Systems Engineering

mariaiulianadascalu@gmail.com

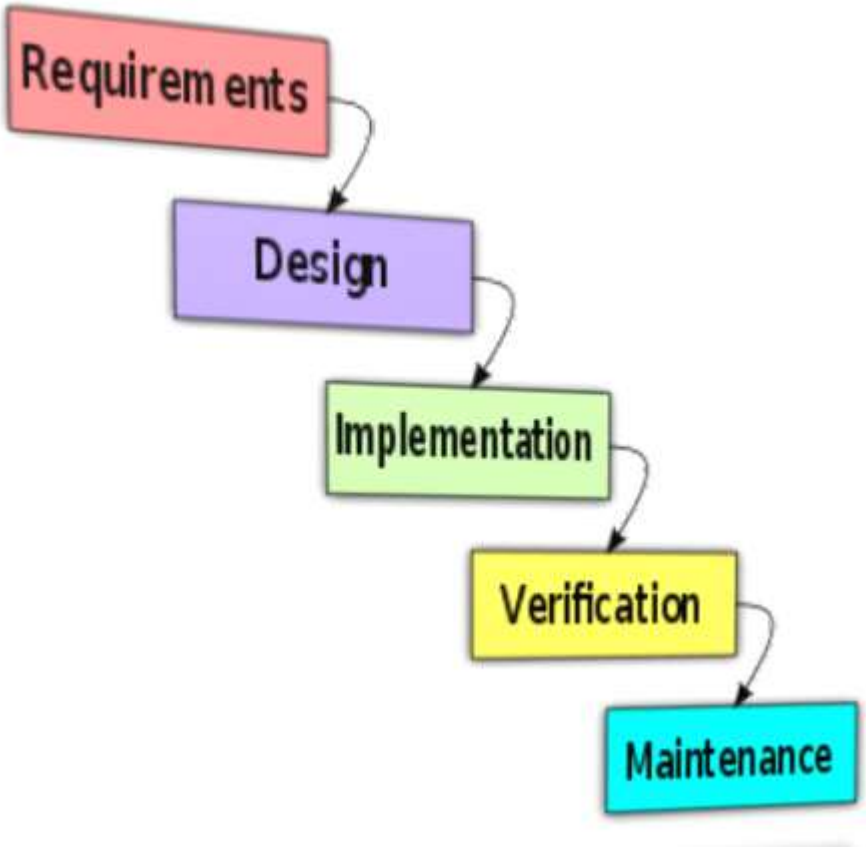
Politehnica University of Bucharest, Romania
Department of Engineering in Foreign Languages

Objectives of the current presentation

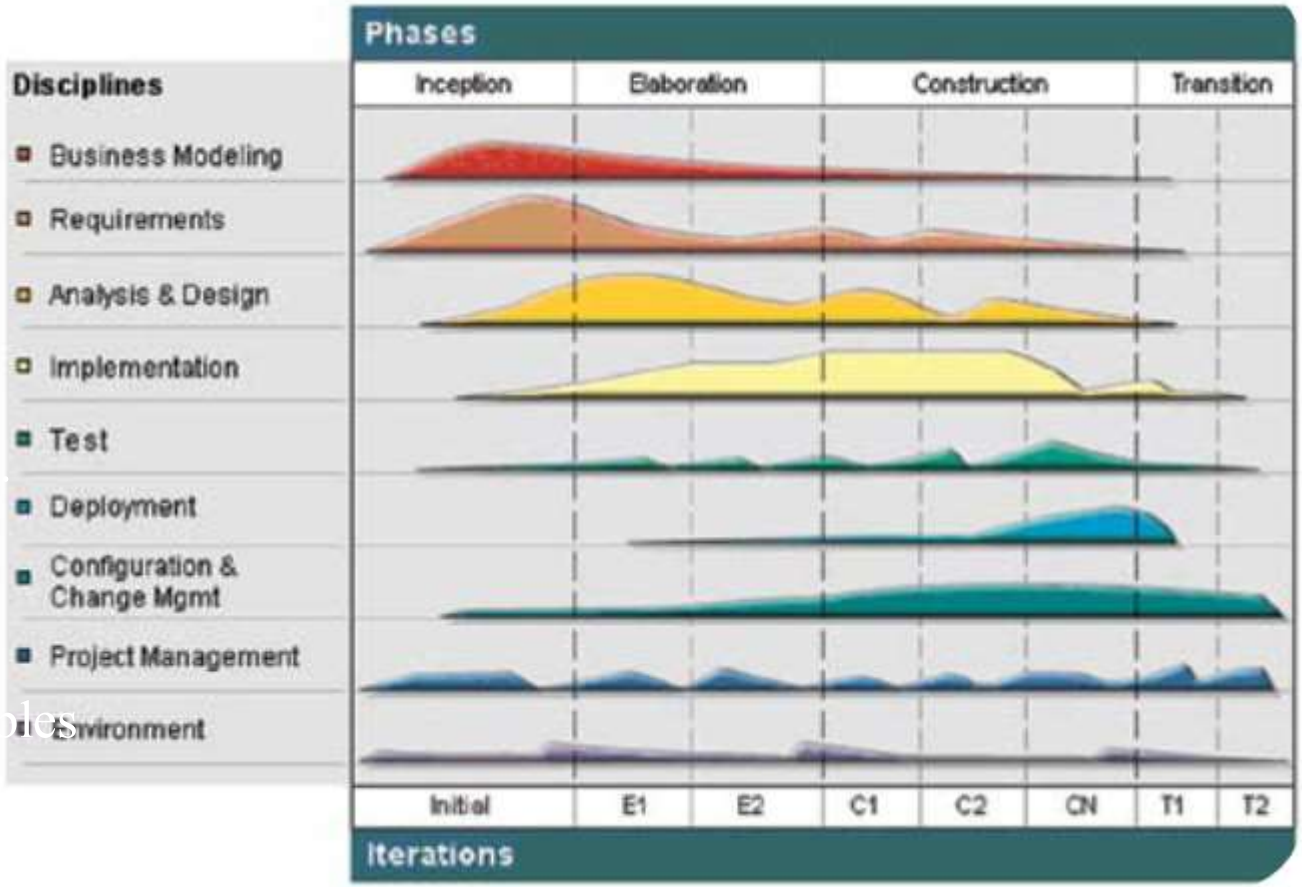
- Continue with Quality Assurance
  - Quality processes
- Requirements Management

Quality of processes can be ensured by…

Waterfall Software Development Model



Rational Unified Process (RUP)



Team Software Process (TSP)

is a detailed process description for small teams, containing:

- details of activities
- QA
- …

Organizations implementing TSP announced:

- productivity improvements of 25% or more
- reductions in cost and schedule variance to less than +/- 10%
- testing costs and schedule reductions of up to 80%

Capability Maturity Model Integration (CMMI)

CMMI can be used to guide process improvement across a project, a division, or an entire organization

Processes are rated according to their maturity levels

CMMI best practices are published in documents called models, each of which addresses a different area of interest: development, acquisition, services

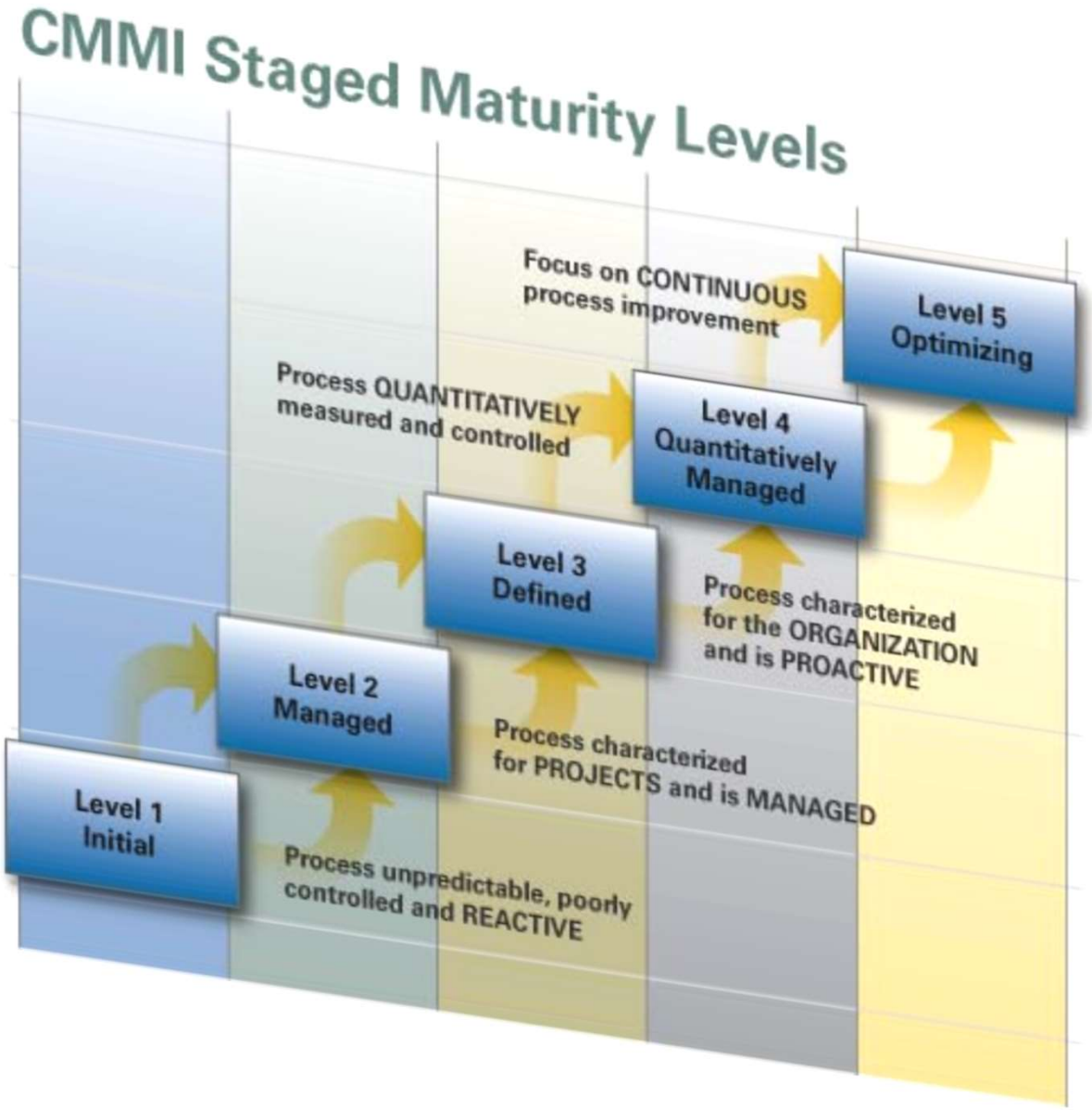
Capability Maturity Model describes how good an organization is in its processes

Starts from unpredictable and reactive (ad-hoc) processes

Ends with processes focused on the continuous improvement

http://www.sei.cmu.edu/cmmi/

CMM Levels


CMMI Staged Maturity Levels
Level 1
Initial
Level 2
Managed
Level 3
Defined
Level 4
Quantitatively
Managed
Level 5
Optimizing
Process unpredictable, poorly
controlled and REACTIVE
Process characterized
for PROJECTS and is MANAGED
Process characterized
for the ORGANIZATION
and is PROACTIVE
Process QUANTITATIVELY
measured and controlled
Focus on CONTINUOUS
process improvement

Key Process Area (KPA)

Each maturity level has some associated KPAs

KPAs are from 4 categories: project management, process management, engineering, support

The relationship is:

Maturity level => KPA

Not vice versa!!!

