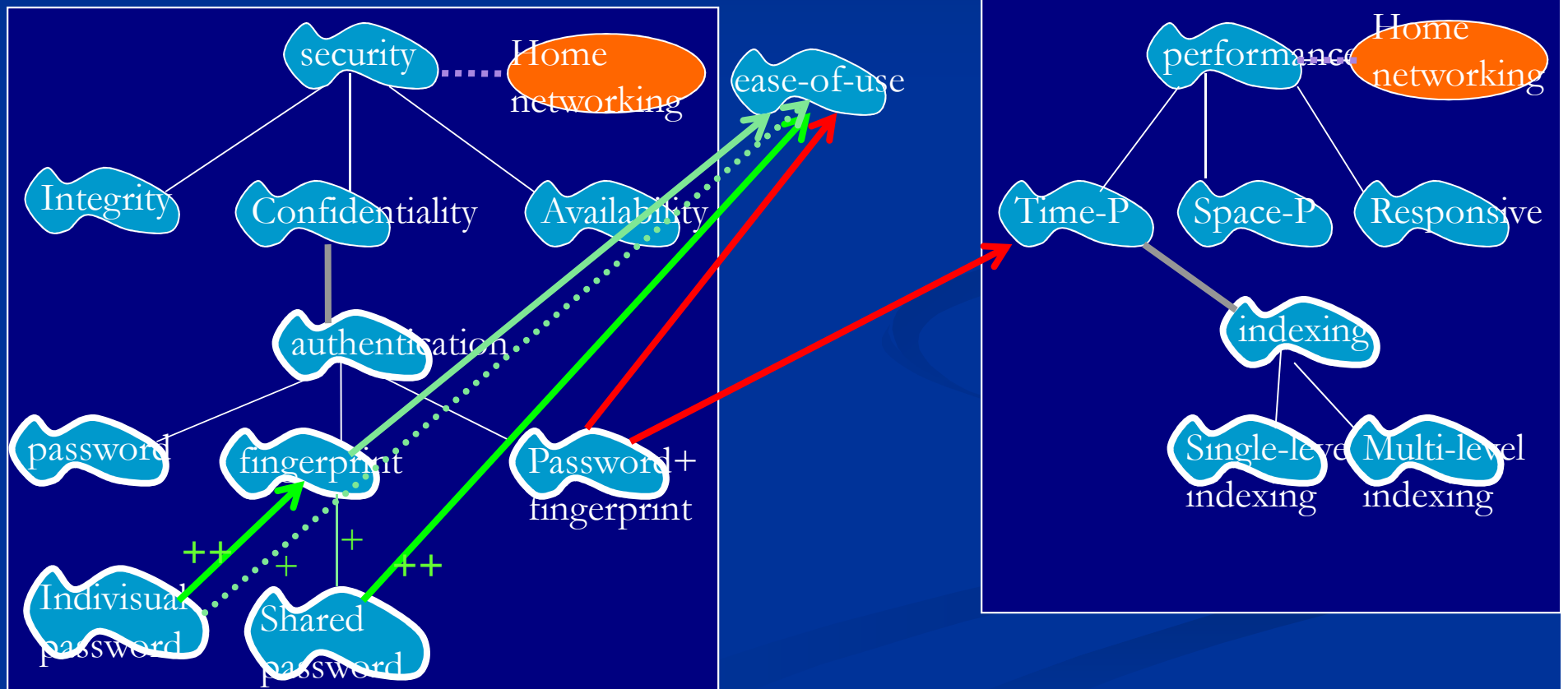
- Conflicting: the whole is less than the sum of its parts
- Synergistic: the whole is greater than the sum of its parts



NFRs:

interacting – graded/relative

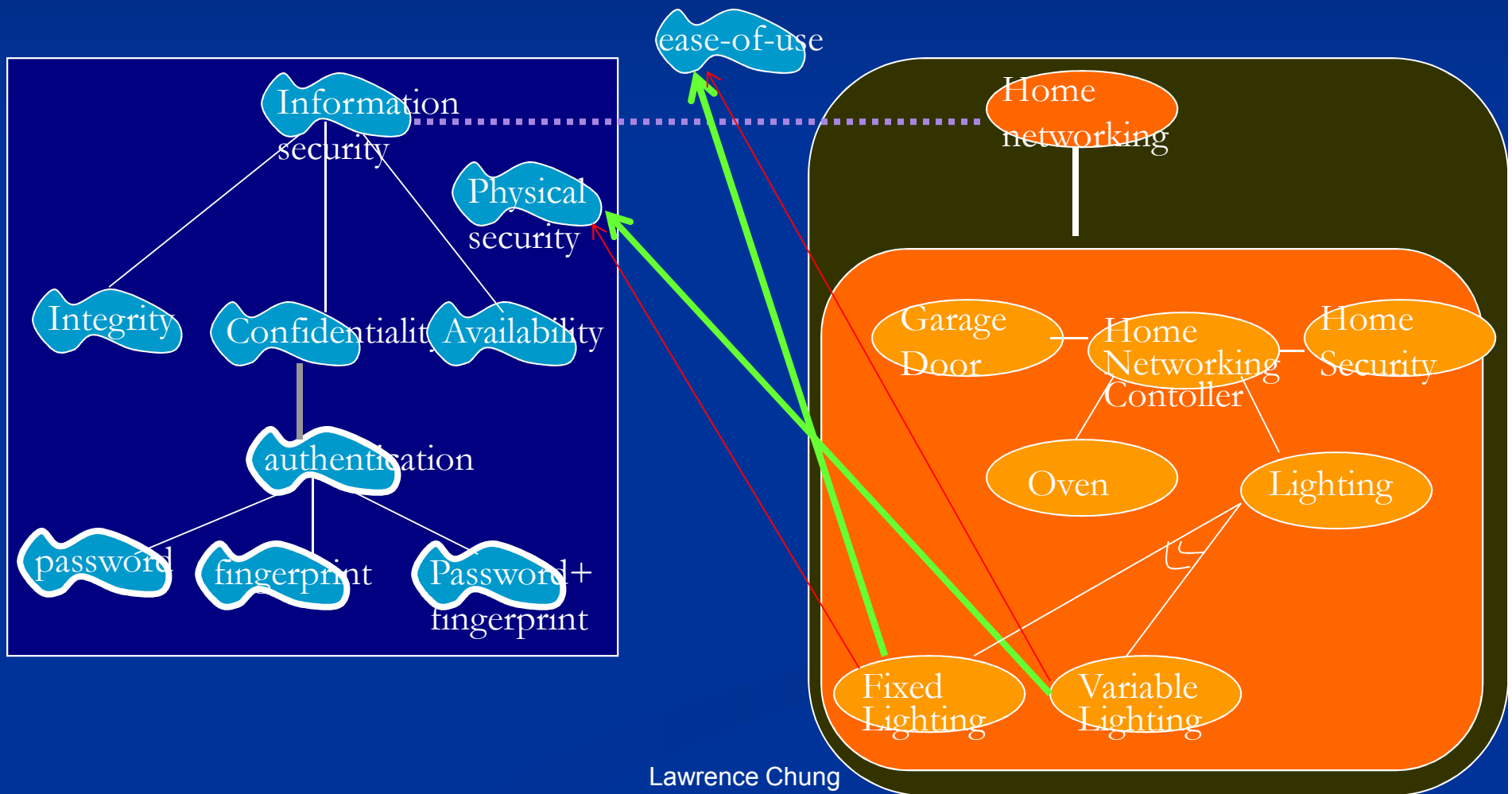
- Different techniques thru *nfr-operationalizations* have different impacts (cf. fr-operationalizations)



NFRs:

interacting – graded and relative

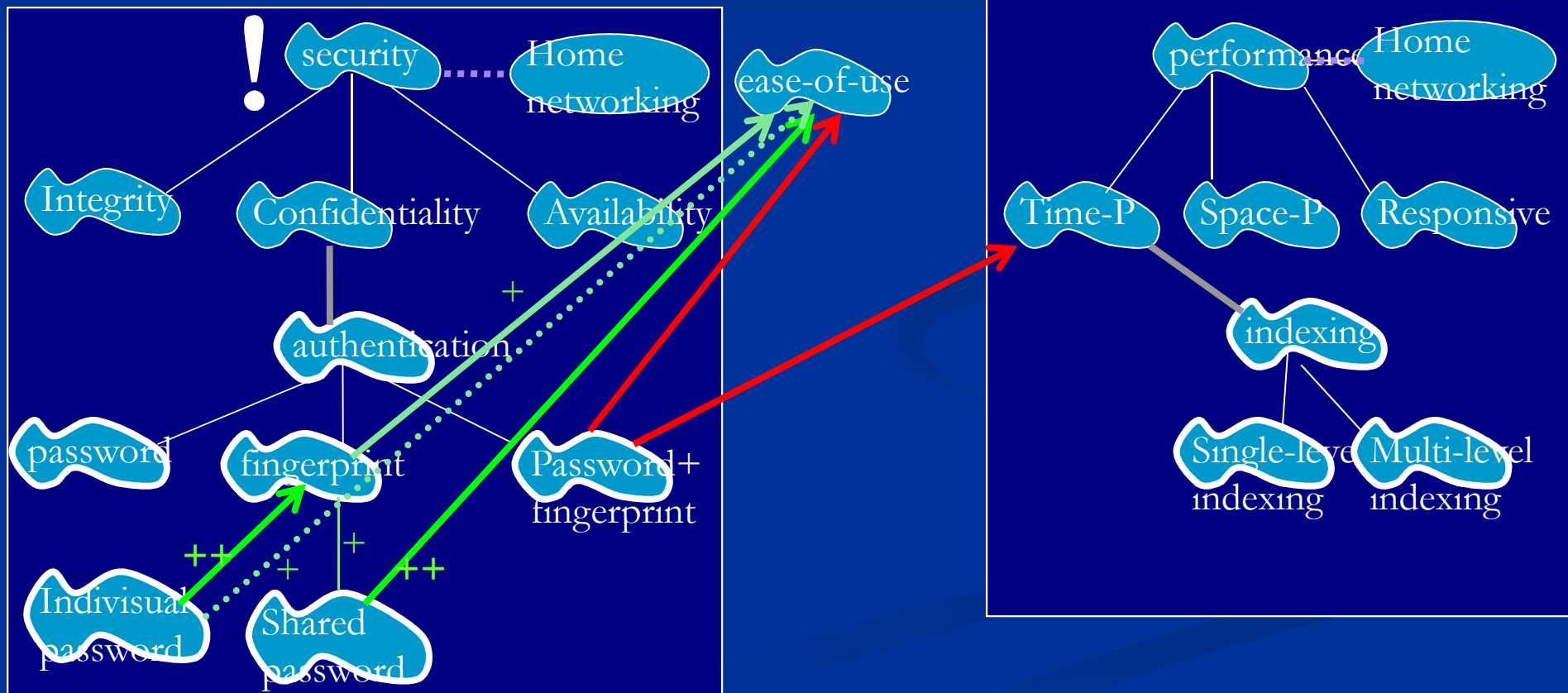
- Through functional choices (*fr-operationalizations*)



NFRs:

interacting – graded/relative

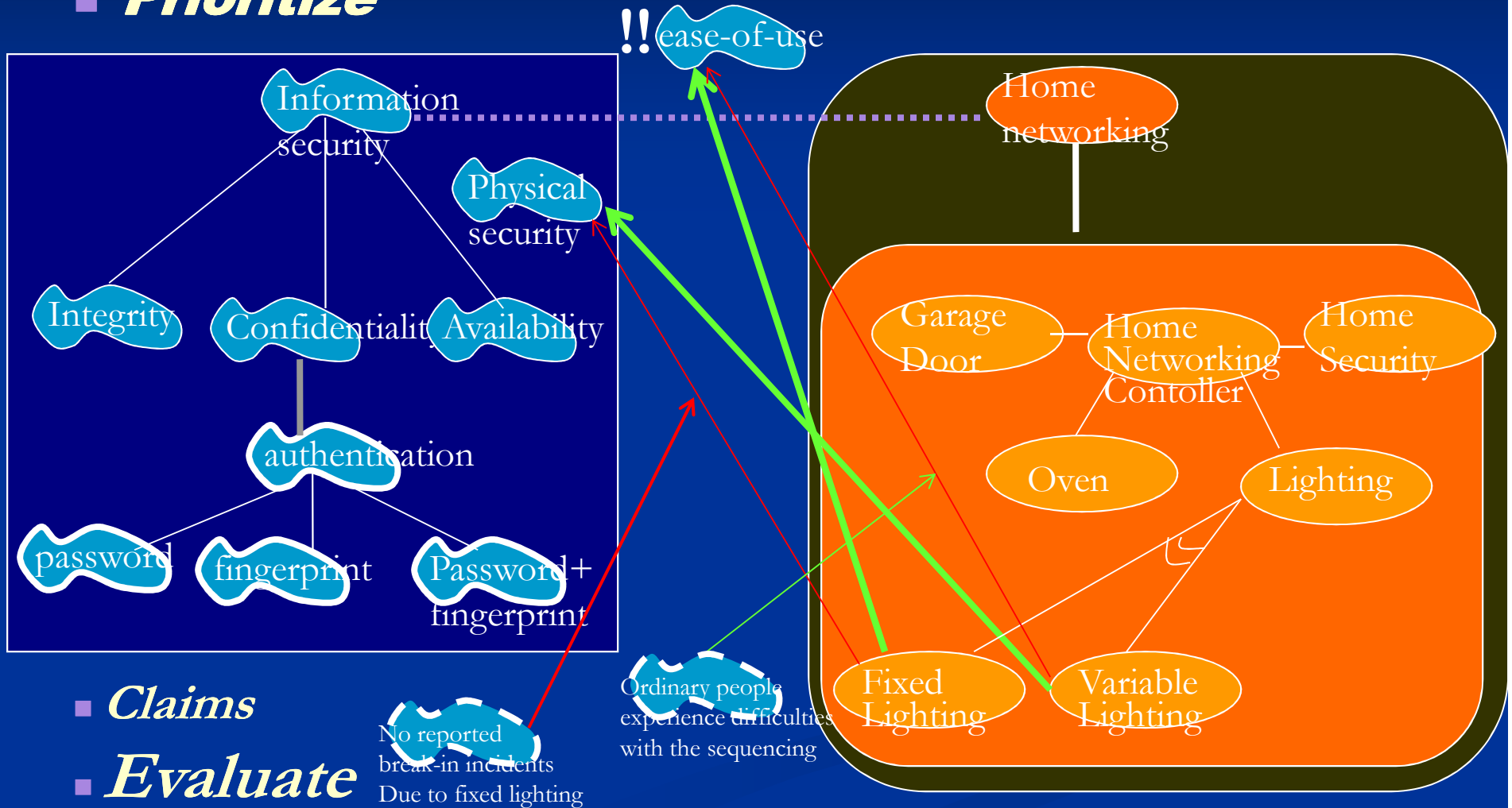
- Different techniques have different impacts
- **Prioritize**



NFRs:

interacting – graded and relative

- Through functional choices
- **Prioritize**



- **Claims**

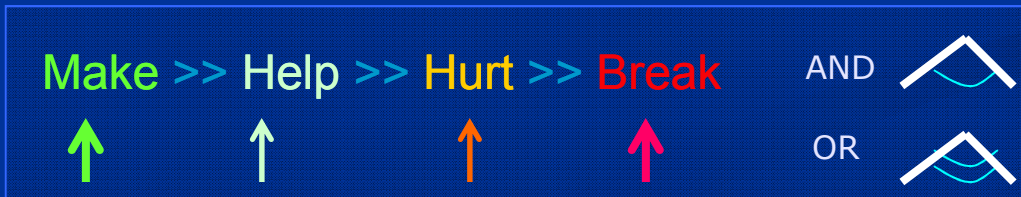
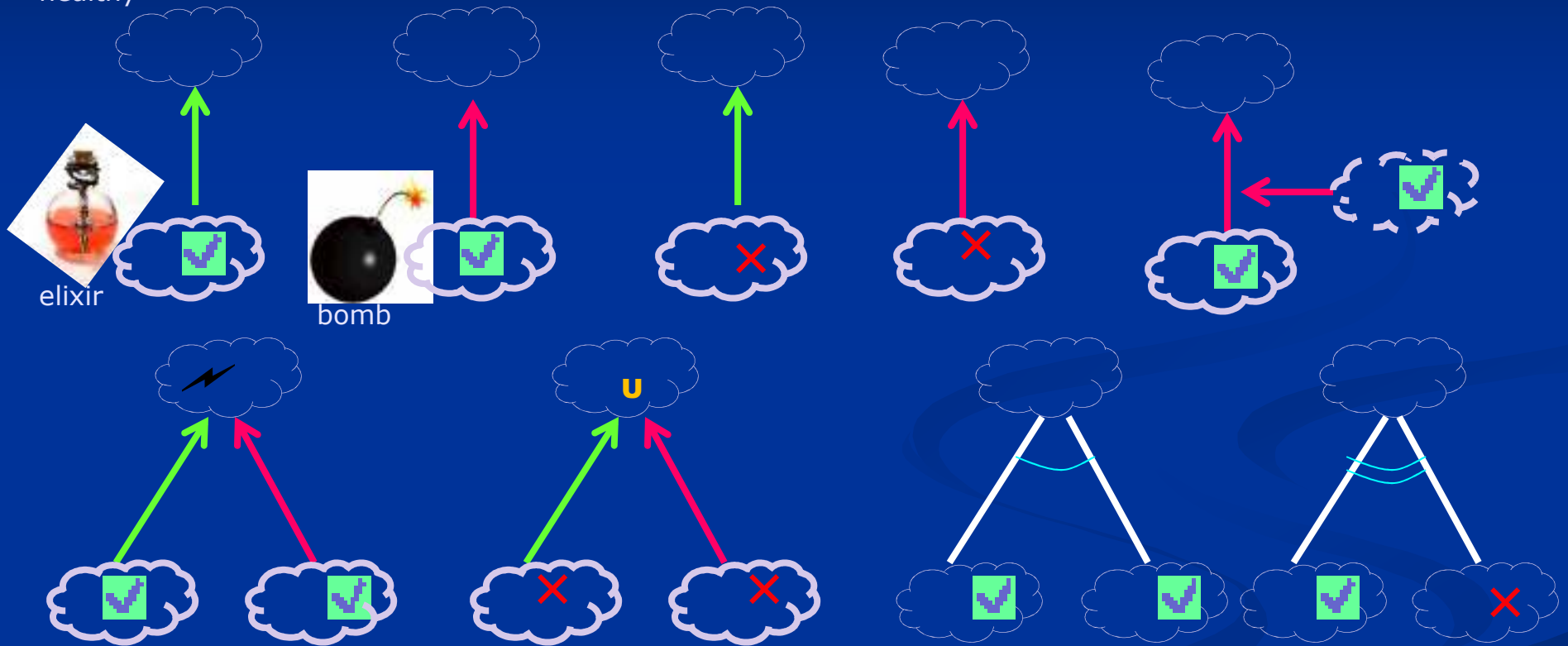
- **Evaluate**

thru propagation of labels (satisfied, denied)

Softgoal Interdependency Graph (SIG): Evaluation Thru Label Propagation

Quiz:

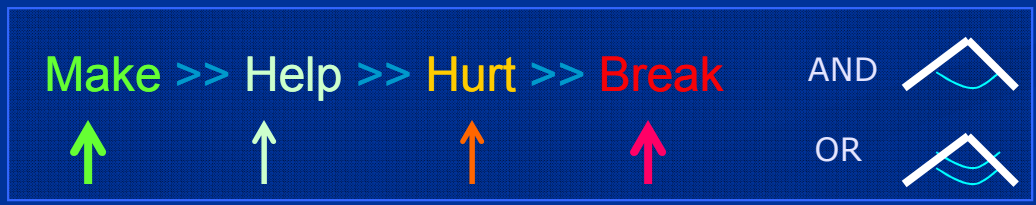
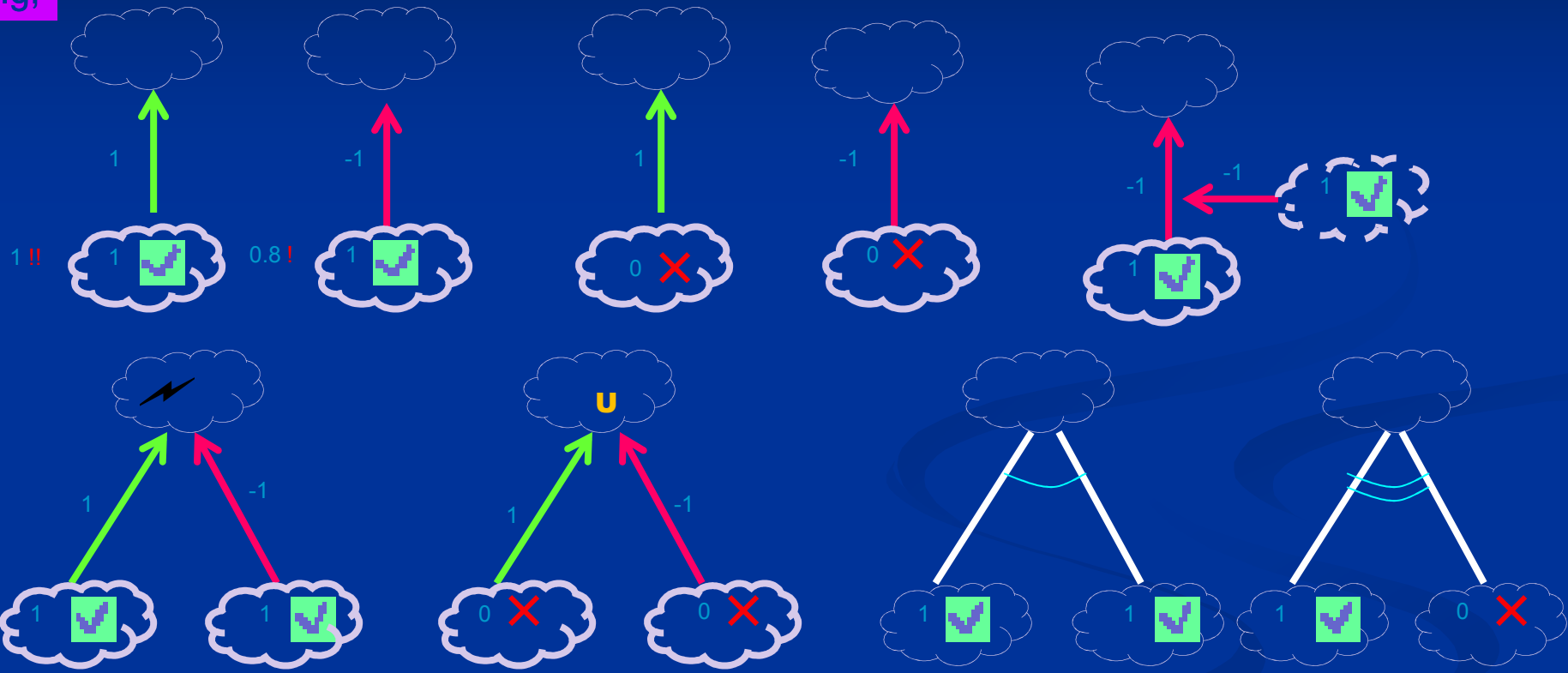
healthy



Softgoal Interdependency Graph (SIG): Evaluation Thru Label Propagation

Quantitative evaluation as a function of numeric values for labels and priorities.

e.g,



Softgoal Interdependency Graph (SIG): Summary of Modeling Concepts

- Softgoals: NFR Softgoals, Operationalizing Softgoals, Claim Softgoals

Integrity

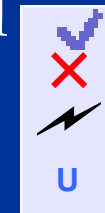
password

No reported
break-in incidents
due to fixed lighting

Softgoals ::= Priority Type [topic list] Label

!

Garage
Door



- Contributions:

□ Make >> Help >> Hurt >> Break



AND

OR

□ “Satisficing”

Softgoal Interdependency Graph (SIG): Semantics

- Proposition = Softgoal U Contribution

AND : *Propositions* $\times 2^{\text{Propositions}}$.

 $\text{satisfied}(G_1) \wedge \text{satisfied}(G_2) \wedge \dots \wedge \text{satisfied}(G_n) \wedge$
 $\text{satisfied}(\text{AND}(G_0, \{G_1, G_2, \dots, G_n\})) \quad \longrightarrow \text{satisficible}(G_0)$

$(\text{denied}(G_1) \vee \text{denied}(G_2) \vee \dots \vee \text{denied}(G_n)) \wedge$
 $\text{satisfied}(\text{AND}(G_0, \{G_1, G_2, \dots, G_n\})) \quad \longrightarrow \text{deniable}(G_0)$

OR : *Propositions* $\times 2^{\text{Propositions}}$.