Table 3.2 Process Areas and Their Associated Categories and Maturity Levels

Process Area	*Category*	*Maturity Level*
Causal Analysis and Resolution	Support	5
Configuration Management	Support	2
Decision Analysis and Resolution	Support	3
Integrated Project Management +IPPD	Project Management	3
Measurement and Analysis	Support	2
Organizational Innovation and Deployment	Process Management	5

CMMI's relationship to quality

Level 1 "expects" the following process areas

- Quality assurance (**PPQA**)
- Measurement and analysis

Level 2 "expects" the following process areas

- Verification
- Validation

Level 3 "expects"

- Automation, including automated measurement processes

PPQA's main specific goals related to QA

SG1: **Objectively** Evaluate Processes and Work Products

- SP1.1: Objectively Evaluate processes
- SP1.2: Objectively evaluate work products and services

SG2: Provide **objective** insight

- SP2.1: Communicate and Ensure Resolution of Noncompliance issues
- SP2.2: Establish records

Examples of measurement data

- Estimated/planned vs. actual data on software size, cost, and schedule
- Productivity data
- Coverage and efficiency of peer reviews

Organization's measurement program includes

- Definition of the organization-wide measurements
- Collection of the organization's measurement data
- Analysis of the organization's measurement data
- Quantitative measurement goals for the organization

Checking the CMMI level is done ‘manually’ during an assessment

- Has to be done by an accredited auditor
- Is supported by methods like SCAMPI (Standard CMMI Appraisal Method for Process Improvement)

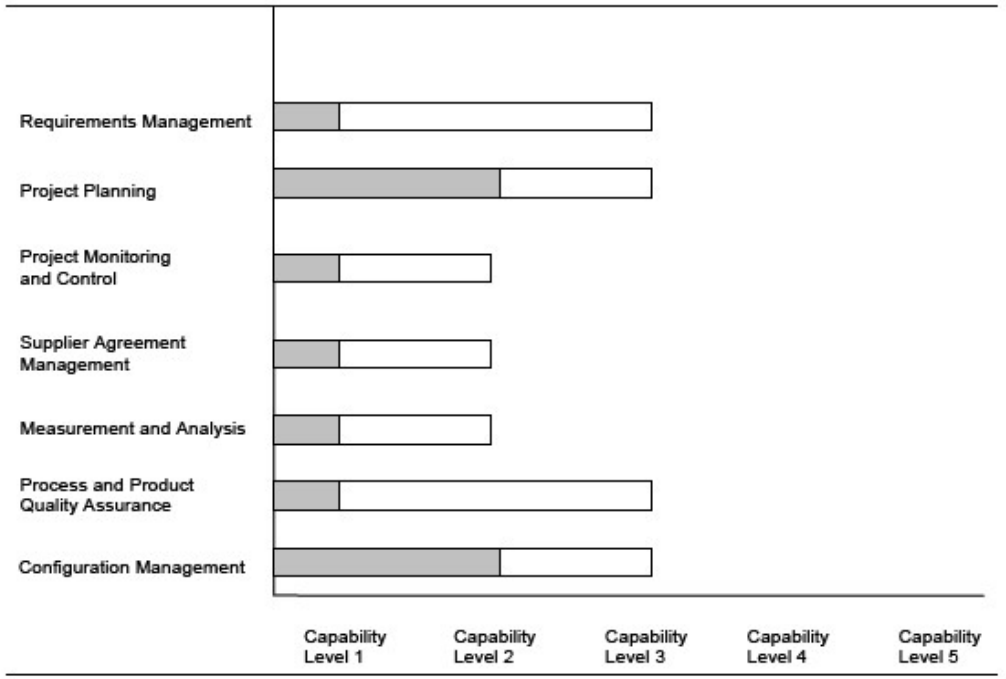


Figure 3.4: An Example of an Achievement Profile and a Target Profile

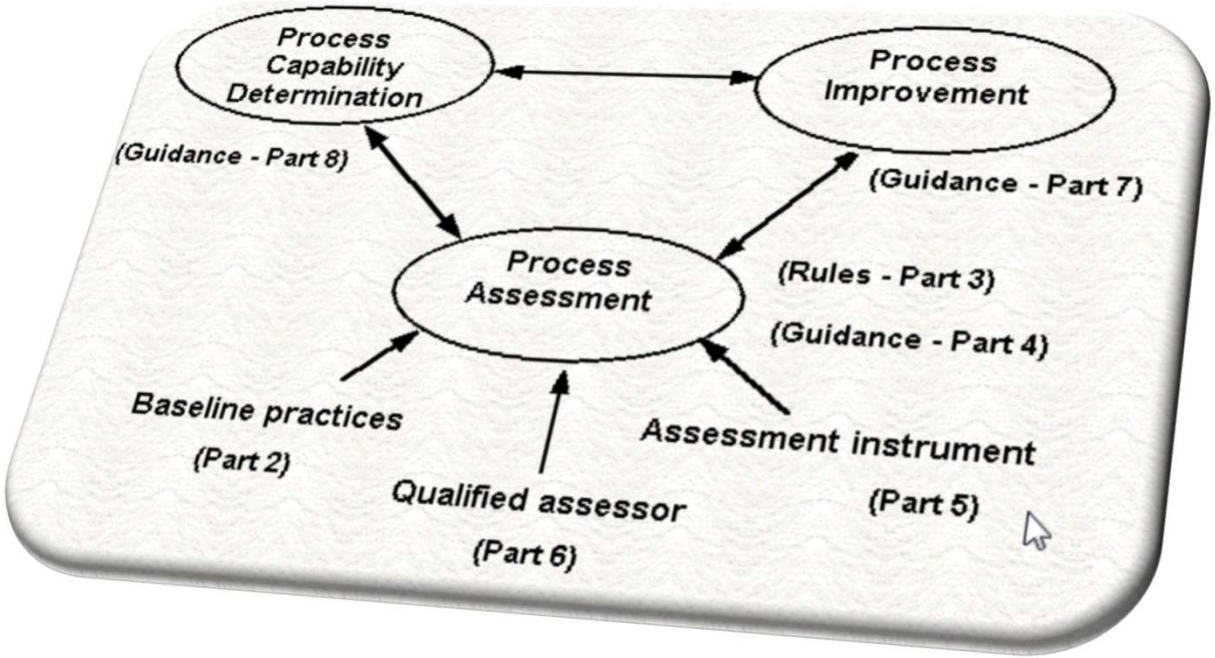
Practical considerations about CMMI

- CMMI is a comparison framework and a descriptive maturity level framework
  - If you want to improve your organization, CMMI does not tell you HOW to do it
  - CMMI describes areas which the organizations should consider
- CMMI level 5 does NOT guarantee that the software will be free of errors
  - The projects might be on budget, requirements correct, testing correct, but…
    - … one needs to know the “Garbage in – garbage out” principle
- CMMI is very good and well-known, but not the only framework like this
- CMMI is very high level
  - If an organization wants to focus on a specific area, say QM, then CMMI is not very helpful

SPICE

SPICE is a major international initiative to support the development of an International Standard for Software Process Assessment

- http://www.sqi.gu.edu.au/spice/what.html



How to improve processes?

Rational Unified Process has its own improvement framework:

http://www-01.ibm.com/software/rational/mcif/

Deming's PDCA: Plan-Do-Check-Act:

iterative problem solving process

PDCA Wheel

The Problem Solving Process

Act:

15. If the countermeasures were not effective, repeat the cycle beginning with the Plan phase

14. Observe the new current condition and set new targets towards the ideal condition, return to Plan

13. If the same countermeasures can be applied to similar problems to benefit others, do so

12. If the countermeasures were effective, make the new method a standard that can be audited and maintained

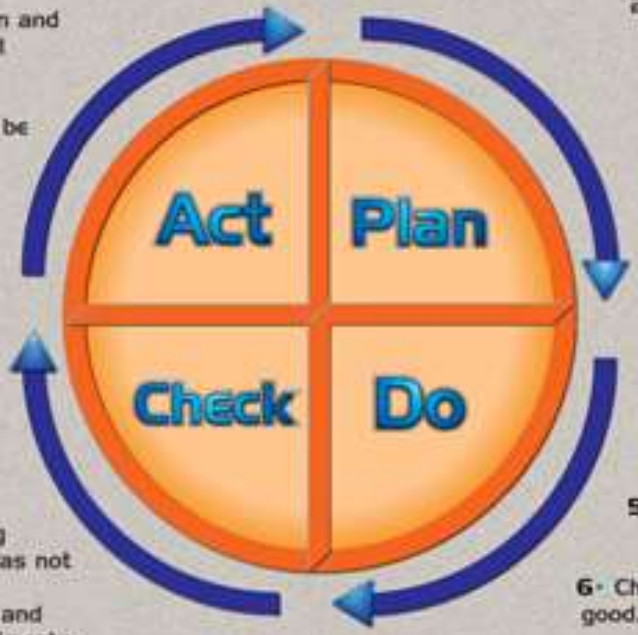
Plan:

1. Clearly and objectively state the problem

2. Give some background and context so that everyone can gain a common understanding

3. Conduct "5 Why" analysis to identify root causes

4. Brainstorm countermeasures and create hypotheses to test them



Check:

11. In each case, ask "Why?" until there is a clear understanding of what was effective and what was not

10. Reflect carefully on what worked and what did not work with the experiments

9. Face the facts

Do:

5. Follow the scientific method to test the hypotheses

6. Change things, and if the change is not good, change it again until it is better (try-storming)

7. Rather than waiting for one perfect solution, try many small things that can be done quickly

8. Gather the facts (Data) based on direct observation of the experiment

KAIZEN Products

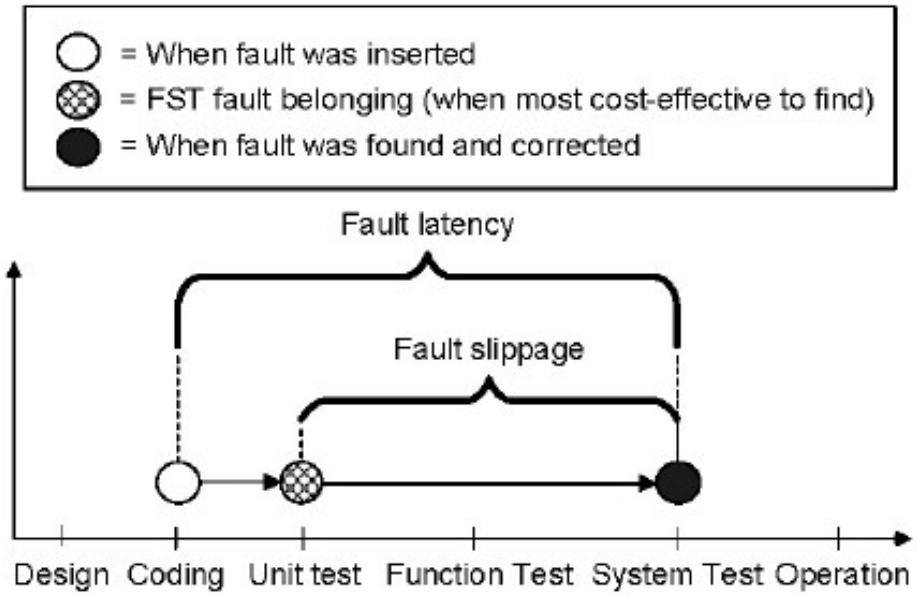
Keep the change.

www.kaizenproducts.com

Measures of Software Process Quality: Example

Fault-Slip Through:

this measure helps to understand the efficiency of testing in each phase



$$\%\,\mathrm{FST\ (to\ phase\ X)} = \frac{\mathrm{SF}^{1}\,(\mathrm{X})}{\mathrm{TF}^{2}\,(\mathrm{X})} \qquad (1)$$

[1]SF=No. faults found in phase X that slipped from earlier phases

[2]TF= Total no. faults found in phase X

Conclusions so-far…

"The quality of a product is largely determined by the quality of the process that is used to develop and maintain it." (Shewhart, Juran, Deming and Humphrey), but…

Once the process is of a good quality this does not mean that the product is of a good quality

No product QA -> no quality

Process is not followed -> no quality

Quality Metrics

According to standard ISO/IEC 15939, measure (i.e. metric) is a

- ”variable to which a value is assigned as the result of measurement”

“Categories” of metrics:

- Simple
- Statistics
- Key performance indicators (KPI)

Measurement Systems: set of metrics that present the status and/or progress of a specific area

Can anything/everything be measured?

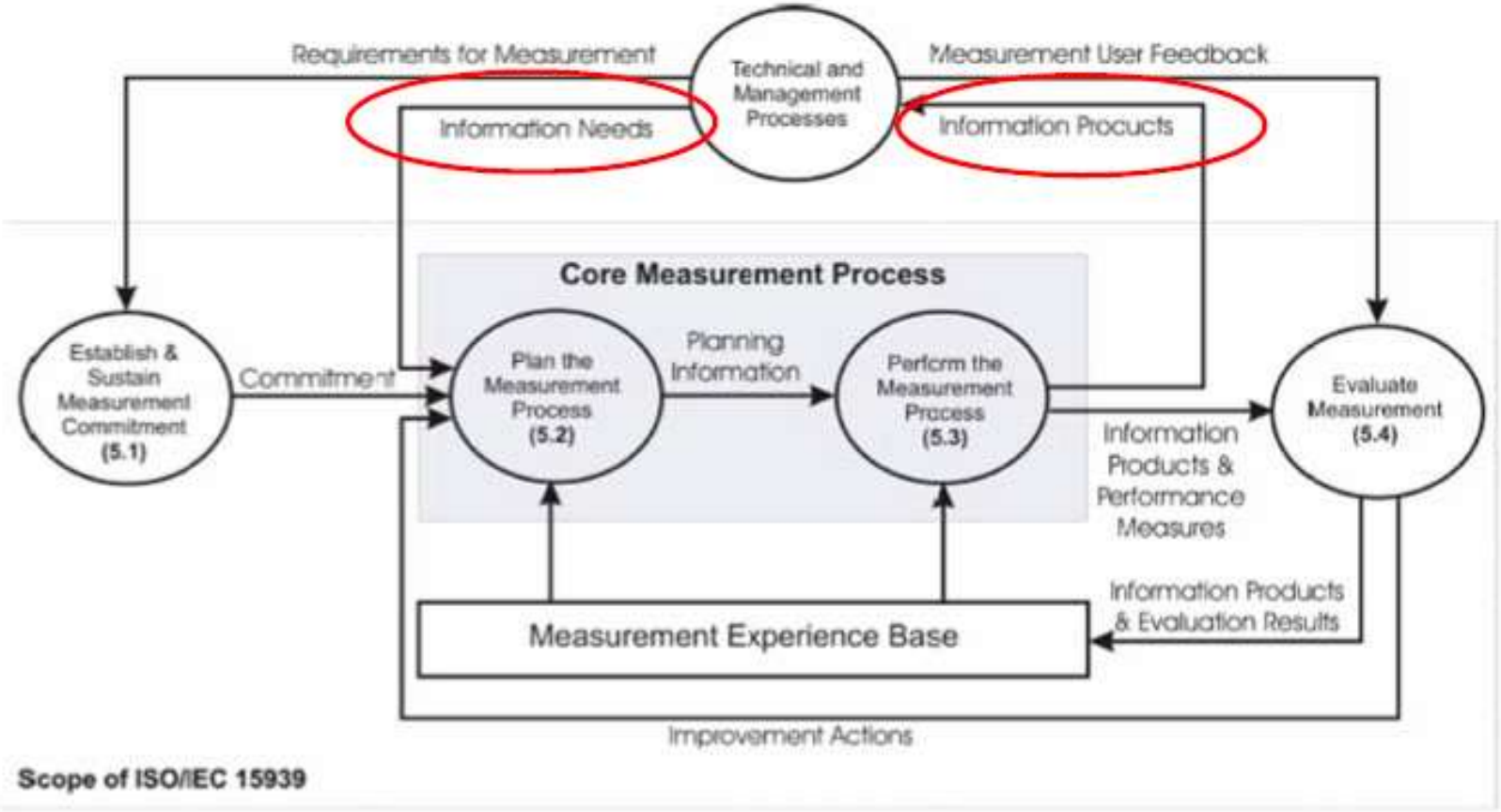
Can we trust metrics/statistics?

ISO/IEC 15939 (1)

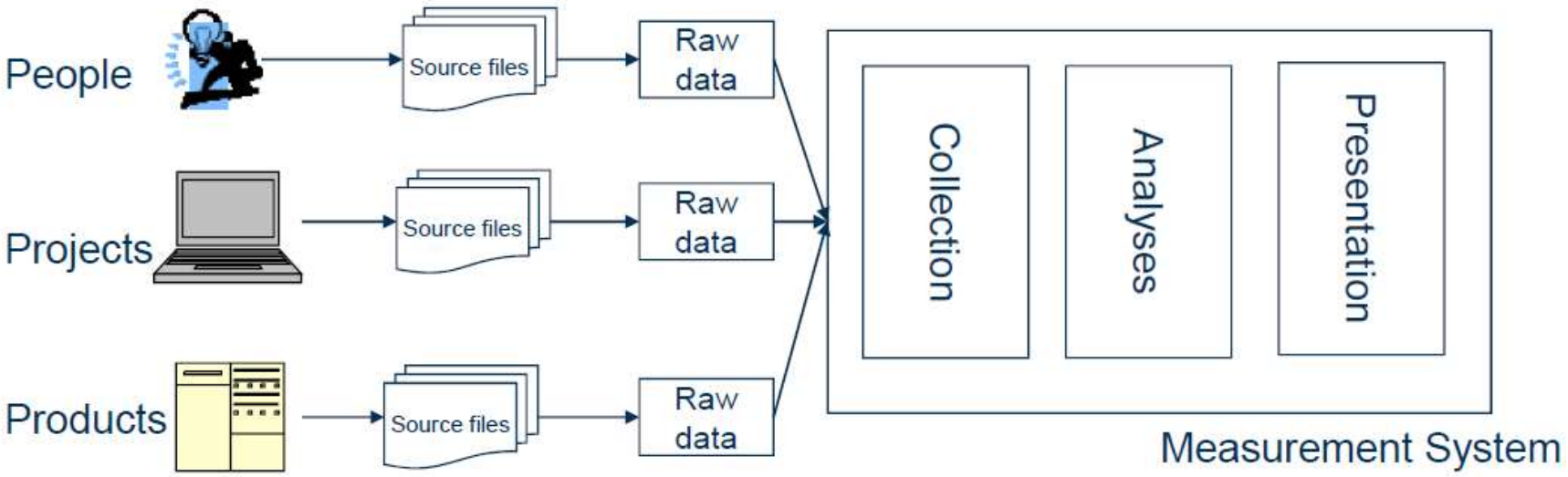
Engineering - Software Measurement Process is an international standard that defines a measurement process for software development and systems engineering:

”This international standard identifies the activities and tasks that are necessary to successfully identify, define, select, apply and improve measurement within an overall project or organizational measurement structure.”