$\text{denied}(G_1) \wedge \text{denied}(G_2) \wedge \dots \wedge \text{denied}(G_n) \wedge$
 $\text{satisfied}(\text{OR}(G_0, \{G_1, G_2, \dots, G_n\})) \quad \longrightarrow \text{deniable}(G_0)$

$(\text{satisfied}(G_1) \vee \text{satisfied}(G_2) \vee \dots \vee \text{satisfied}(G_n)) \wedge$
 $\text{satisfied}(\text{OR}(G_0, \{G_1, G_2, \dots, G_n\})) \quad \longrightarrow \text{satisficible}(G_0)$

Softgoal Interdependency Graph (SIG): Semantics

MAKE *Propositions* × *Propositions*.

$$\text{satisfied}(G_1) \wedge \text{satisfied}(\text{MAKE}(G_0, G_1)) \longrightarrow \text{satisficable}(G_0)$$

HELP *Propositions* × *Propositions*.

$$\text{denied}(G_1) \wedge \text{satisfied}(\text{HELP}(G_0, G_1)) \longrightarrow \text{deniable}(G_0)$$

If $\text{satisfied}(\text{HELP}(G_0, G_1))$ then there exist propositions G_2, \dots, G_n such that $\neg(\text{satisfied}(G_2) \wedge \dots \wedge \text{satisfied}(G_n) \longrightarrow \text{satisficable}(G_0))$

but

$$\text{satisfied}(G_1) \wedge \text{satisfied}(G_2) \wedge \dots \wedge \text{satisfied}(G_n) \wedge \text{satisfied}(\text{HELP}(G_0, G_1)) \longrightarrow \text{satisficable}(G_0)$$

BREAK *Propositions* × *Propositions*.

$$\text{satisfied}(G_1) \wedge \text{satisfied}(\text{BREAK}(G_0, G_1)) \longrightarrow \text{deniable}(G_0)$$

HURT *Propositions* × *Propositions*.

$$\text{denied}(G_1) \wedge \text{satisfied}(\text{HURT}(G_0, G_1)) \longrightarrow \text{satisficable}(G_0)$$

If $\text{HURT}(G_0, G_1)$ then there exist G_2, \dots, G_n such that

$$\neg(\text{satisfied}(G_2) \wedge \dots \wedge \text{satisfied}(G_n) \wedge \text{satisfied}(\text{HURT}(G_0, G_1)) \longrightarrow \text{deniable}(G_0))$$

but

$$\text{satisfied}(G_1) \wedge \text{satisfied}(G_2) \wedge \dots \wedge \text{satisfied}(G_n) \wedge \text{satisfied}(\text{HURT}(G_0, G_1)) \longrightarrow \text{deniable}(G_0)$$

und : *Propositions* × *Propositions*.

$$\text{Und}(G_0, G_1) = \text{MAKE}(G_0, G_1) \vee \text{HELP}(G_0, G_1) \vee \text{HURT}(G_0, G_1) \vee \text{BREAK}(G_0, G_1)$$

eql : *Propositions* × *Propositions*.

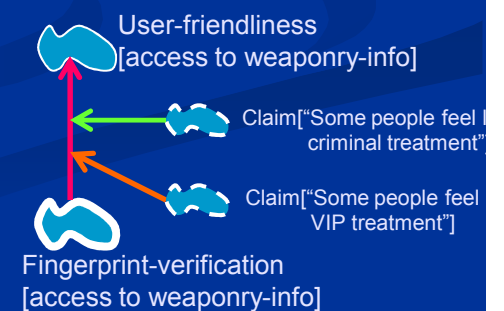
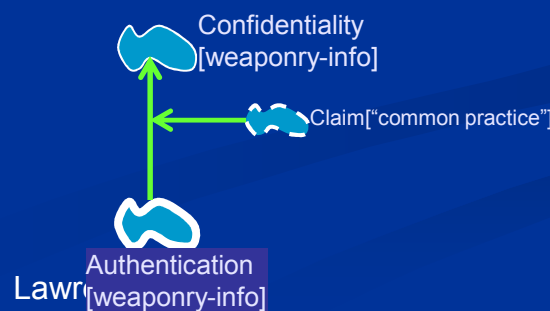
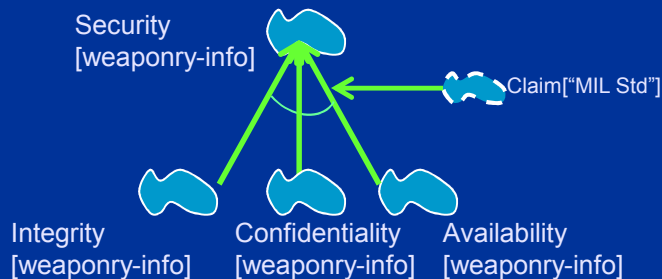
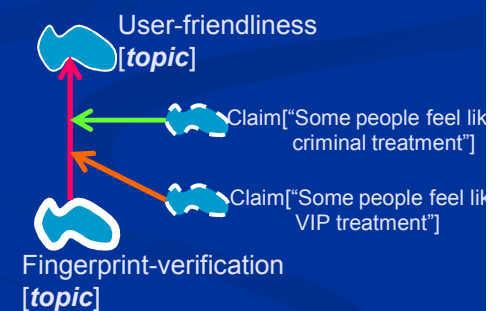
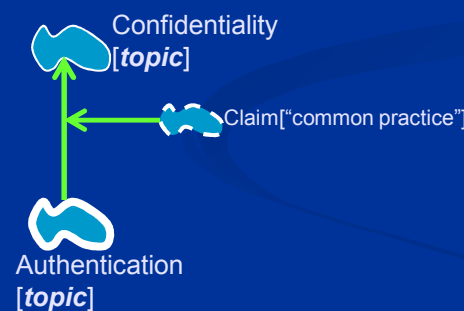
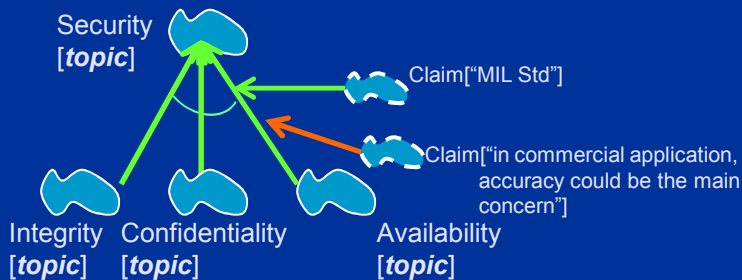
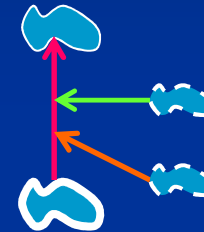
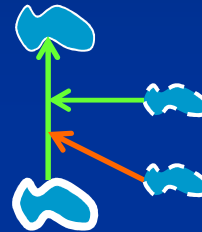
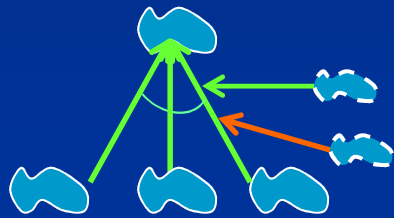
Softgoal Interdependency Graph (SIG): Process of Construction

An iterative, interleaving process!!!

- Post NFR Softgoals:
- Refine NFR Softgoals as many times until the meaning is clear
 - Refine the type
 - Refine the topic list
 - Refine the priority
- Operationalize NFR Softgoals
- Refine Operationalizing Softgoals as many times until all the parts and relationships are designed (N.B: recall “one person’s floor is another person’s ceiling”)
 - Refine the type
 - Refine the topic list
 - Refine the priority
- Provide justifications in terms of Claim Softgoals, for any kind of softgoals
- Evaluate the degree to which each softgoal is satisfied.

The NFR Framework: Reuse of Knowledge of NFRs

- Introduces **Catalogues** of NFRs much like patterns for design are built
 - Methods:
 - Correlation Rules:



The NFR Framework

{Chung et.al.} Non-functional Requirements in Software Engineering [8] [Figure 7.16](#) [Structure notes]

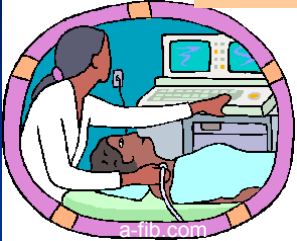
- Patterns:

The NFR Framework

and the Reference Model

Recall:

Example 1: Patient Monitoring



D_1 : There will always be a nurse close enough to hear the buzzer

D_2 : The sound from the heart falling below a certain threshold indicates that heart has (is about to) stop

R_1 : A warning system notifies the nurse if the patient's heartbeat stops

S_1 : If the sound from the sensor falls below a certain threshold, the buzzer shall be actuated

C – with a microphone as a sensor and a buzzer as an actuator

P - Program

Designation Categories:

e_h : the nurse and the heartbeat of the patient.

e_v : sounds from the patient's chest.

s_v : the buzzer at the nurse's station.

s_h : internal representation of data from the sensor.

Recall:

The NFR Framework

and the Reference Model



Example 1: Patient Monitoring

Need: monitoring if a patient's heart is failing

Problem: monitoring if a patient's heart is failing is difficult and sometimes has been unsuccessful

□ A nurse cannot stay close to the patient always and on alert

well qualified and capable *best quality patient care*

D1: There will always be a nurse close enough to hear the buzzer

D2: The sound from the heart falling below a certain threshold indicates that heart has (is about to) stop

R1: A warning system notifies the nurse if the patient's heartbeat (is about to) stop

clear

low-cost, low-maintenance, easily configurable, proven *truly*

best quality

S1: If the sound from the sensor falls below a certain threshold, the buzzer shall be actuated

C – with a microphone as a sensor and a buzzer as an actuator

P - Program

asap

in a safe and secure manner, but loud enough

well qualified and capable

best quality

Designation Categories:

e_h : the nurse and the heartbeat of the patient.

e_v : sounds from the patient's chest.

s_v : the buzzer at the nurse's station.

s_h : internal representation of data from the sensor.

Lawrence Chung

The NFR Framework

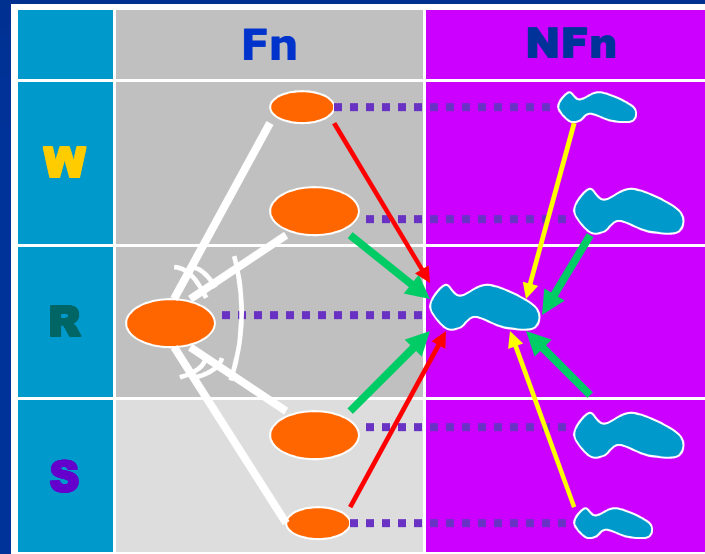
in relation to the Reference Model, KAOS, Tropos

□ **WRSPM:** $S, D \models R$

□ **KAOS:** $S, Ac, D \models R$ with $S, Ac, D \neq \text{false}$
 $R, As, D \models G$ with $R, As, D \neq \text{false}$

□ **The NFR Framework:**

- *nfr-operationalizations*
- *fr-operationalizations*



- *Any phenomena/functional description, indicative or optative or expectational, and any agent can be associated with softgoals*
- $\text{satisfied}(Q(S^G)), \text{satisfied}(Q(D^G)) \models \text{satisfied}(Q(R^G))$
- $\text{satisfied}(Q(P^G)), \text{satisfied}(Q(M^G)) \models \text{satisfied}(Q(S^G))$