ISO/IEC 15939 (2)



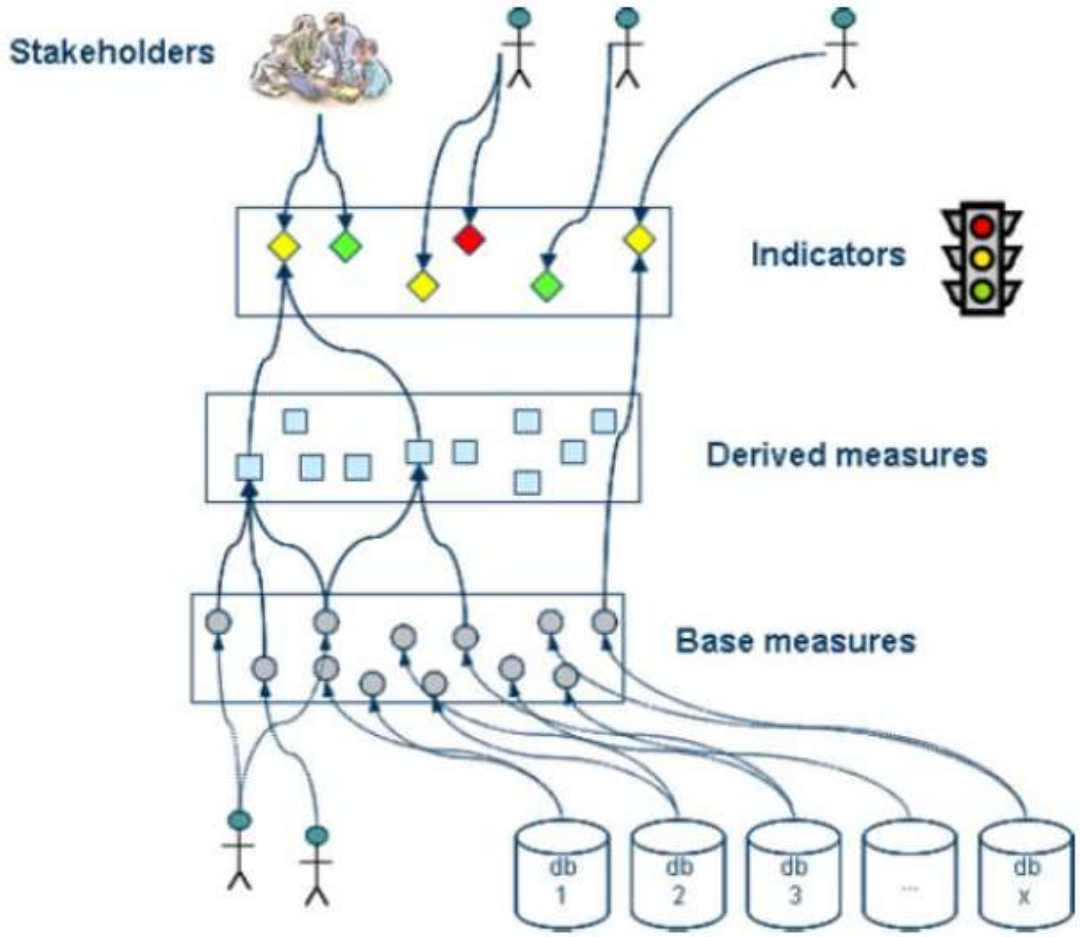
Measurement System: the Principles



Many data points

<10 data points

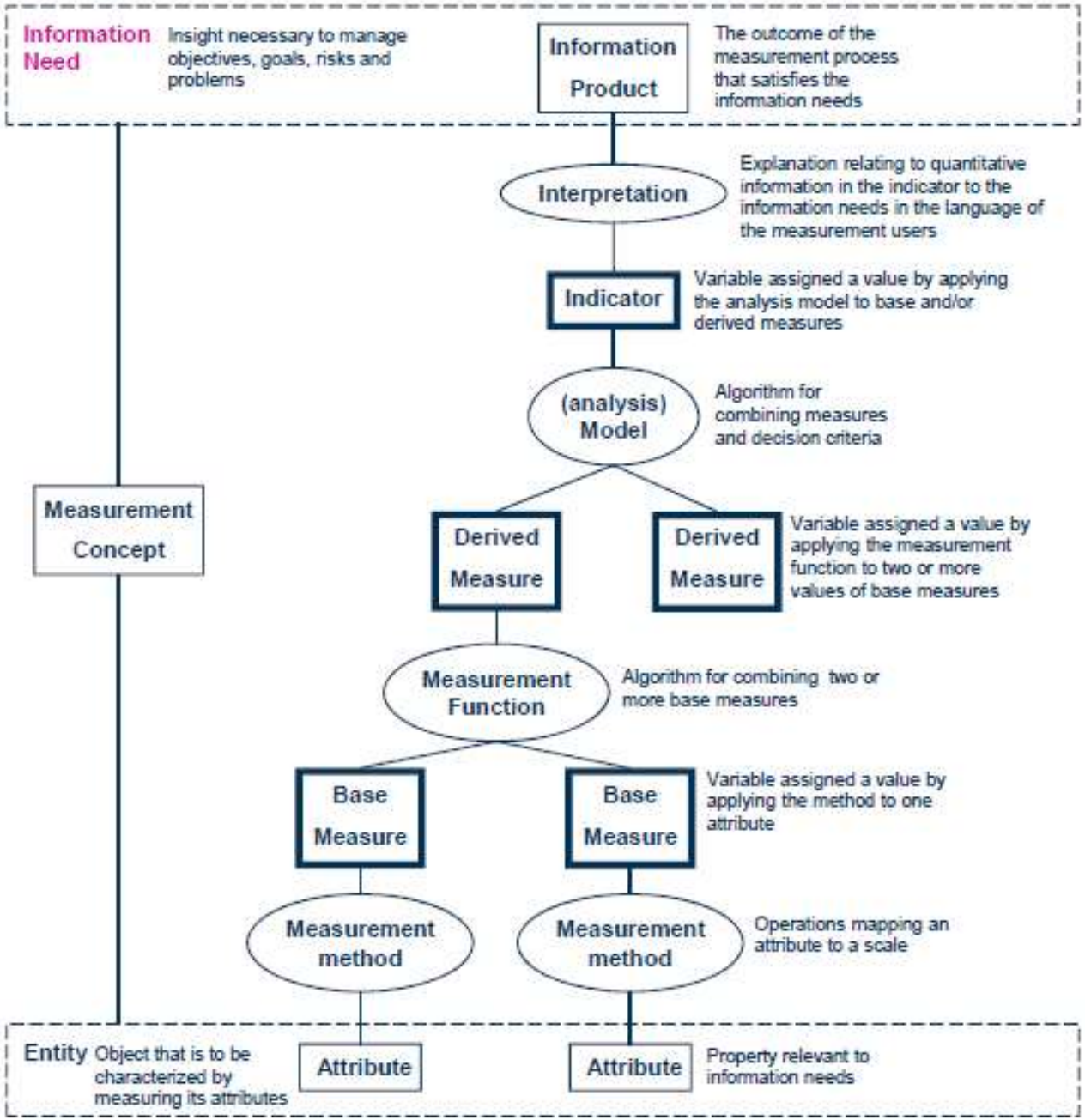
Measurement System: the Conceptual Model



Measurement Information Model

"is a structure linking information needs to the relevant entities and attributes of concern" – from Annex A (ISO/IEC 15939)

Defines measurement constructs which link information needs with attributes of entities


Information Need
Insight necessary to manage objectives, goals, risks and problems
Information Product
The outcome of the measurement process that satisfies the information needs
Interpretation
Explanation relating to quantitative information in the indicator to the information needs in the language of the measurement users
Indicator
Variable assigned a value by applying the analysis model to base and/or derived measures
(analysis) Model
Algorithm for combining measures and decision criteria
Measurement Concept
Derived Measure
Derived Measure
Variable assigned a value by applying the measurement function to two or more values of base measures
Measurement Function
Algorithm for combining two or more base measures
Base Measure
Base Measure
Variable assigned a value by applying the method to one attribute
Measurement method
Measurement method
Operations mapping an attribute to a scale
Entity
Object that is to be characterized by measuring its attributes
Attribute
Attribute
Property relevant to information needs

Indicator
Variable assigned a value by applying the analysis model to BM and/or DM

(analysis) **model**
Algorithm for combining measures and decision criteria

Derived Measure
Variable assigned a value by applying the MF to two or more values of BMs

Measurement Function
Algorithm for combining two or more BM

Base Measure
Variable assigned a value by applying the method to one attribute

Measurement Method
Operation relevant to information needs

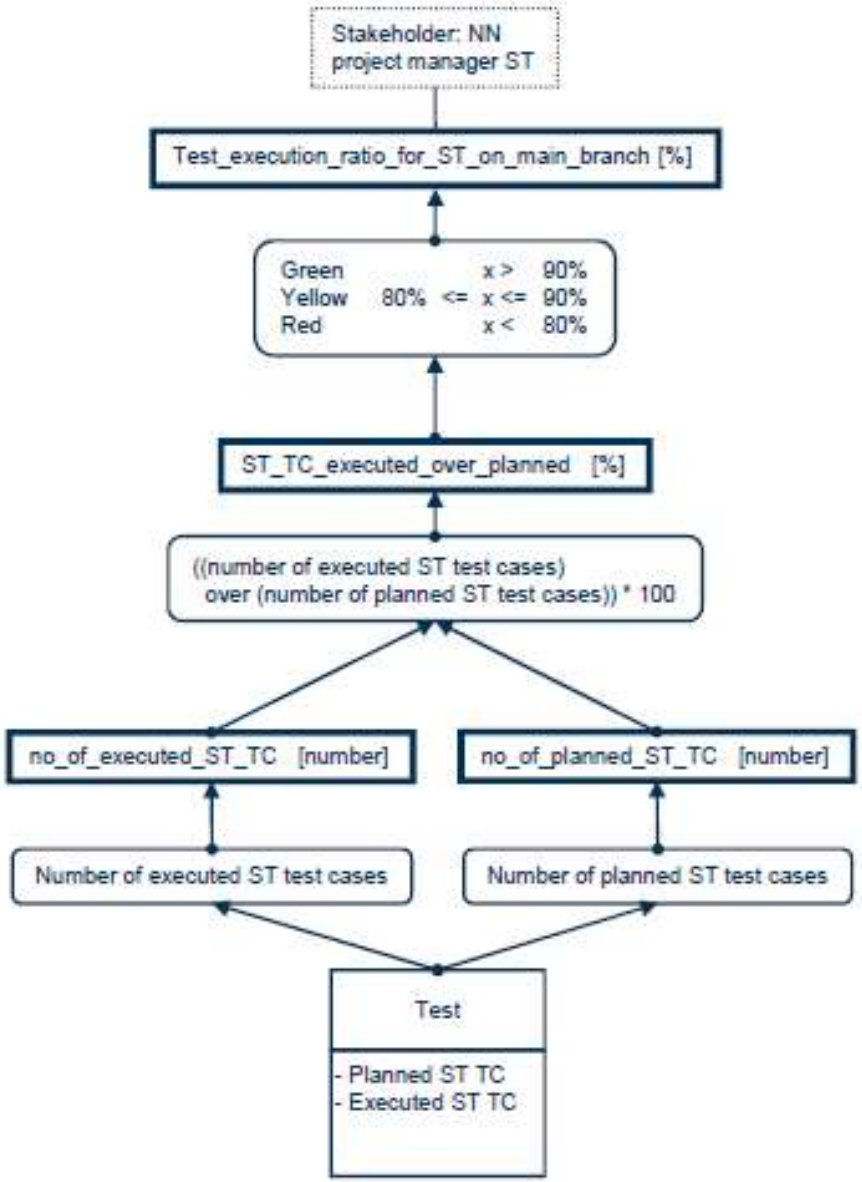
Entity
Object that is to be characterized by measuring its attributes

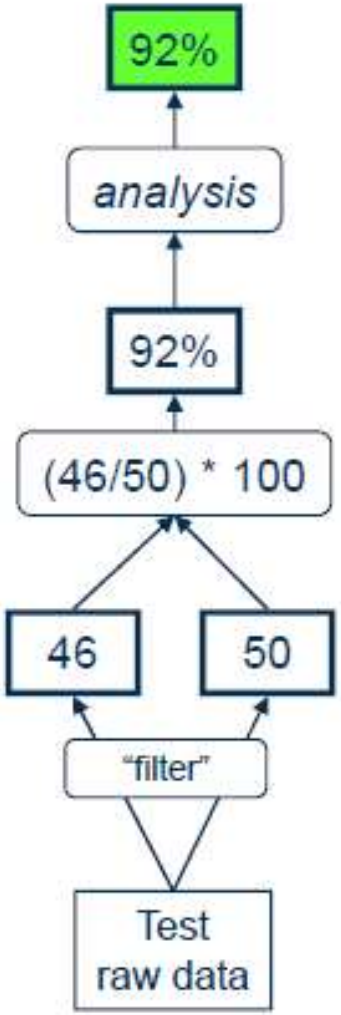
Indicators

Derived Measures

Base Measures

Entities



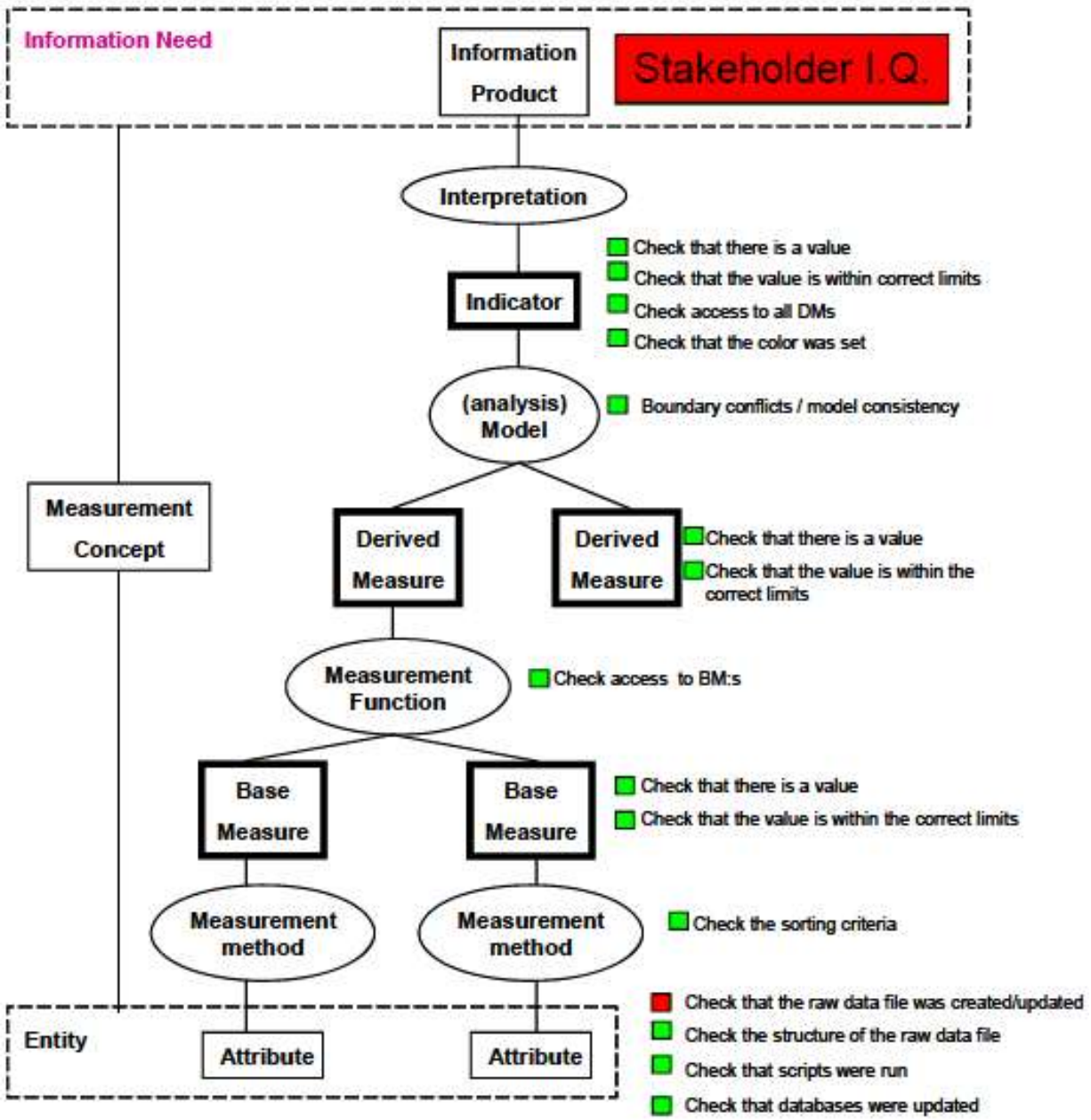


Main Benefices of Using Measurement Systems

New means of communication: common language, based on an ISO standard

Automation (data collection, analysis, presentation)

Reliability (assured by check-indicators)


Information Need
Information Product
Stakeholder I.Q.
Interpretation
Check that there is a value
Check that the value is within correct limits
Check access to all DMs
Check that the color was set
Indicator
(analysis) Model
Boundary conflicts / model consistency
Measurement Concept
Derived Measure
Derived Measure
Check that there is a value
Check that the value is within the correct limits
Measurement Function
Check access to BM:s
Base Measure
Base Measure
Check that there is a value
Check that the value is within the correct limits
Measurement method
Measurement method
Check the sorting criteria
Entity
Attribute
Attribute
Check that the raw data file was created/updated
Check the structure of the raw data file
Check that scripts were run
Check that databases were updated

Metrics for calculating the reliability (as a quality attribute from McCall's Quality Model):

average time between failures, average time for recovery after failures, average downtime per month)

Custom metrics:

for an Educational Recommender System

For a NetTraffic Interpreter

Requirements Management

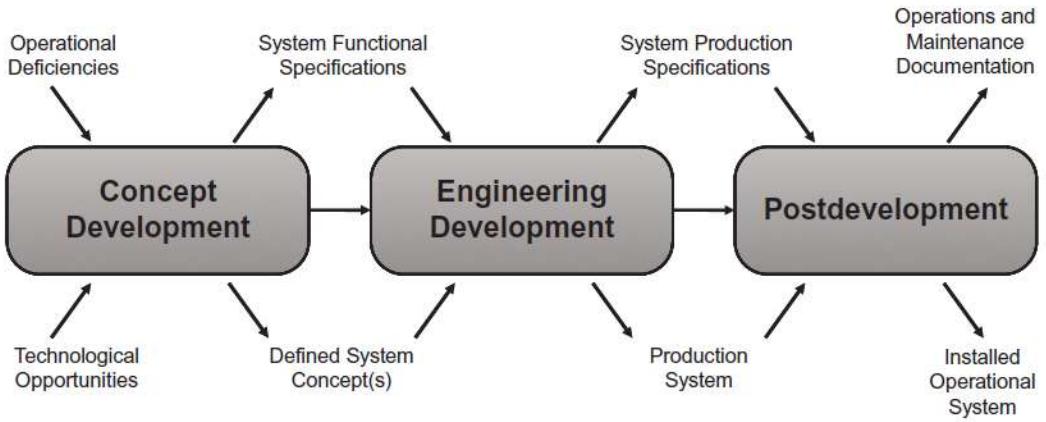
Preliminary Definitions

- System
  - An integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements. (International Council on SE - INCOSE SE Handbook)
- Systems Engineering
  - An interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem. (INCOSE)
- Requirement
  - specifies a capability or condition that must (or should) be satisfied, a function that a system must perform, or a performance condition a system must achieve.
- Specification
  - group of similar requirements