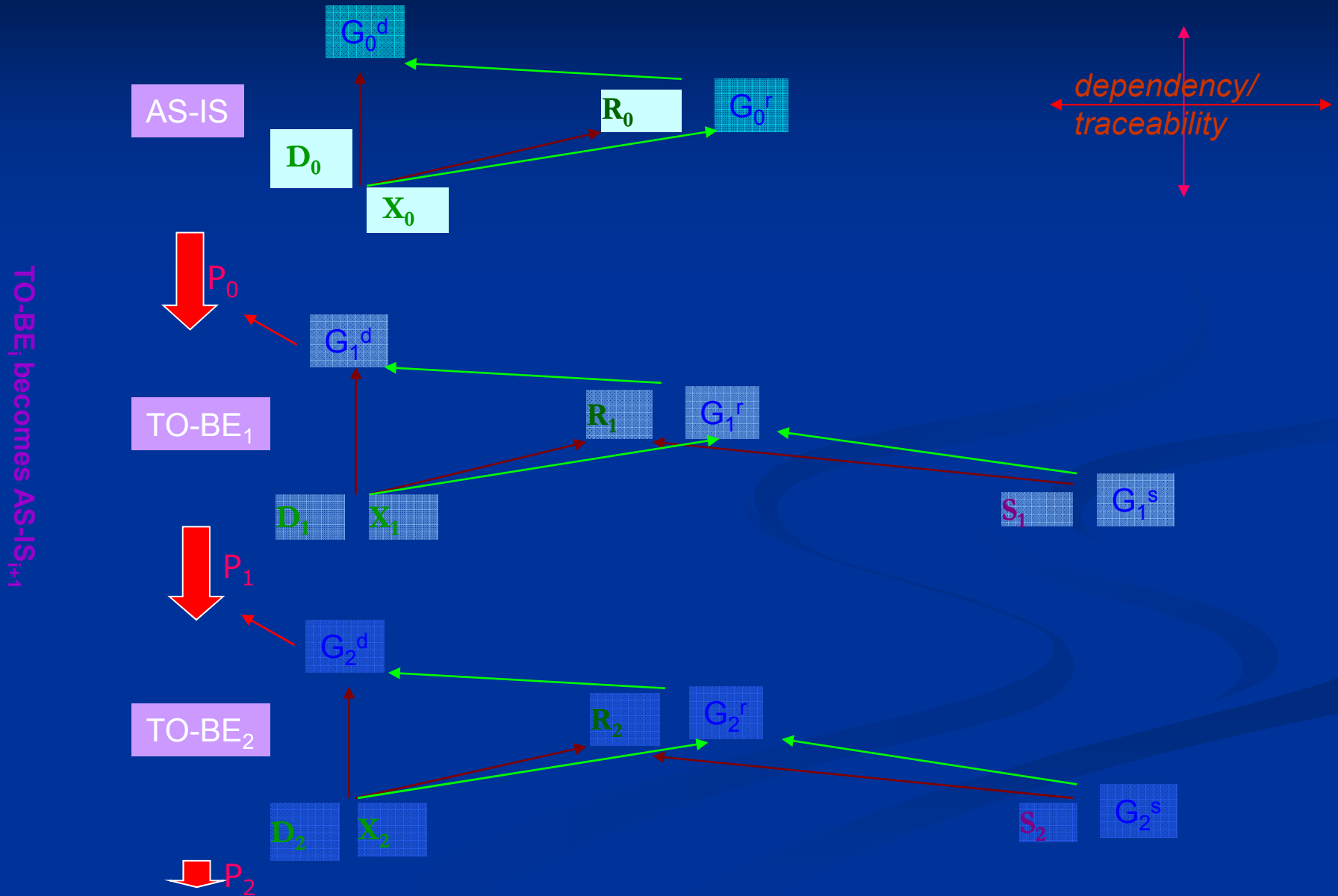
$M^G, \text{Prog}^G \models S^G; S^G, D^G \models R^G; R^G, D^G \models G; (G \models \neg P) \vee (G \sim \neg P)$

What the Metaphysics of Quality would do is take this separate category, Quality, and show how it contains within itself both subjects and objects. The Metaphysics of Quality would show how things become enormously more coherent--fabulously more coherent--when you start with an assumption that Quality is the primary empirical reality of the world.... [Robert Pirsig]

Recall

Property Preserving Evolution

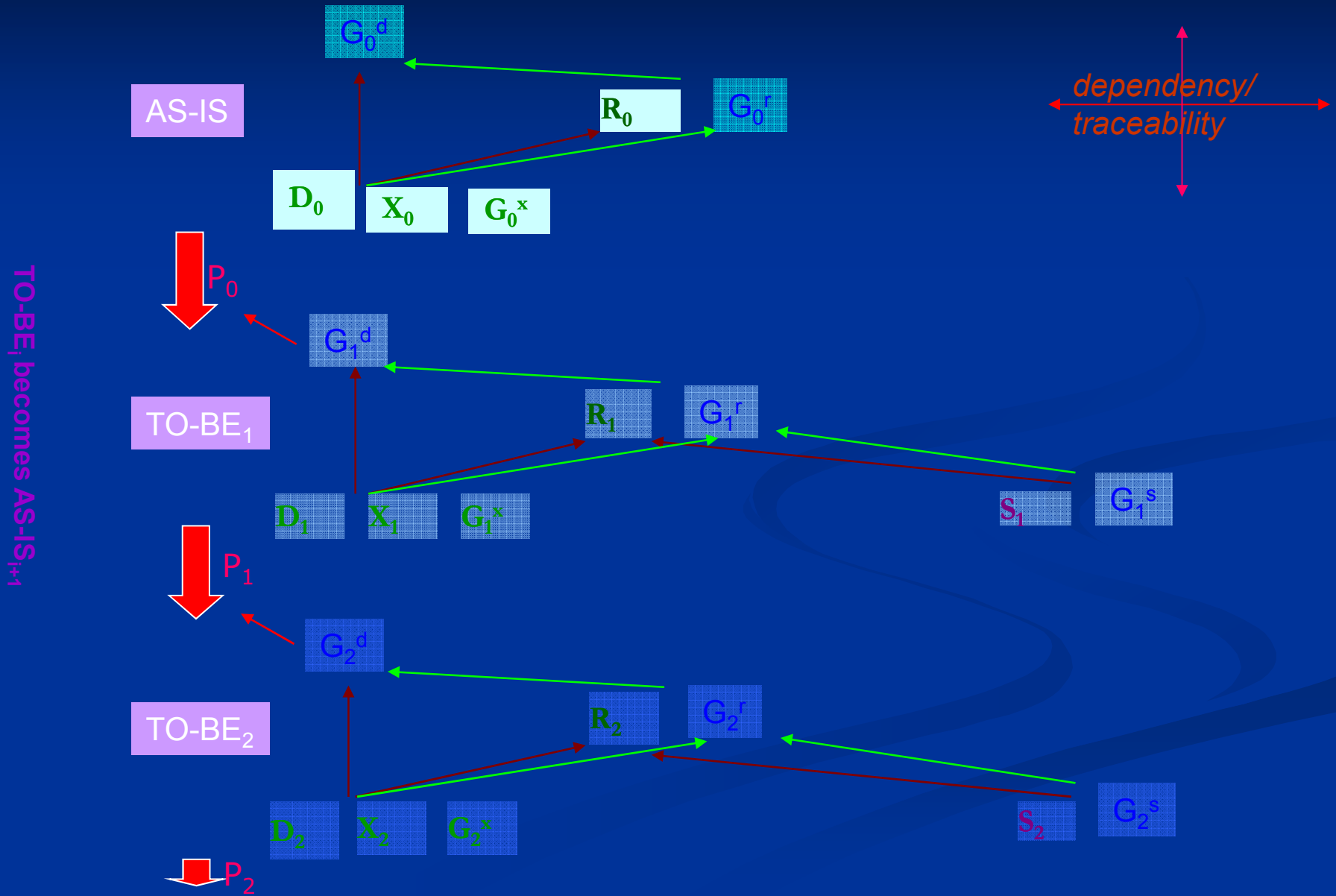
$$(G_i^s, S_i), (D_i, X_i) \models R_i; \quad (G_i^s, S_i), (D_i, X_i) \models G_i^r; \quad (G_i^r, R_i), (D_i, X_i) \models G_i^d$$



Recall

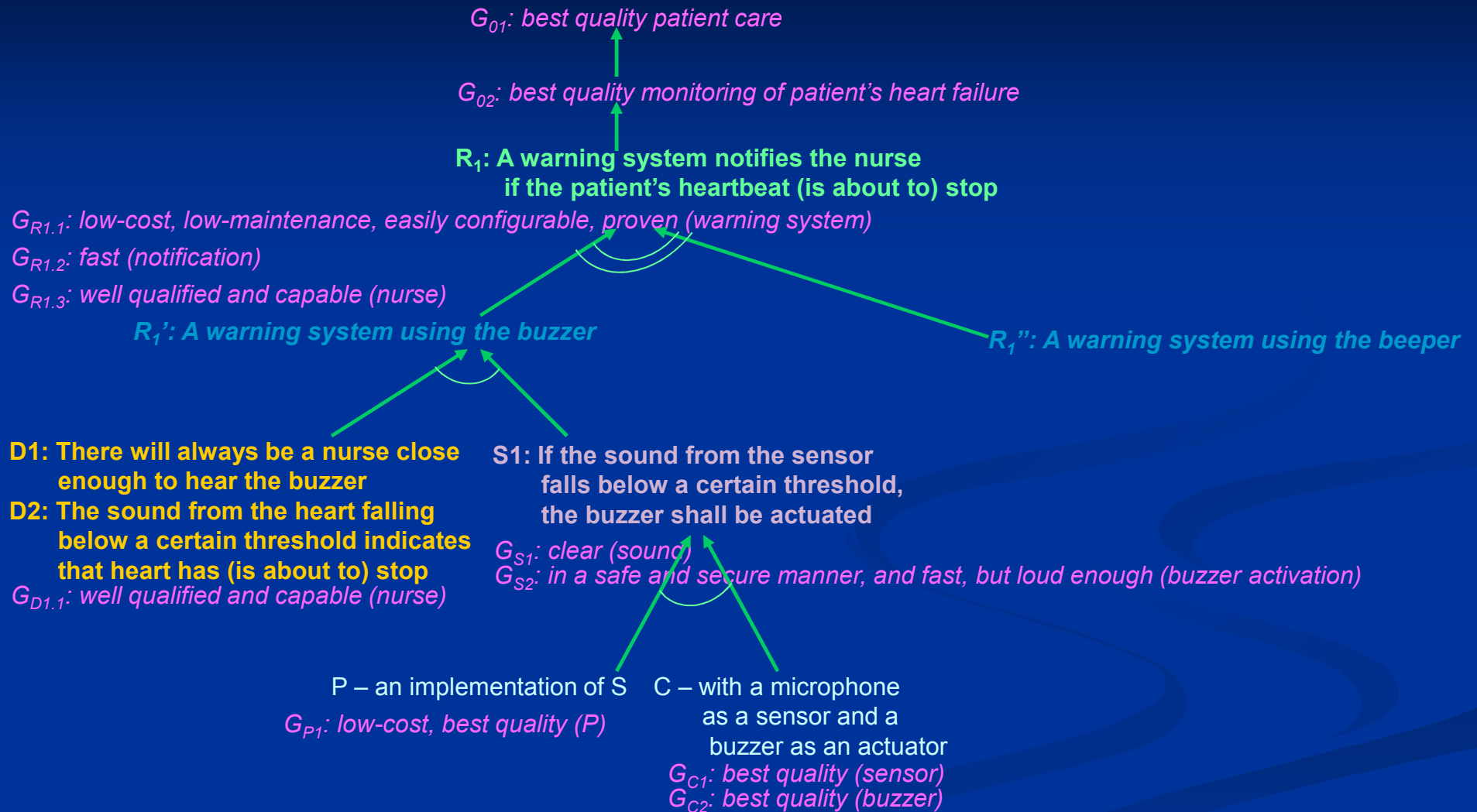
Property Preserving Evolution

$$(G_i^s, S_i), (D_i, X_i, G_i^x) \models R_i; (G_i^s, S_i), (D_i, X_i, G_i^x) \models G_i^r; (G_i^r, R_i), (D_i, X_i, G_i^x) \models G_i^d$$