SE Life Cycle Model

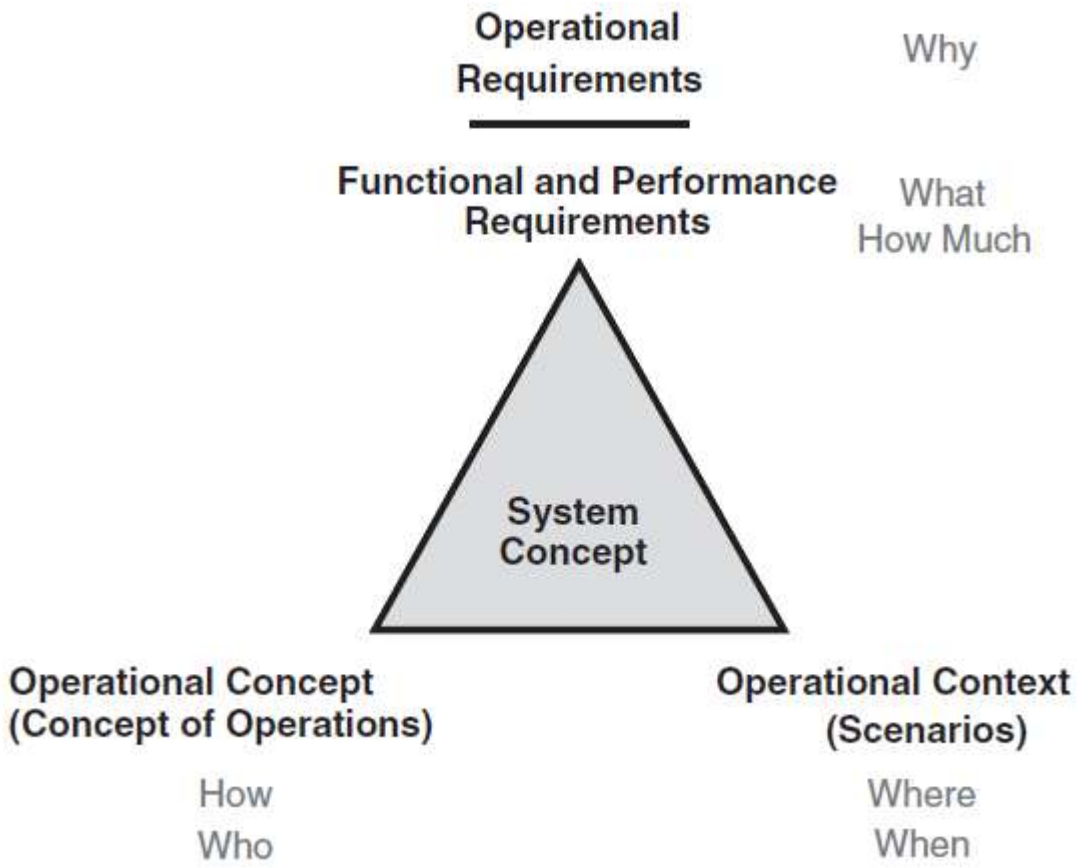
- Partitioned into 3 stages and 8 distinct phases:
- Concept development
  - Needs analysis
  - Concept exploration
  - Concept definition
- Engineering development:
  - Advanced development
  - Engineering design
  - Integration and evaluation
- Postdevelopment:
  - Production
  - Operations and support

- Inputs and outputs between processes:



Source: SE Principles and Practice, A. Kossiakoff & all

Conceptual Design of the System



Types of Requirements

- Operational: describe the **overall objectives** of the system
- Functional: describe the tasks that the system should perform during its operation
- Physical: describe appearance, volume, weight, power, other general constrains of the system
- Performance: should provide minimal numerical thresholds for the capabilities of the system

SE Challenge related to Requirements

- that requirements are consistent (not contradictory) and feasible
- that requirements have been validated to adequately reflect real stakeholder needs
- that requirements have been verified to ensure that they are satisfied by the system design

Definition(s) of quality

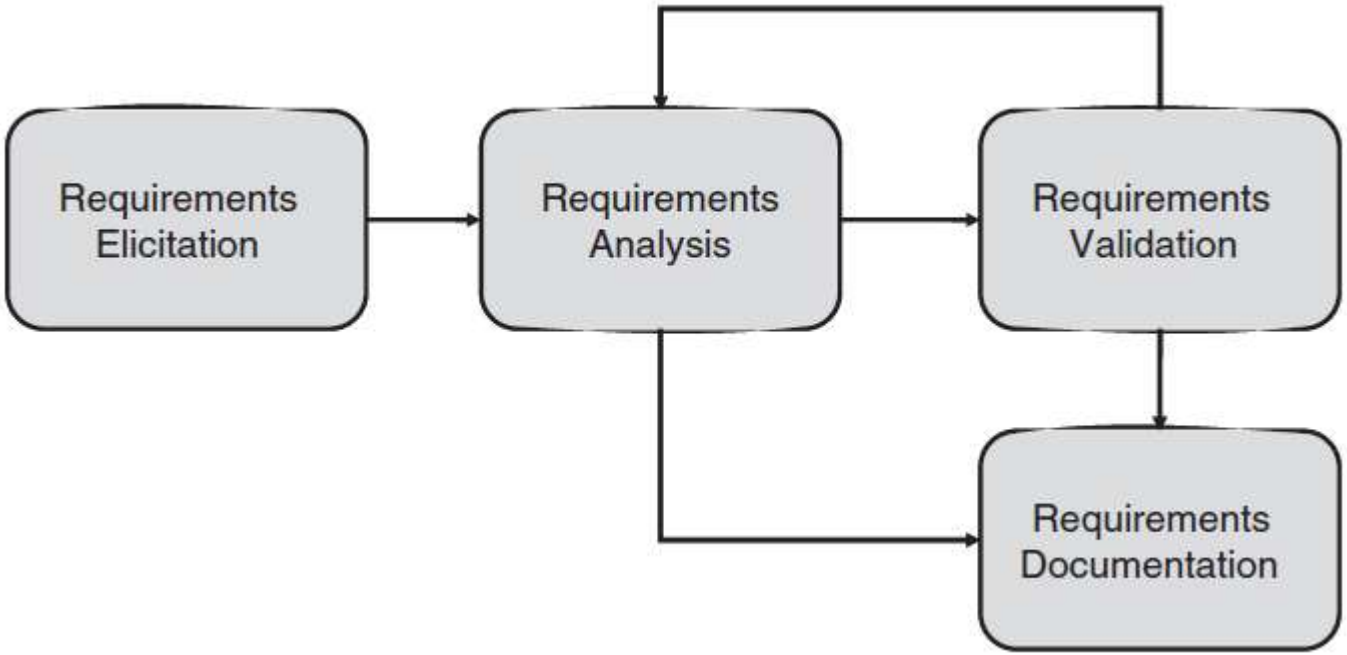
- IEEE (IEEE_Std_610.12-1990) defines product quality as:
  - the degree to which a system, component, or process meets specified **requirements**/ customer or user needs or expectations
- Pressman(*) defines the quality as:
  - conformance to explicitly stated functional and performance **requirements**, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software

(*) Pressman, 2004: Software Engineering: a practitioner's approach

Requirements Management Process

- Problem definition
- Current approaches to solving that problem
- Project scope
- Stakeholders and their needs
- **System requirements development**
- Create traceability between requirements
- Requirements prioritization
- Requirements assignment
- Change management of the requirements

System Requirements Development



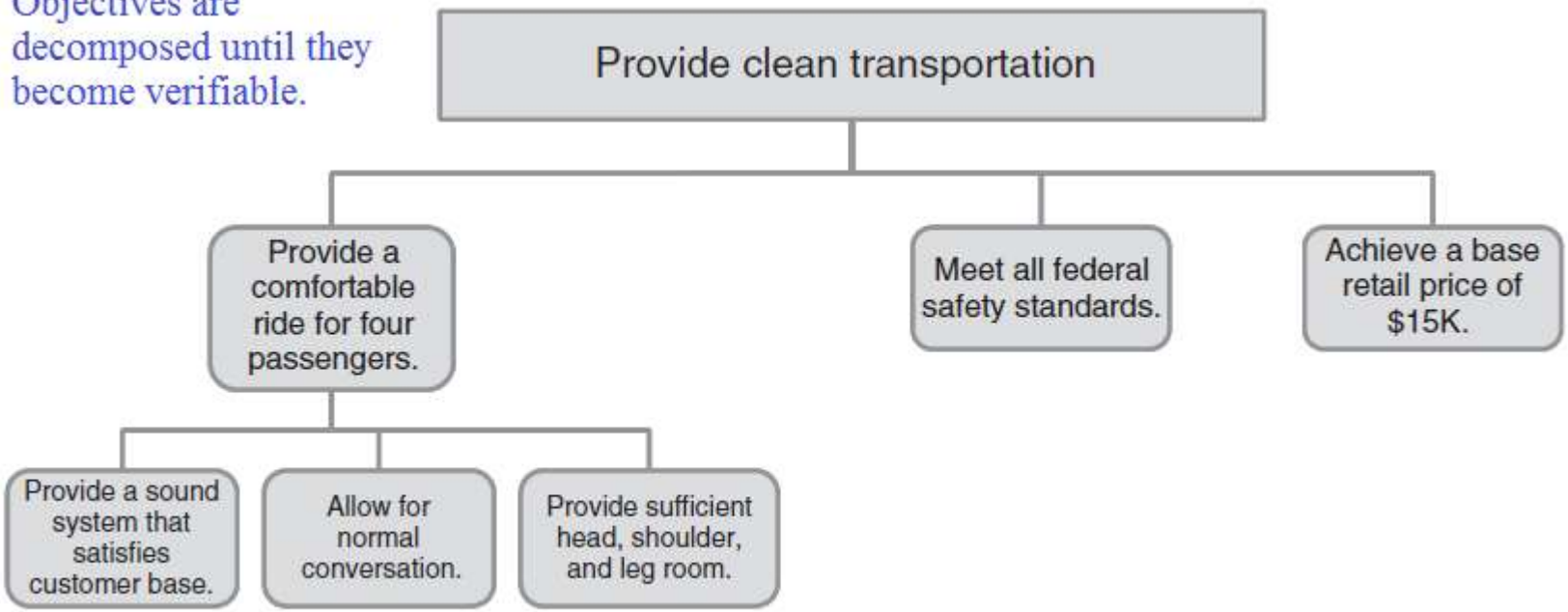
Requirements Analysis

- Is the requirement traceable to a user need or operational requirement?
- Is the requirement redundant with any other requirement?
- Is the requirement consistent with other requirements?
- Is the requirement unambiguous and not subject to interpretation?
- Is the requirement technologically feasible?
- Is the requirement affordable?
- Is the requirement verifiable?

- Does the set of requirements cover all of the user needs and operational requirements?
- Is the set of requirements feasible in terms of cost, schedule, and technology?
- Can the set of requirements be verified as a whole?

Objectives tree

Objectives are decomposed until they become verifiable.



Example objectives tree for an automobile.