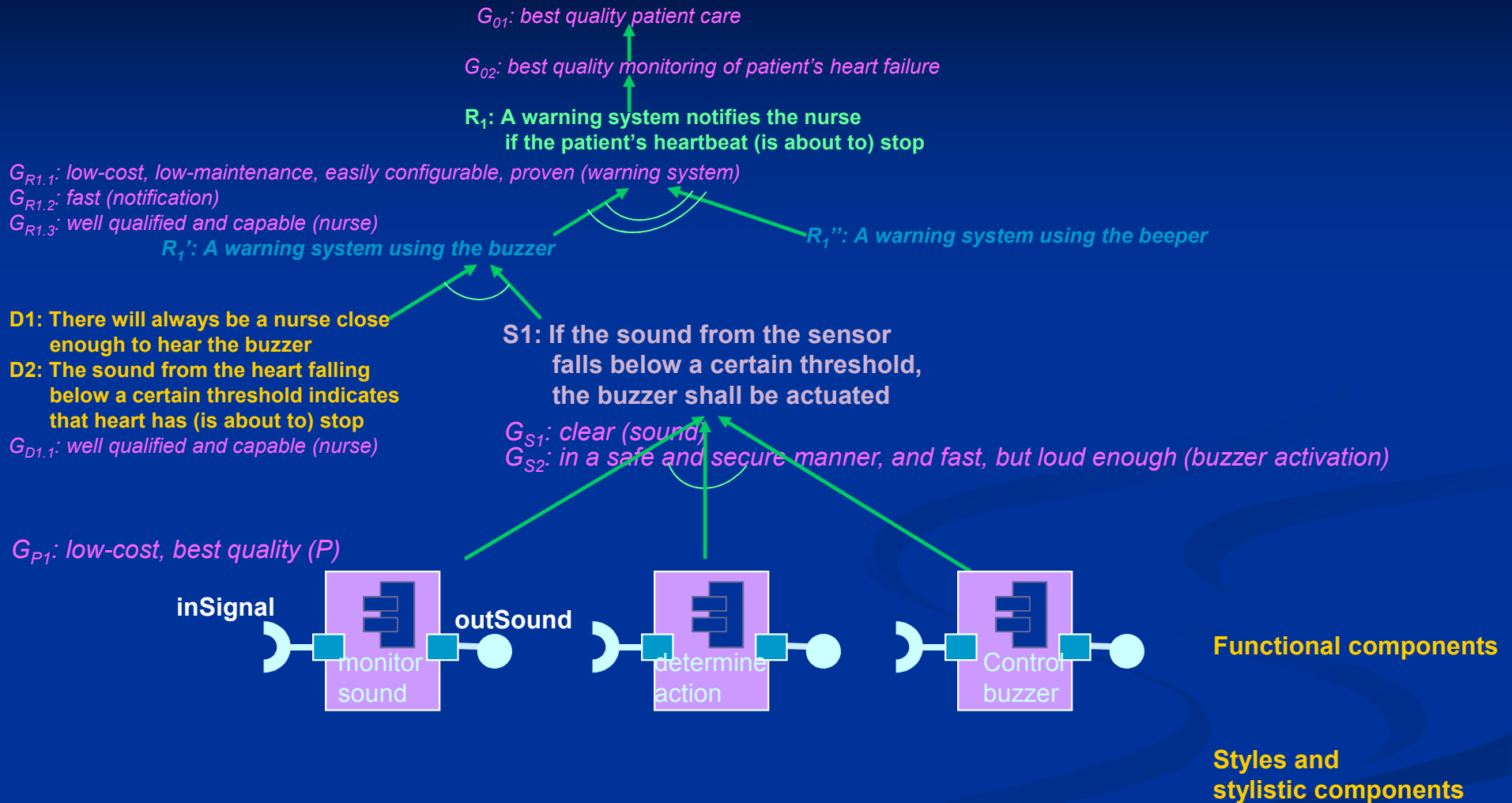
The NFR Framework

and the Reference Model



The NFR Framework

From Specification to Architecture

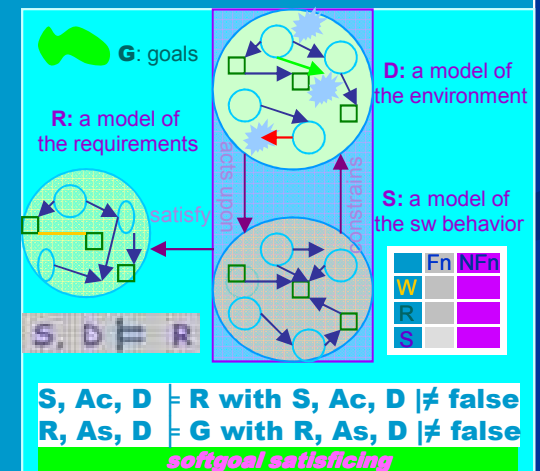


Data[1..*]

Lawrence Chung

Non-Functional Requirements

How 2 – Dos and Don'ts



NFRs – Dos & Don'ts

➤ *Dos*

- Relate to FRs
- Clarify scope/topic
- Identify agents, whenever useful
- Discover relationships between definitions of NFRs
- Discover relationships between solutions to NFRs
- Refine definitions as many times as needed
- Refine solutions as many times as needed
- Prioritize
- Discover conflicts
- Safeguard against conflicts
- Discover synergies
- Discover operationalizations as reasons for conflicts/synergies
- Determine strengths of contributions
- Justify strengths of contributions
- Explore alternatives
- Discover solutions from requirements
- Discover requirements from solutions
- Consider use of multiple solutions
- Consider scenarios
- If necessary, quantify
- Evaluate, ...subjectively, ...objectively
- Establish traceability

➤ *Don'ts*

- Absolute security, absolute reliability, absolute safety,
- One definition fits all
- One solution solves all problems
- The contribution is such and such, since I say so
- Refine the definition only once
- They are falling down from the sky
- Dissociate from FRs
- May be more important than FRs, but should consume less resources
- You name it; our system does it
- No quantification, no existence
- Everybody needs the same
- Be only pessimistic
- Asking why “+” reveals ignorance
- Beg the question
- Evaluate & only evaluate
- Brainwash nothing but objectivity

Conflict resolution 1

■ Delete email w. any zip file attachment

-> misunderstanding betw. sender and receiver

<- move email w. any zip file attachment into a junk file folder

-> If the receiver does not check the junk file folder, still misunderstanding

<- at the time the file is moved, notify this to the receiver

-> if the receiver still does not check the junk file folder or checks it late, still misunderstanding

<- at the time the file is moved, notify the sender too

-> If the receiver checks the junk file folder and opens it and the file is an attack, still a security breach

Delete email w. any zip file attachment and block any future email from the same sender

Conflict resolution 2

- If the receiver opens email w. zip file and the file is an attack, a security breach
- Delete email any w. zip file attachment
 - > misunderstanding betw. sender and receiver
 - <- move email w. any zip file attachment into a junk file folder
 - > If the receiver does not check the junk file folder, still misunderstanding
 - <- at the time the file is moved, notify this to the receiver
 - > if the receiver still does not check the junk file folder or checks it late, still misunderstanding
 - <- at the time the file is moved, notify the sender too
 - > If the receiver checks the junk file folder and opens it and the file is an attack, still a security breach
- Delete email w. any zip file attachment and block any future email from the same sender
- If the email is from a sender who is not in the list of allowed senders, delete it
- Leave the email but delete the attachment only

Conflict resolution 3

- Security[PC] \rightarrow S[email] \rightarrow S[sender] \wedge S[recipient] \wedge S[body] \wedge S[attachment]
- Denied (S[attachment]) \rightarrow denied (S[email]) \rightarrow denied (S[PC])
- Zip(attachment) \wedge attack(attachment) \wedge open(attachment) \rightarrow denied (S[attachment])
/* If the receiver opens email w. zip file and the file is an attack, a security breach */
 \sim Zip(attachment) \vee \sim attack(attachment) \vee \sim open(attachment) \rightarrow \sim denied (S[attachment]) helps
 \sim denied(S[email])
- Delete email w. any zip file attachment
 - \rightarrow misunderstanding betw. sender and receiver
 - \leftarrow move email w. any zip file attachment into a junk file folder
 - \rightarrow If the receiver does not check the junk file folder, still misunderstanding
 - \leftarrow at the time the file is moved, notify this to the receiver
 - \rightarrow if the receiver still does not check the junk file folder or checks it late, still misunderstanding
 - \leftarrow at the time the file is moved, notify the sender too
 - \rightarrow If the receiver checks the junk file folder and opens it and the file is an attack, still a security breach
- Leave the email, but delete the attachment only
- Leave the email, but delete the attachment only if it is an attack
- Leave the email but change the name of the attachment to "...renameToZip"
- If the email is from a sender who is not in the list of allowed senders, delete it
- Delete email containing file attachment and block sender from sending email from the sender

Conflict resolution 4

- Security[PC] \rightarrow S[email] \rightarrow S[sender] \wedge S[recipient] \wedge S[body] \wedge S[attachment]
- Denied (S[attachment]) \rightarrow denied (S[email]) \rightarrow denied (S[PC])
- Zip(attachment) \wedge attack(attachment) \wedge open(attachment) \rightarrow denied (S[attachment])
/* If the receiver opens email w. zip file and the file is an attack, a security breach */
 \sim Zip(attachment) \vee \sim attack(attachment) \vee \sim open(attachment) \rightarrow \sim denied (S[attachment]) helps
 \sim denied(S[email])
- Delete email w. any zip file attachment, **at the time of reception**
 \rightarrow misunderstanding betw. sender and receiver
 - <- move email w. any zip file attachment into a junk file folder
 - \rightarrow If the receiver does not check the junk file folder, still misunderstanding
 - <- at the time the file is moved, notify this to the receiver
 - \rightarrow if the receiver still does not check the junk file folder or checks it late, still misunderstanding
 - <- at the time the file is moved, notify the sender too
 - \rightarrow If the receiver checks the junk file folder and opens it and the file is an attack, still a security breach
- Leave the email, but delete the attachment only
- Leave the email, but delete the attachment only if it is an attack: **detectable[attack(attachment)]**
- Leave the email but change the name of the attachment to "...renameToZip"
- If the email is from a sender who is not in the list of allowed senders, delete it
- Delete email containing file attachment with extension .zip from the sender's mailbox

NFRs – Where

➤ *Wherever better/cheaper/faster/happier matters*

- Requirements Engineering
- System Architecting
- Software Architecting
- Design
- Implementation
- Validation & Verification
- Testing
- Maintenance
- Software Process
- Project Planning and Management
- Configuration Management
- Decision making

NFRs – How to represent

➤ *From informal to tabular to visual (a la html->xml->oo-xml/eb-xml/...; CRC cards->classes; use cases & use case templates)*

➤ ***Dos***

- Bring in FRs
- Clarify scope/topic
- Identify agents, whenever useful
- Discover relationships between definitions of NFRs
- Discover relationships between solutions to NFRs
- Refine definitions as many times as needed
- Refine solutions as many times as needed
- Prioritize
- Discover conflicts
- Safeguard against conflicts
- Discover synergies
- Discover operationalizations as reasons for conflicts/synergies
- Determine strengths of contributions
- Justify strengths of contributions
- Explore alternatives
- Discover solutions from requirements
- Discover requirements from solutions
- Consider use of multiple solutions
- Consider scenarios
- If necessary, quantify
- Evaluate
- Evaluate subjectively
- Evaluate objectively
- Establish traceability

Name	
Description	
Type	
Topic	
Agent	
Viewpoint	
Priority	
Affected NFRs	
Affecting NFRs/Operationalizations	
Claim	
Sat Status	

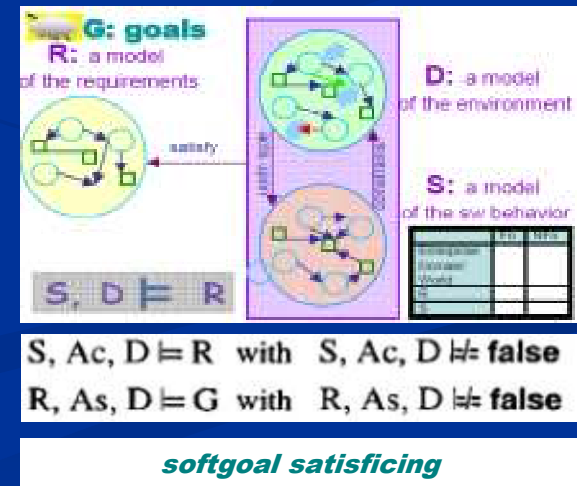
Non-Functional Requirements

Practices and Recommendations:
A Brief Synopsis

- Why
- What
- Some Classification Schemes
- NFRs and RE Processes
- Product-Oriented Approach: Some Individual NFRs
- The NFR Framework
- Appendix
 - With Rational Unified Process and UML
 - With Volere Requirements Specification Templates
 - Others

Appendix

- Dependability
- RUP Specification
- Volere Specification
- How to Augment UML



Dependability

■ Dimensions of Dependability

- **Availability** - The ability of the system to deliver services when requested
- **Reliability** - The ability of the system to deliver services as specified
- **Safety** - The ability of the system to operate without catastrophic failure
- **Security** - The ability of the system to protect itself against accidental or deliberate intrusion

■ Cost of development - Geometric rise in cost from low dependability to highest

■ Effects of low dependability

- Often unused
- Failure recovery costs may be high
- Difficult to retrofit dependability
- Loss of information

■ Repeatable improvement process helps

- CMM -SEI
- More later

■ Critical Systems

- Safety critical
- Mission critical
- Business critical

■ Dependability a key aspect

- A system failure causes
 - Significant economic loss
 - Physical damage
 - Threat to or loss of human life

Dependability

■ Cost of failure

- direct
 - Loss of life / Injury
 - Loss of business
- Indirect
 - Litigation
 - Good will

■ **Availability** and **Reliability**

- Factors effecting
 - Environment office versus university
 - Perception (frequency of occurrence)

■ Degrees

- Failure - service that is expected is not delivered
- Error – behavior that does not conform to the specification
- Fault – incorrect state – un-anticipated
- Human error

■ Improve **reliability**

- Fault avoidance
- Fault detection and removal – testing and debugging
- Fault tolerance - self checking and redundancy

■ Errors of this type are random

- Remain after testing due to unforeseen combinations of input or use
- Random based on user methods
 - Not all inputs done the same
 - Learn to avoid
 - Therefore removal of some faults will not improve perception

Dependability - **Safety**

- Ability to operate normally or abnormally without threat to life or environment
- Classes
 - Primary safety critical
 - Embedded as controller
 - Secondary
 - Their output could effect indirectly other processes (CAD)
- Reasons for less than 100% certainty of fault tolerant/free
 - Incomplete specification
 - Hardware malfunction – causing exceeded limits in software
 - Incorrect input

- Methods to lessen chance of safety failure
 - Hazard avoidance
 - Added control features (I.e. two man rule)
 - Hazard detection and removal
 - Scans for known causes and cause preventive action
 - Damage limitation (control)
 - Firewalls and other protective reactions to results
- Terms
 - Accident
 - Hazard
 - Damage
 - Hazard Severity
 - Hazard Probability
 - Risk

Specification

■ Safety

- IEC 61508 safety life cycle
 - Concept to death
 - Hazard analysis
 - Safety requirements definition
 - Planning , validation, development, external risk reduction
 - Separate safety validation – installation and commissioning
 - O&M
 - Decommissioning
 - Hazard and Risk Analysis
 - Iterative process
 - Hazard Identification
 - Hazard description
 - Risk analysis and hazard classification
 - Risk assessment
 - Hazard decomposition
 - Analysis as to potential causes (fault-tree analysis)
 - Risk reduction analysis
 - Preliminary safety requirements

■ Fault tree

- Deductive – start with a hazard
- Inductive – start with failure
- Fault tree starts with the failure and works backwards to potential causes

■ Risk assessment

- Classifications
 - Intolerable
 - As low as reasonably practical (ALARP)
 - Acceptable
- For each hazard
 - Probability
 - Severity
 - Estimated risk

■ Risk reduction

- Avoidance
- Detection and removal
- Damage limitation

Dependability - **Security**

- Lack of **security** comprise to **availability** and **reliability**

- Types

- Denial of service
- Corruption of programs or data
- Unauthorized disclosure

- Terms

- Exposure
- Vulnerability
- Attack
- Threats
- Controls

- Methods

- Vulnerability avoidance
- Detection and neutralization
- Damage limitation

- **Security Specification**

- Similar to safety
- Impractical to specify
- Usually are “shall not”

- Cycle in General

- Asset ID and evaluation
 - Degree of importance
- Threat analysis and risk assessment
- Threat assignment lists all threats against each asset
- Technology analysis what is available to counteract
- Security specification

Specification

Requirements specification

- Functional for error detection and recovery
- Non functional for **reliability** and **availability**
- Shall not requirements

Reliability specification

- Hardware
- Software
- Operator

Decrease probability of failure

- For a series of dependent components $P_t = \text{sum of } P_1 \text{ to } P_n$
- But if there are n replicated (redundant) and independent components then the $P_t = p_a$ to the n th

Metrics for reliability

- POFD probability of failure on demand $.0001 = 1 \text{ on } 10000$
 - Systems with unpredictable demand over long time periods – emergency systems
- ROCOF Rate of failure occurrence $2/1000$
 - Systems with a regular demand atm/airline reservations
- MTTF Mean time to Failure avg time between observed failures $500 = \text{avg of } 1 \text{ in } 500 \text{ time units}$
 - Systems with long transactions (auto save)
- AVAIL probability system is available at any given time $.999$ equals in every given 1000 time units system is likely to be available for 999 of these
 - Systems of continuous service; tp switch

Lawrence Chung

Non-functional **reliability** requirements

- ID type of failure to occur
- Partition them into
 - Transient
 - Permanent
 - Recoverable
 - Unrecoverable
 - Non-corrupting
 - Corrupting
- Define the appropriate requirement (metric)
 - E.g. recoverable w/intervention – POFOD
 - If automatic the ROCOF
- Assign a proper metric as a functional reliability metric

NFRs: With Rational Unified Process and UML

Home Appliance Control System Vision Version 1.2

Revision History

Date	Version	Description	Author
------	---------	-------------	--------

Table of Contents

1.	Introduction	5
1.1	Purpose	5
1.2	Scope	5
1.3	Definitions, Acronyms, and Abbreviations	5
1.4	References	5
2.	Positioning	5
2.1	Business Opportunity	5
2.2	Problem Statement	5
2.3	Product Position Statement	6
3.	Stakeholder and User Descriptions	6
3.1	Market Demographics	6
3.2	Stakeholder Summary	6
3.3	User Summary	7
3.4	User Environment	7
3.5	Stakeholder Profiles	7
	3.5.1 Homeowner	7
	3.5.2 Business Owner	8
	3.5.3 Customer Care	8
3.6	User Profiles	9
3.7	Key Stakeholder or User Needs	9
3.8	Alternatives and Competition	9
3.8.1	House Sitter	9
3.8.2	Home Security System and Monitoring Company	9

4.	Product Overview	9
4.1	Product Perspective	9
4.2	Summary of Capabilities	10
4.3	Assumptions and Dependencies	11
4.4	Cost and Pricing	11
4.5	Licensing and Installation	11
5.	Product Features	11
5.1	Start system	11
5.2	Shutdown system	11
5.3	View status of system	11
5.4	Add a new group of sequences	12
	...	

6.	Constraints	14
6.1	Security	14
6.2	Usability	15
6.3	Responsiveness	15
6.4	Capacity	15
	Appendix A. COTS Components	15

NFRs:

With Rational Unified Process and UML

6. Constraints

6.1 Security

Security for the HACS includes authentication, access control, data integrity, and data privacy.

Authentication of the user is by identifier and password.

Homeowners and Business Owners can monitor and change the state of the system.

Customer Care users can only monitor the system and manually place a medical alert 911 emergency request for an ambulance.

Transmissions should be encrypted for privacy

6.2 Usability

Easy to use (especially safety related features)

Request for an ambulance, police or fire truck needs to be at the push of a button or voice activated

6.3 Responsiveness

System responds quickly to user requests or changes in the environment.

System responds within 2 seconds on average to local user requests and changes in the environment.

System responds within 4 seconds on average to remote user requests and changes in the environment.

6.4 Capacity

Maximum number of sequences for indoor lights is twenty (20)

Maximum number of indoor lights that can be controlled is fifty (50)

Maximum number of sequences for outdoor lights is twenty (20)

Maximum number of outdoor lights that can be controlled is fifty (50)

Maximum number of sequences for radios, CD players, televisions is twenty (20)

Maximum number of radios, CD players, televisions that can be controlled is ten (10)

Maximum number of sequences for safety and security equipment is twenty (20)

Maximum number of sensors, security cameras, security VCRs, emergency notifications, that can be controlled is fifty (50)

NFRs: With Volere Requirements Specification Template

The Atlantic Systems Guild Limited

Table of Contents

(<http://www.volere.co.uk/template.htm>)

PROJECT DRIVERS:

- [1. The Purpose of the Project](#)
- [2. Client, Customer, Stakeholders](#)
- [3. Users of the Product](#)

PROJECT CONSTRAINTS:

- [4. Mandated Constraints](#)
- [5. Naming Conventions and Definitions](#)
- [6. Relevant Facts and Assumptions](#)

FUNCTIONAL REQUIREMENTS:

- [7. The Scope of the Work](#)
- [8. The Scope of the Product](#)
- [9. Functional and Data Requirements](#)

NON-FUNCTIONAL REQUIREMENTS:

- [10. Look and Feel](#)
- [11. Usability and Humanity](#)
- [12. Performance](#)
- [13. Operational](#)
- [14. Maintainability and Support](#)
- [15. Security](#)
- [16. Cultural and Political](#)
- [17. Legal](#)

PROJECT ISSUES:

- [18. Open Issues](#)
- [19. Off-the-shelf Solutions](#)
- [20. New Problems](#)
- [21. Tasks](#)
- [22. Cutover](#)
- [23. Risks](#)
- [24. Costs](#)
- [25. User Documentation and Training](#)
- [26. Waiting Room](#)
- [27. Ideas for Solutions](#)

NFRs: With Volere Requirements Specification Template

10 Look and Feel Requirements

10a. The interface

Content

The section contains requirements relating to spirit of the interface. Your client may have given you particular demands such as corporate branding, style, colors to be used, degree of interaction and so on. This section captures the requirements for the interface rather than the design for the interface.

Motivation

To ensure that the appearance of the product conforms to the organization's expectations.

Examples

The product shall comply with corporate branding standards.

The product shall be attractive to a teenage audience.

The product shall appear authoritative.

Considerations

Interface design may overlap the requirements gathering process. This particularly true if you are using prototyping as part of your requirements process. As prototypes develop it is important to capture the requirements that relate to the look and feel. In other words, be sure that you understand your client's intentions for the product's look and feel. Record these as requirements instead of merely having a prototype to which the client has nodded his approval.

10b. The style of the product

Content

A description of salient features of the product that are related to the way a potential customer will see the product. For example, if your client wants the product to appeal to the business executive, then a look and feel requirement is that the product has a conservative and professional appearance. Similarly if the product is for sale to children, then the look and feel requirement is that it be colorful and look like it's intended for children. ...

Motivation

Given the state of today's market and people's expectations, ... Once the functional requirements are satisfied, it is often the appearance of products that determines whether they are successful or not. ...

Considerations

The look and feel requirements specify the your client's vision of the product's appearance. The requirements may at first

NFRs: With Volere Requirements Specification Template

11 Usability and Humanity Requirements

11a. Ease of use.

Content

This section describes your client's aspirations for how easy it will be for the intended users of the product to operate it. The product's usability is derived from the abilities of the expected users of the product and the complexity of its functionality.

The usability requirements should cover such things as:

Efficiency of use - how quickly or accurately the user can use the product.

Ease of remembering - how much is the casual user expected to remember about using the product

Error rates - for some products it is crucial that the user commits very few, or no, errors.

Overall satisfaction in using the product - this is especially important for commercial, interactive products where there is a lot of competition. Web sites are good example of this.

Feedback - how much feedback does the user need in order to feel confident that the product is actually accurately doing what the user expects. The necessary degree of feedback will be higher for some products (eg: safety critical) than in others.

Motivation

To guide the product's designers into building a product that will meet the expectations of its eventual users.

Examples

The product shall be easy for 11 year-old children to use.

The product shall help the user to avoid making mistakes.

The product shall make the users want to use it.

The product shall be used by people with no training, and possibly no understanding of English.

Fit Criterion

These examples may seem simplistic, but they do express the intention of the client. To completely specify what is meant by the requirement it is necessary to add a measurement of acceptance. We call this a fit criterion. The fit criterion for the above examples would be:

[An agreed percentage, say 90%] of a test panel of 11 year olds shall be able to successfully complete [list of tasks] within [specified time]

One month's use of the product shall result in a total error rate of less than [an agreed percentage, say 2%]

An anonymous survey shall show that [an agreed percentage, say 75%] of the users are regularly using the product after [an agreed time] familiarization period.

NFRs: With Rational Unified Process and UML

Home Appliance Control System Vision Version 1.2

Revision History

Date	Version	Description	Author
------	---------	-------------	--------

Table of Contents

1.	Introduction	5
1.1	Purpose	5
1.2	Scope	5
1.3	Definitions, Acronyms, and Abbreviations	5
1.4	References	5
2.	Positioning	5
2.1	Business Opportunity	5
2.2	Problem Statement	5
2.3	Product Position Statement	6
3.	Stakeholder and User Descriptions	6
3.1	Market Demographics	6
3.2	Stakeholder Summary	6
3.3	User Summary	7
3.4	User Environment	7
3.5	Stakeholder Profiles	7
	3.5.1 Homeowner	7
	3.5.2 Business Owner	8
	3.5.3 Customer Care	8
3.6	User Profiles	9
3.7	Key Stakeholder or User Needs	9
3.8	Alternatives and Competition	9
3.8.1	House Sitter	9
3.8.2	Home Security System and Monitoring Company	9

NFRs: With Rational Unified Process and UML

4.	Product Overview	9
4.1	Product Perspective	9
4.2	Summary of Capabilities	10
4.3	Assumptions and Dependencies	11
4.4	Cost and Pricing	11
4.5	Licensing and Installation	11
5.	Product Features	11
5.1	Start system	11
5.2	Shutdown system	11
5.3	View status of system	11
5.4	Add a new group of sequences	12
5.5	Modify an existing group of sequences	12
5.6	Delete an existing group of sequences	12
5.7	Categorize a group	12
5.8	Schedule a group	12
5.9	Start a group	12
5.10	Stop a group	12
5.11	View the status of whole system	12
5.12	View the status of indoor lights	12
5.13	View the status of outdoor lights	12
5.14	View the status of entertainment equipment (radios, cd players, televisions)	12
5.15	View the status of the safety system	12
5.16	View the status of the security system	12
5.17	Make a new sequence	12
5.18	Modify an existing sequence	12
5.19	Delete an existing sequence	12
5.20	Schedule a sequence	12
5.21	Start a sequence	12
5.22	Stop a sequence turn on indoor lights (all)	12
5.23	Turn off indoor lights (all)	12
5.24	Turn on selected indoor lights	12
5.25	Turn off selected indoor lights	12
5.26	Make a new sequence	13
5.27	Modify an existing sequence	13
5.28	Delete an existing sequence	13