Operational Effectiveness Model

- Used in estimating the degree to which a given system concept may be expected to meet a set of postulated operational objectives
- Based on
  - a mathematical model of the operational environment: a set of scenarios – postulated actions that represent a range of possible encounters to which the system must react
  - the candidate system being analyzed

Measure of Effectiveness (MOE)

- Definition: a metric of a system's overall performance that indicates the degree to which it achieves it objectives under specified conditions.
- Components:
  - The metric itself
  - Its units
  - The context under which the metric applies

Exercise

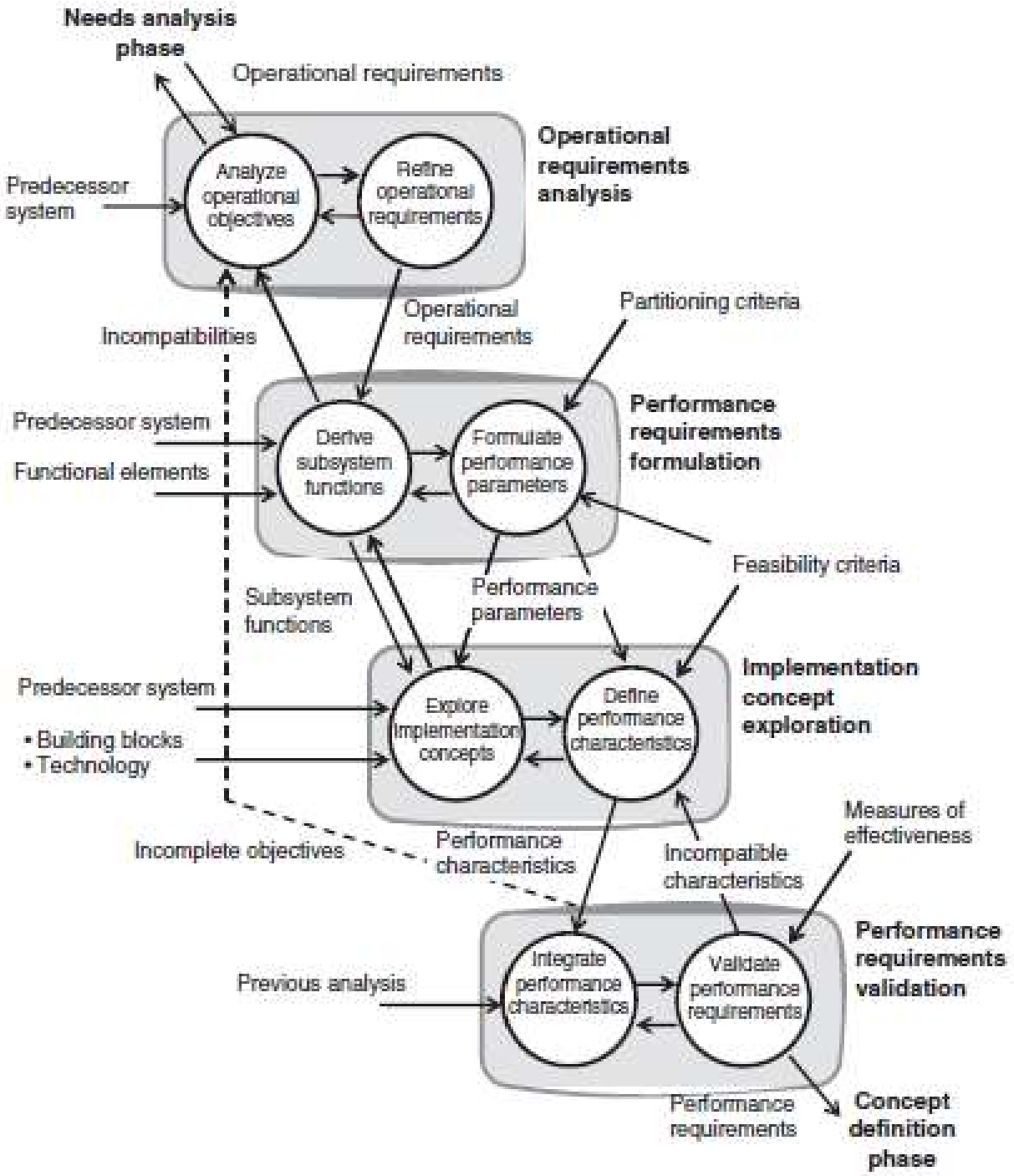
- For the effectiveness analysis of a sport utility vehicle (SUV), list 5 most important product characteristics that should exist and be measured in the analysis.

- For one of the characteristics, describe an operational scenario for obtaining a MOE.

Analysis of Alternatives

- Definition of a range of alternative system approaches to the general operational mission and a comparative evaluation of their MOE
- Steps for defining alternative concepts:
  - start with the existing system as a baseline
  - partition the system into its major subsystems
  - postulate alternatives that replace one or more of the subsystems essential to the mission with an advanced, less costly, or otherwise superior version
  - vary the chosen subsystems singly or in combination
  - consider modified architectures, if appropriate
  - continue until you have a total of *four to six meaningful* alternatives

Formulation of performance requirements



Important Aspects to Requirements Management

Requirements:

- provide early assurance that all top-level requirements are fully satisfied in the product, with *traceability* to where they are satisfied
- prevent unintentional addition of cost (avoid 'gold plating')
- establish clear responsibility for requirements implementation
- provide clear visibility across teams into requirements allocation and cross-functional interactions
- easily and quickly assess the impact of changes to requirements
- provide data for early and thorough validation and verification of requirements and design artefacts
- avoid unpleasant downstream surprises

Well-stated requirements exhibit the following attributes:

- The requirement is **Necessary:** What would happen if you didn't include this requirement?
- The requirement is **Verifiable:** How will you know you have met the requirement?
- The requirement is **Attainable**

Tools for Requirements Management

- IBM Rational Doors: http://www-142.ibm.com/software/products/us/en/ratidoor
- aNimble: http://sourceforge.net/projects/nimble/
- http://www.incose.org/productspubs/products/rmsurvey.aspx

http://rmblog.accompa.com/2012/05/requirements-management-tools/ (4 groups of tools: enterprise-level, mid-market, entry-level, open-source)

Modeling Requirements with SysML

Document-based SE vs. Model-based SE

- Document-based SE:
  - the generation of textual specifications and design documents, in hard-copy or electronic file format, that are then exchanged between customers, users, developers, and testers

- Model-based SE:
  - the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases

Systems Modeling Languages

- Traditional models, based on structured analysis and design (SAAD):
  - focus on functions and standard block diagramming techniques
  - use the top-down decomposition of the system

- Models based on object-oriented analysis and design (OOAD):
  - focus on entities
  - use the bottom-up approach

SysML

- A graphical modeling language in response to the UML for Systems Engineering RFP
- developed by the OMG (Object Management Group), INCOSE and AP233 Working Group for ISO
- a UML Profile that represents a subset of UML2 with extensions (explained later-on)
- supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- ***a visual modeling language that provides s***emantics (meaning) and notation (representation of meaning)
- ***not a methodology or a tool***

The SysML Language Specification

- available since September 2007

- defines the SysML language concepts used to model systems

- derived from the Unified Modelling Language

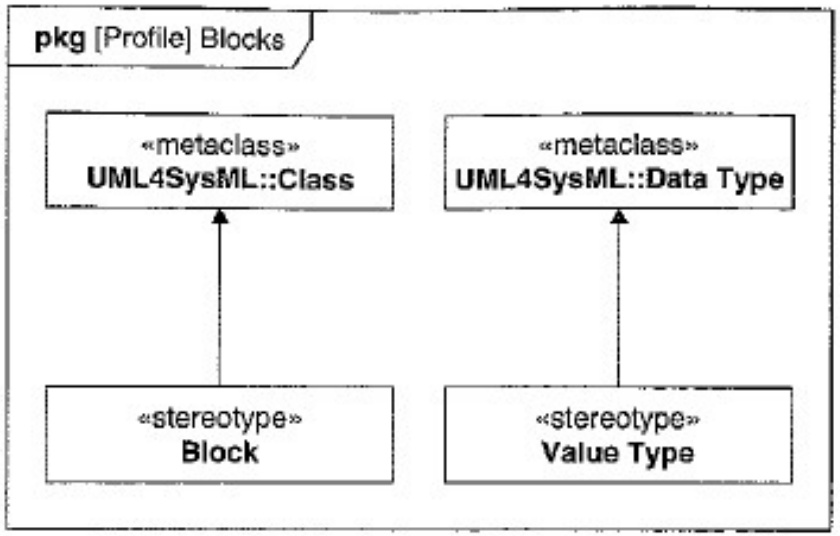
http://www.omg.org/spec/SysML/1.2/

The Architecture of SysML

- Domain concepts
- Meta-model: mapping of domain concepts to language concepts
- User-model: instantiation and representation of the language concepts as they apply to a particular system

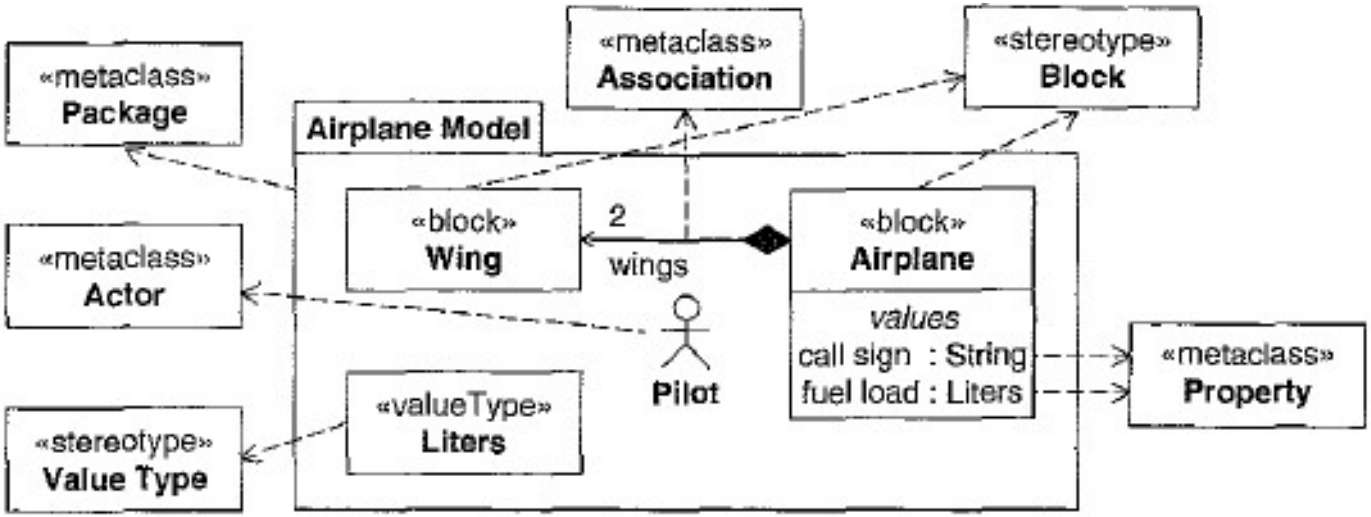
Meta-model

- A profile in UML is the mechanism used to customize the UML language.
- A profile contains stereotypes, which are used to extend the meta-classes from UML to create new/modified concepts.
- The SE extensions to UML in SysML are described using a profile called the SysML profile.