NFRs:

With Rational Unified Process and UML

5.29	Schedule a sequence	13	
5.30	Start a sequence	13	
5.31	Stop a sequence turn on outdoor lights (all)		13
5.32	Turn off outdoor lights (all)	13	
5.33	Turn on selected outdoor lights	13	
5.34	Turn off selected outdoor lights	13	
5.35	Make a new sequence	13	
5.36	Modify an existing sequence	13	
5.37	Delete an existing sequence	13	
5.38	Schedule a sequence	13	
5.39	Start a sequence	13	
5.40	Stop a sequence	13	
5.41	Turn on radios, cd players, televisions (all)		13
5.42	Turn off radio, cd player, television (all)	13	
5.43	Turn on selected radio, cd player, television		13
5.44	Turn off selected radio, cd player, television		13
5.45	Automatic notification of emergency	14	
5.46	Make a new sequence	14	
5.47	Modify an existing sequence	14	
5.48	Delete an existing sequence	14	
5.49	Schedule a sequence	14	
5.50	Start a sequence	14	
5.51	Stop a sequence	14	
5.52	Turn on security system (all features)	14	
5.53	Turn off security system (all features)	14	
5.54	Turn on safety system (all features)	14	
5.55	Turn off safety system (all features)	14	
5.56	Turn on selected features of security system		14
5.57	Turn off selected features of security system		14
5.58	Turn on selected features of safety system		14
5.59	Turn off selected features of safety system		14

NFRs: With Rational Unified Process and UML

6.	Constraints	14
6.1	Security	14
6.2	Usability	15
6.3	Responsiveness	15
6.4	Capacity	15
Appendix A. COTS Components		15

NFRs:

With Rational Unified Process and UML

6. Constraints

6.1 Security

Security for the HACS includes authentication, access control, data integrity, and data privacy.

Authentication of the user is by identifier and password.

Homeowners and Business Owners can monitor and change the state of the system.

Customer Care users can only monitor the system and manually place a medical alert 911 emergency request for an ambulance.

Transmissions should be encrypted for privacy

6.2 Usability

Easy to use (especially safety related features)

Request for an ambulance, police or fire truck needs to be at the push of a button or voice activated

6.3 Responsiveness

System responds quickly to user requests or changes in the environment.

System responds within 2 seconds on average to local user requests and changes in the environment.

System responds within 4 seconds on average to remote user requests and changes in the environment.

6.4 Capacity

Maximum number of sequences for indoor lights is twenty (20)

Maximum number of indoor lights that can be controlled is fifty (50)

Maximum number of sequences for outdoor lights is twenty (20)

Maximum number of outdoor lights that can be controlled is fifty (50)

Maximum number of sequences for radios, CD players, televisions is twenty (20)

Maximum number of radios, CD players, televisions that can be controlled is ten (10)

Maximum number of sequences for safety and security equipment is twenty (20)

Maximum number of sensors, security cameras, security VCRs, emergency notifications, that can be controlled is fifty (50)

NFRs: With Volere Requirements Specification Template

The Atlantic Systems Guild Limited

Table of Contents

(<http://www.volere.co.uk/template.htm>)

PROJECT DRIVERS:

1. The Purpose of the Project
2. Client, Customer, Stakeholders
3. Users of the Product

PROJECT CONSTRAINTS:

4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

FUNCTIONAL REQUIREMENTS:

7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

NON-FUNCTIONAL REQUIREMENTS:

10. Look and Feel
11. Usability and Humanity
12. Performance
13. Operational
14. Maintainability and Support
15. Security
16. Cultural and Political
17. Legal

PROJECT ISSUES:

18. Open Issues
19. Off-the-shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

NFRs:

With Volere Requirements Specification Template

10 Look and Feel Requirements

10a. The interface

Content

The section contains requirements relating to spirit of the interface. Your client may have given you particular demands such as corporate branding, style, colors to be used, degree of interaction and so on. This section captures the requirements for the interface rather than the design for the interface.

Motivation

To ensure that the appearance of the product conforms to the organization's expectations.

Examples

The product shall comply with corporate branding standards.

The product shall be attractive to a teenage audience.

The product shall appear authoritative.

Considerations

Interface design may overlap the requirements gathering process. This particularly true if you are using prototyping as part of your requirements process. As prototypes develop it is important to capture the requirements that relate to the look and feel. In other words, be sure that you understand your client's intentions for the product's look and feel. Record these as requirements instead of merely having a prototype to which the client has nodded his approval.

10b. The style of the product

Content

A description of salient features of the product that are related to the way a potential customer will see the product. For example, if your client wants the product to appeal to the business executive, then a look and feel requirement is that the product has a conservative and professional appearance. Similarly if the product is for sale to children, then the look and feel requirement is that it be colorful and look like it's intended for children.

You would also consider here the design of the package if this were to be a manufactured product. The package may have some requirements as to its size, style, and consistency with other packages put out by your organization, etc. Keep in mind the European laws on packaging. There is a requirement that the package not be significantly larger than the product it

NFRs:

With Volere Requirements Specification Template

11 Usability and Humanity Requirements

11a. Ease of use.

Content

This section describes your client's aspirations for how easy it will be for the intended users of the product to operate it. The product's usability is derived from the abilities of the expected users of the product and the complexity of its functionality.

The usability requirements should cover such things as:

Efficiency of use - how quickly or accurately the user can use the product.

Ease of remembering - how much is the casual user expected to remember about using the product

Error rates - for some products it is crucial that the user commits very few, or no, errors.

Overall satisfaction in using the product - this is especially important for commercial, interactive products where there is a lot of competition. Web sites are good example of this.

Feedback - how much feedback does the user need in order to feel confident that the product is actually accurately doing what the user expects. The necessary degree of feedback will be higher for some products (eg: safety critical) than in others.

Motivation

To guide the product's designers into building a product that will meet the expectations of its eventual users.

Examples

The product shall be easy for 11 year-old children to use.

The product shall help the user to avoid making mistakes.

The product shall make the users want to use it.

The product shall be used by people with no training, and possibly no understanding of English.

Fit Criterion

These examples may seem simplistic, but they do express the intention of the client. To completely specify what is meant by the requirement it is necessary to add a measurement of acceptance. We call this a fit criterion. The fit criterion for the above examples would be:

[An agreed percentage, say 90%] of a test panel of 11 year olds shall be able to successfully complete [list of tasks] within [specified time]

One month's use of the product shall result in a total error rate of less than [an agreed percentage, say 2%]

NFRs:

With Volere Requirements Specification Template

11b. Personalization and internationalization requirements

Content

This section describes the way in which the product can be altered or configured to take into account the user's personal preferences or choice of language. The personalization requirements should cover such things as:

Languages, spelling preferences, language idioms

Currencies including the symbols and decimal conventions

Personal configuration options - there are a myriad of these

Motivation

To ensure that the product's users do not have to struggle with, or meekly accept, the cultural conventions of the builder.

Examples

The product shall retain the buyer's buying preferences.

The product shall allow the user to select a chosen language.

Considerations

Consider the locations of the potential customers and users of your product. Any out of country users will welcome the opportunity to convert to their home spelling and expressions.

By allowing users to customize the way in which they use the product, you are giving them the opportunity to participate more closely with your organization, as well as give them their own personal user experience.

You might also consider the configurability of the product. This allows different users to have different functional variations of the product.

NFRs:

With Volere Requirements Specification Template

11c. Ease of learning.

Content

A statement of how easy it should be to learn to use the product. This will range from zero time for products intended for placement in the public domain (for example a parking meter or a web site) to a considerable time for complex, highly technical products. (We know of one product where it was necessary for graduate engineers to spend 18 months in training before being qualified to use the product.)

Motivation

To quantify the amount of time that your client feels is allowable before a user can successfully use the product. This requirement will guide designers in how users will learn the product. For example, the designers may build elaborate interactive help facilities into the product, or the product may be packaged with a tutorial. Alternatively the product may have to be constructed so that all of its functionality is apparent upon first encountering it.

Examples

The product shall be easy for an engineer to learn.

A clerk shall be able to be productive within a short time.

The product shall be able to be used by members of the public who will receive no training before using it.

The product shall be used by engineers who will attend 5 weeks of training before using the product.

Fit Criterion

Fit criterion for the above example requirements are:

An engineer shall produce a [specified result] within [specified time] of beginning to use the product, without needing to use the manual.

After receiving [number of hours] training a clerk shall be able to produce [quantity of specified outputs] per [unit of time].

[Agreed percentage] of a test panel shall successfully complete [specified task] within [specified time limit].

The engineers shall achieve [agreed percentage] pass rate from the final examination of the training.

Considerations

Refer back to Section 3, the Users of the System, to ensure that you have considered the ease of learning requirements from the perspective of all the different types of users.

NFRs:

With Volere Requirements Specification Template

11d. Understandability and Politeness requirements.

This section is concerned with discovering requirements related to concepts and metaphors that are familiar to the intended end-users.

Content

This specifies the requirement for the product to be understood by its users. While usability refers to ease of use, efficiency etc., understanding determines whether the users instinctively know what the product will do for them. In other words, the product fits into their view of the world. You can think of this as the product being polite to its users and not expecting them to know or learn things that have nothing to do with their business problem.

Motivation

To avoid forcing the user to learn terms and concepts that are part of the product's internal construction and are not relevant to the user's world. To make the product more comprehensible and thus more likely to be adopted by its intended users.

Examples

The product shall use symbols and words that are naturally understandable by the user community.

The product shall hide the details of its construction from the user.

Considerations

Refer back to Section 3, the Users of the Product, and consider the world from the point of view of each of the different types of users.

11e. Accessibility requirements.

Content

The requirements for how easy it should be for people with common disabilities to access the product. These disabilities might be to do with sight, physical disablement, hearing, cognitive, or others.

Motivation

In many countries it is required that some products are made available to the disabled. In any event, it seems self-defeating to exclude this sizable community of potential customers.

Examples

The product shall be usable by partially-sighted users.

The product shall conform to the Americans with Disabilities Act.

NFRs:

With Volere Requirements Specification Template

12 Performance Requirements

12a. Speed and latency requirements

Examples

Any interface between a user and the automated system shall have a maximum response time of 2 seconds

The response shall be fast enough to avoid interrupting the user's flow of thought

The product shall poll the sensor every 10 seconds

The product shall download the new status parameters within 5 minutes of a change

Fit Criterion - Unit of measurement, Required range of values

12b. Safety critical requirements

Examples

The product shall not emit noxious gases that damage people's health.

The heat exchanger shall be shielded from human contact.

Fit Criterion - Description of the perceived risk, Factors that could cause the damage

Unit for measuring the factors that could cause the damage

"The product shall be certified to comply with the Health Department's standard E110-98. This is to be certified by qualified testing engineers."

"No member of a test panel of [specified size] shall be able to touch the heat exchanger. The heat exchanger must also comply with safety standard [specify which one]."

12c. Precision or accuracy requirements

Examples

All monetary amounts shall be accurate to 2 decimal places.

Accuracy of road temperature readings shall be within + or - 2 degrees centigrade.

Fit Criterion - Unit of measure plus degree of precision

NFRs:

With Volere Requirements Specification Template

12d. Reliability and Availability requirements

Examples

The product shall be available for use 24 hours per day, 365 days per year.

The product shall be available for use between the hours of 8:00am and 5:30pm.

The escalator shall run from 6am until the last flight arrives at 10pm.

The product shall achieve 99% up time.

12e. Robustness or Fault Tolerance requirements

Examples

The product shall continue to operate in local mode whenever it loses its link to the central server.

The product shall provide 10 minutes of emergency operation should it become disconnected from the electrical source.

12f. Capacity requirements

Examples

The product shall cater for 300 simultaneous users within the period from 9:00am to 11:am. Maximum loading other periods will be 150.

During a launch period the product shall cater for up to 20 people to be in the inner chamber.

Fit Criterion - quantified, and thus can be tested.

12g. Scalability requirements

Examples

The product shall be capable of processing the existing 100,000 customers. This number is expected to grow to 500,000 within three years.

The product shall be able to process 50,000 transactions an hour within two years of its launch.

12h. Longevity requirements

Examples

The product shall be expected to operate within the maximum maintenance budget for a minimum of 5 years

NFRs:

With Volere Requirements Specification Template

13 Operational Requirements

13a. Expected physical environment

Examples

The product shall be used by a worker, standing up, outside in cold, rainy conditions.

The product shall be used in noisy conditions with a lot of dust.

13b. Expected technological environment

13c. Partner applications

Examples

We must be able to interface with any html browser.

The new version of the spreadsheet must be able to access data from the previous 2 versions.

13d. Productization Requirements

Examples

The product shall be distributed as a ZIP file.

The product shall be able to be installed by an untrained user without recourse to separately-printed instructions.

NFRs:

With Volere Requirements Specification Template

14 Maintainability and Support Requirements

14a. Maintenance Requirements

Examples

- New MIS reports must be available within one working week of the date the requirements are agreed
- A new weather station must be able to be added to the system overnight

14b. Are there special conditions that apply to the maintenance of this product?

Examples

- The maintenance releases will be offered to end-users once a year.
- Every registered user will have access to our help site via the Internet.

Fit Criterion

Description of type of maintenance + amount of effort budgeted

14c. Supportability Requirements

14d. Adaptability requirements

Examples

- The product is expected to run under Windows 95 and Unix
- The product might eventually be sold to the Japanese market

Fit Criterion

Specification of system software on which the product must operate.

Specification of future environments in which the product is expected to operate.

14e. Installation requirements

Example

- The product shall be able to be installed in the specified environment within 2 working days.

NFRs:

With Volere Requirements Specification Template

15 Security Requirements

15a. Access requirements

Examples

Only direct managers can see the personnel records of their staff.

Only holders of current security clearance can enter the building.

Fit Criterion

System function name or system data name

User role/s and/or names of people who have clearance

15b. Integrity requirements

Examples

The product shall prevent its data from incorrect data being introduced.

The product shall protect itself from intentional abuse.

15c. Privacy requirements

Examples

The product shall make its user aware of its information practices before collection data from them.

The product shall notify customers of changes to its information policy.

15d. Audit requirements

15e. Immunity requirements

Content

The requirements for what the product has to do to protect itself from infection by unauthorized or undesirable software programs, such as viruses, worms, Trojan horses and others.

Motivation

To build a product that is as secure as possible from malicious interference.

Considerations

Each day brings more malevolence from the unknown, outside world. People buying software, or any other kind of product, expect that it can protect itself from outside interference,

NFRs:

With Volere Requirements Specification Template

16 Cultural and Political Requirements

16a. Cultural requirements

Examples

The product shall not be offensive to religious or ethnic groups.

The product shall be able to distinguish between French, Italian and British road numbering systems.

16b. Political requirements

Examples

The product shall be installed using component X.

The product shall make all functionality available to the managing director.

The product shall be developed using XYZ standards.

Considerations

Did you intend to develop the product on a Macintosh, when the office manager has laid down a edict that only Windows machines are permitted?

Is a director also on the board of a company that manufactures products similar to the one that you intend to build?

Whether you agree with these political requirements has little bearing on the outcome. The reality is that the system has to comply with political requirements even if you can find a better/more efficient/more economical solution.

A few probing questions here may save some heartache later. The political requirements might be purely concerned with the politics inside your organization. However there are situations when you need to consider the politics inside your customers' organizations or the national politics of

the country.

NFRs:

With Volere Requirements Specification Template

17 Legal Requirements

17a. Compliance requirements

Examples

Personal information shall be implemented so as to comply with the data protection act.

Fit Criterion

Lawyers' opinion that the product does not break any laws.

Considerations

Consider consulting lawyers to help identify the legal requirements.

Are there any copyrights/intellectual property that must be protected? Alternatively, do any competitors have copyrights that you might be in danger of infringing?

17b. Standards requirements

Example

The product shall comply with MilSpec standards.

The product shall comply with insurance industry standards.

The product shall be developed according to SSADM standard development steps.

Fit Criterion

The appropriate standard-keeper certifies that the standard has been adhered to.

Considerations

It is not always apparent that there are applicable standards because their existence is often taken for granted. Consider the following:

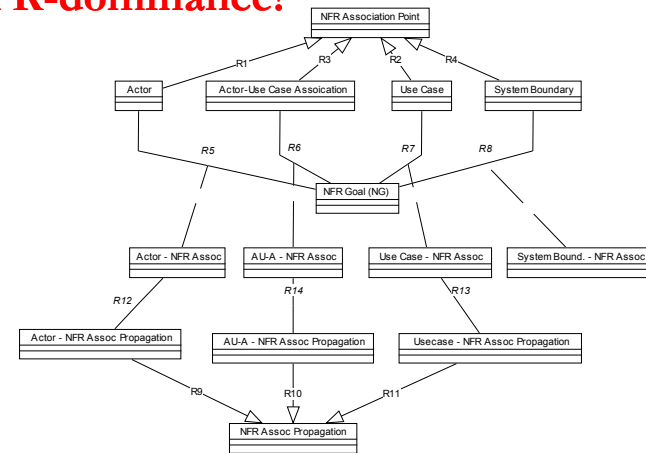
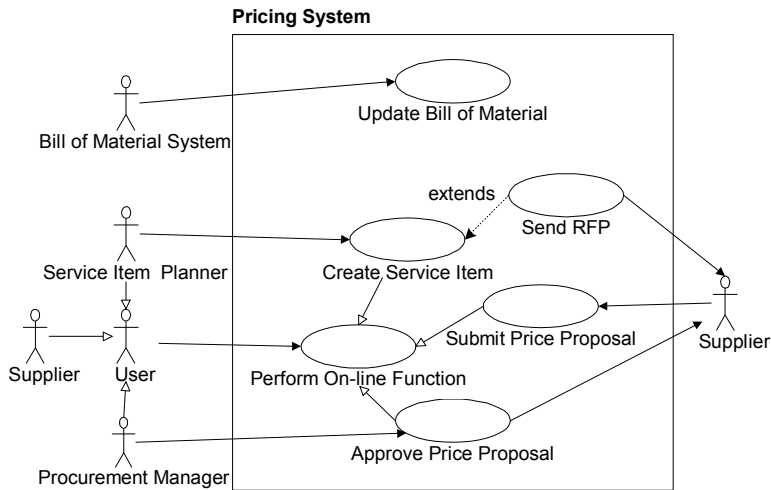
Are there any industry bodies that have applicable standards?

Has the industry a code of practice, watchdog or ombudsman?

Are there any special development steps for this type of product?

NFRs: With Rational Unified Process and UML

UML – de facto standard for OOA; but **FR-dominance!**

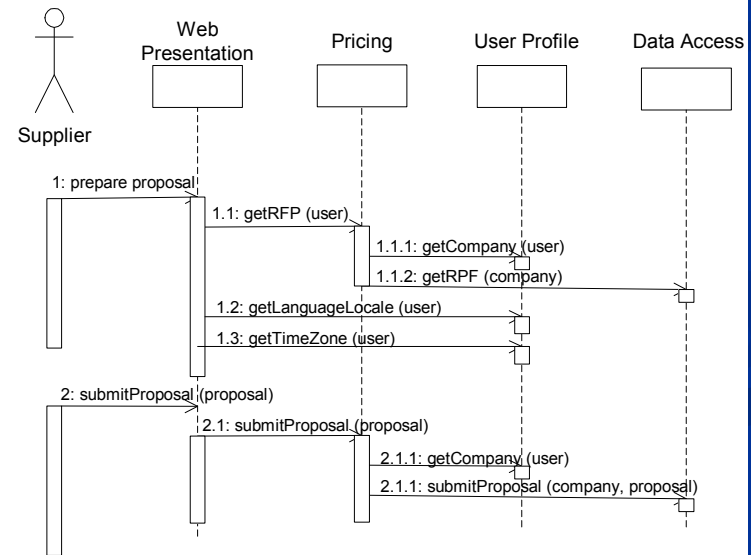


A Meta-model for partial FRs and NFRs Integration

Use cases as primary tool for FRs elicitation and modeling

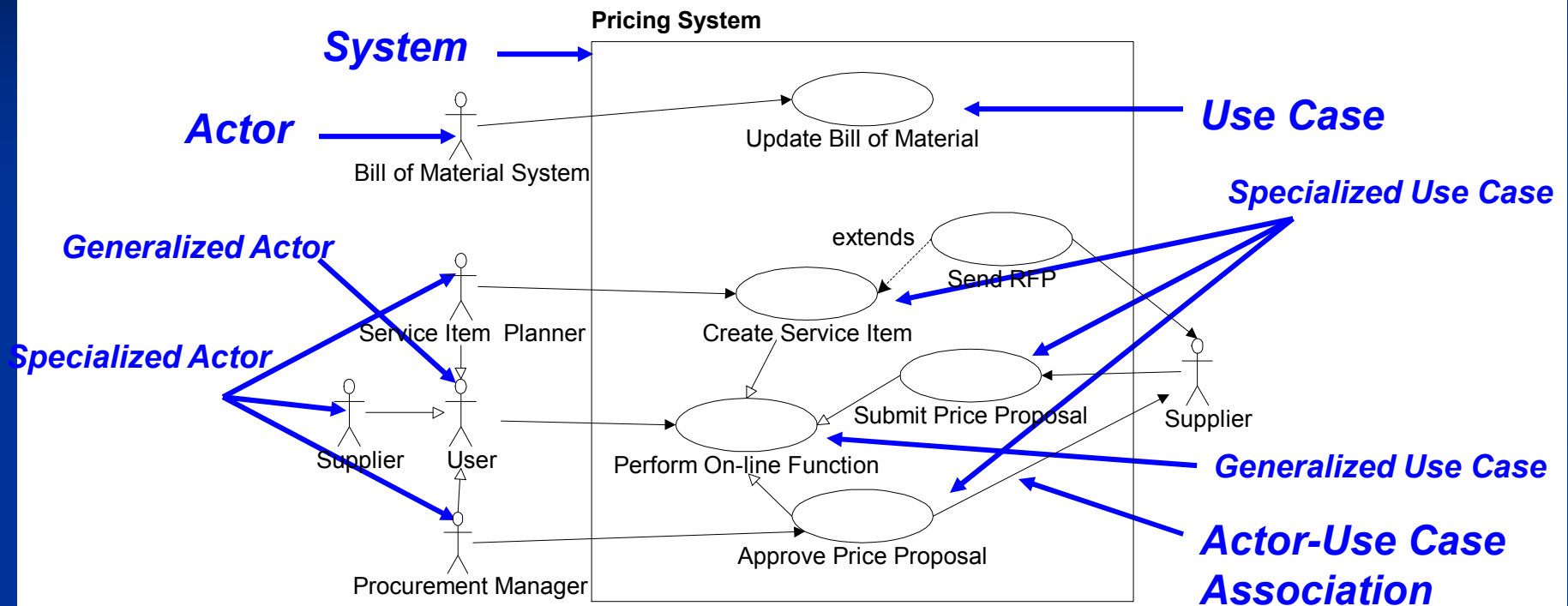
Package Dependency Diagram **Class diagram** to describe components/objects and their relationships

Use cases are realized with **interaction diagram** showing interaction between components or objects



NFRs: With Rational Unified Process and UML

What Are Use Cases?



System = the system in question that provides the functionality represented by use cases

Actor = an external entity (human or system)

Use case functionality (FRs) provided by the system

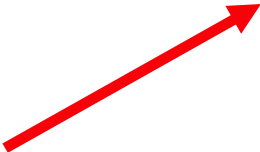
Actor-Use Case Association = an interface between an actor and the system

Use case details, including NFRs, are embedded textually using a template

NFRs: With Rational Unified Process and UML

Inadequate Handling of NFRs

Title	Submit Price Proposal
Description	Supplier submits price proposal against a RFP (request for proposal).
Actors	Supplier
Basic Flow	1.Supplier selects an RFP and requests system to submit a proposal against the RFP. 2.System prompts the Supplier for proposal information. 3.Supplier provides the following proposal information... 4....
Alternate Flows	In step 3, Supplier may request to ...
Special Requirements	Supplier may not see other suppliers' identity and submitted proposals.



Textual description for NFRs embedded in the use case special requirements section – not 1st class citizens

Problems:

1. NFRs not modeled and organized, and not *visually*
2. NFRs not traceable to architecture and design
3. Error prone if NFR applicable to multiple use cases

NFRs:

With Rational Unified Process and UML

Other Integration Schemes

Method	Integration Point	NFR Modeling Constructs	Drawbacks
Cysnerios's [1]	Text (LEL)	SIG, Class/ERD extensions	Not using the preferred use case modeling for FR elicitation
Lee's [2]	Use cases	Use cases	Using use cases (FR constructs) to represent NFRs. No organizational constructs.
Moreira's [3]	Text (use case template)	Unnamed use cases with stereo type name indicating the NFR, e.g., <<Security>>	Using use cases (FR constructs) to represent NFRs. Non-standard usage of unnamed entity. No organizational constructs.
Dimitrov's [4]	Use cases, Sequence diagram, State chart, Activity diagram	Informal annotation on diagrams	Specific to performance NFR. No organizational constructs.

No single scheme providing all of:

- Use case driven
- Modeling constructs for representing and organizing NFRs
- Preserving underlying use case principles (e.g., ovals for FRs but not for NFRs)
- Generic for a wide range of NFRs