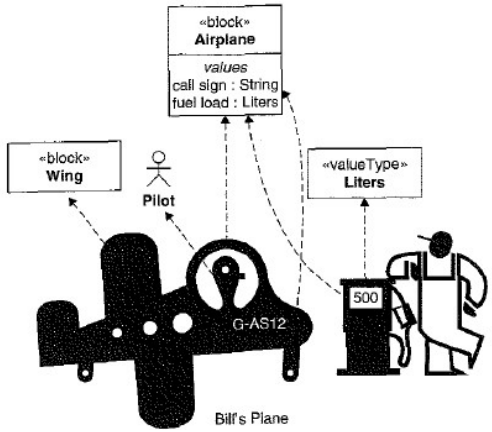


Fragment of SysML profile

User-model



Relating real-world concept to user-model concepts:



SysML Diagram Taxonomy

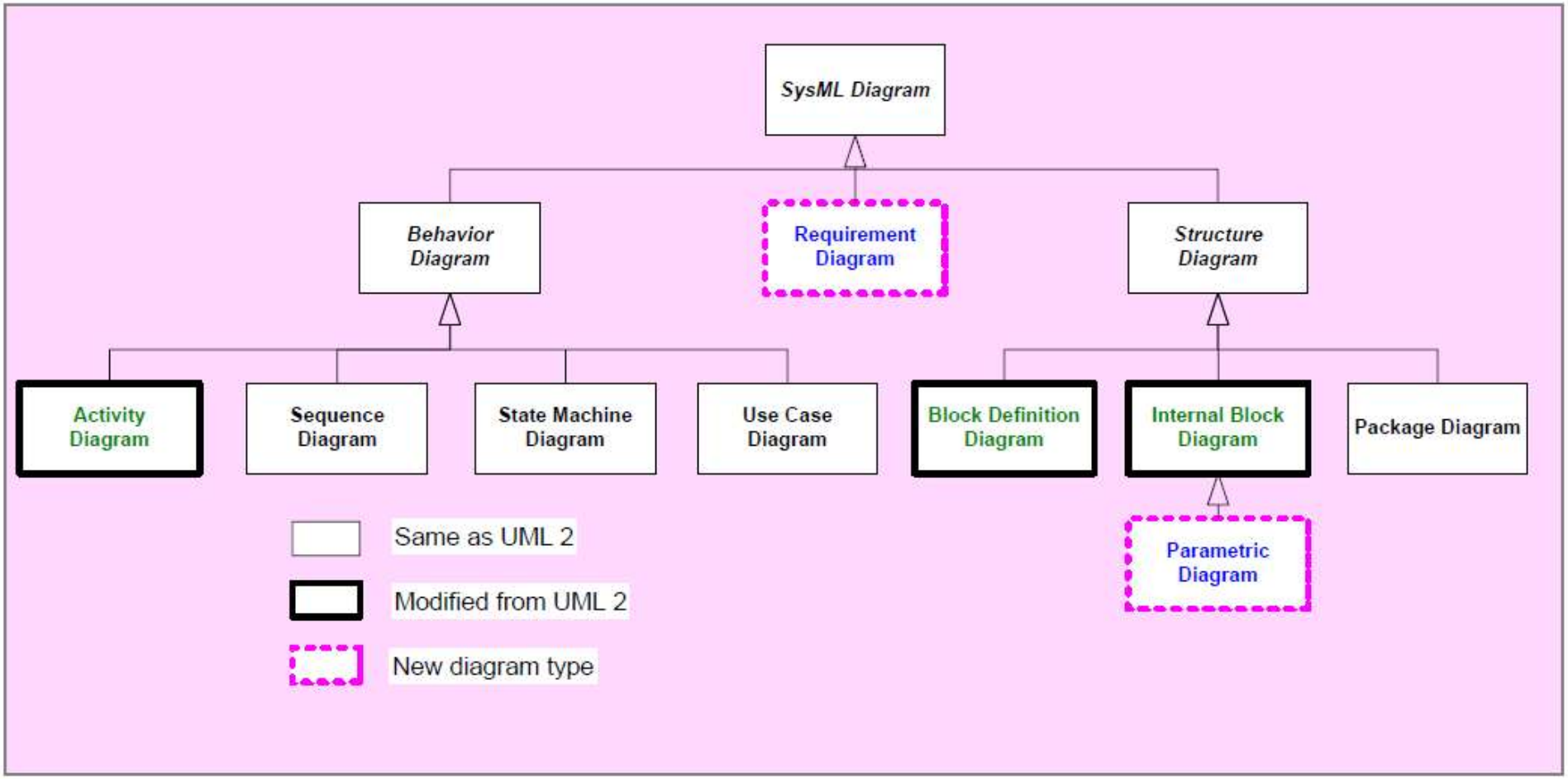
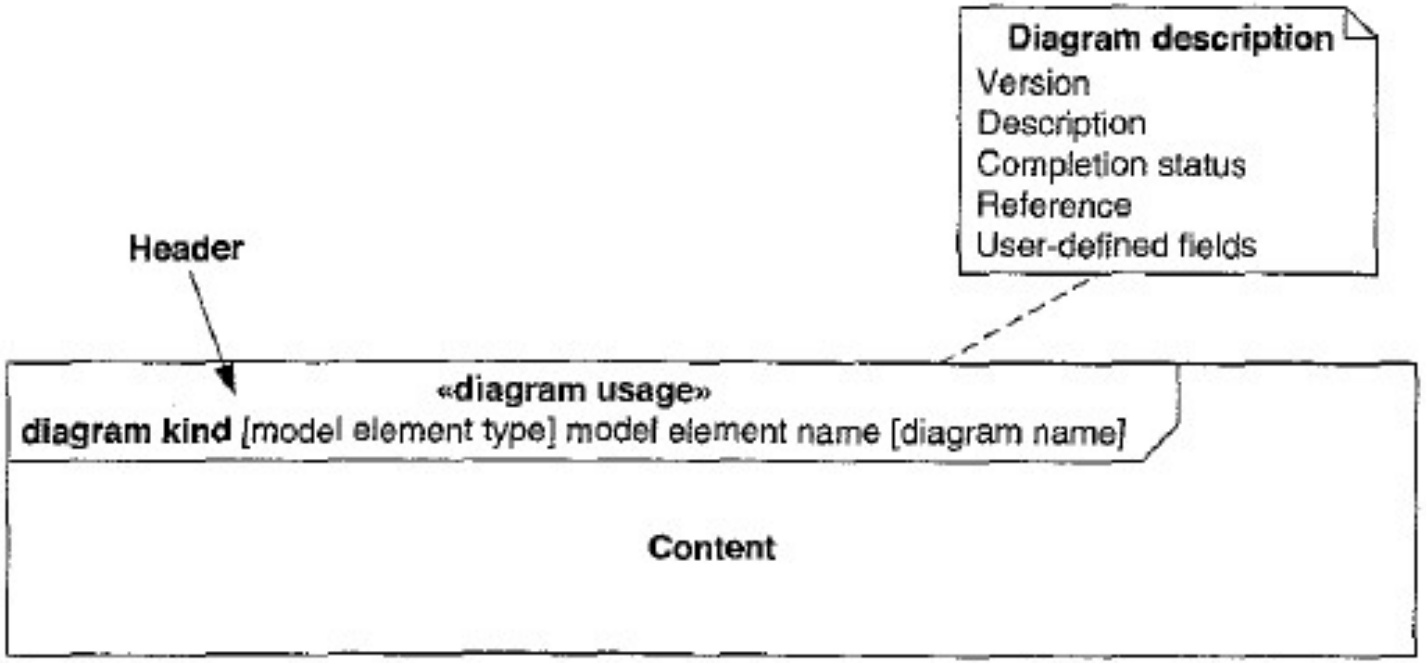


Diagram Frame



- Activity diagram-act
- Block definition diagram-bdd
- Internal block diagram-ibd
- Package diagram-pkg
- Parametric diagram-par

Requirement diagram-req
Sequence diagram-sd
State machine diagram-stID
Use case diagram-uc

Model Element Type

- Activity diagram-activity control operator
- Block definition diagram-block, constraint block, package, model, model library
- Internal block diagram-block
- Package diagram-package, model, model library, view
- Parametric diagram-block, constraint block
- Requirement diagram-package, model, model library, requirement
- Sequence diagram-interaction
- State machine diagram-state machine
- Use case diagram-package, model, model library

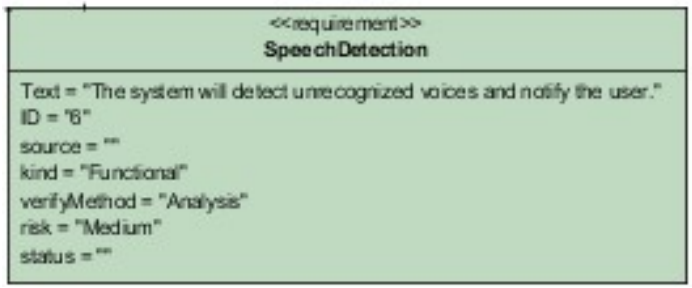
Other Notations

- Tables, matrices, trees (for large amount of information)

table [SafetyRequirement] System Specifications [Decomposition of system's specification]						
ID	Name	Kind	Text	Verify Method	Risk	Status
1	SafetyRequirement	Functional	The system should monitor the activity inside the house, block the access of any stranger and ensure user's safety 24/7 under all conditions.	Test	Medium	Mandatory
1.2	TimeCondition	Functional	The system must be safe and secure 24/7.	Inspection	Medium	Implemented
1.3	SecurityCondtion	Functional	The system must be safe and secure under all conditions.	Test	Medium	Mandatory
4	OpticalSensorDetection	Functional	The system will use video cameras to ensure security.	Demonstration	Low	Implemented
5	FingerPrintDetection	Interface	The system will use finger prints for login into the system.	Demonstration	High	Implemented
6	SpeechDetection	Functional	The system will detect unrecognized voices and notify the user.	Analysis	Medium	Implemented

SysML Requirements Diagram

- SysML can be used to model text-based requirements and relate them to other requirements and to other model elements.



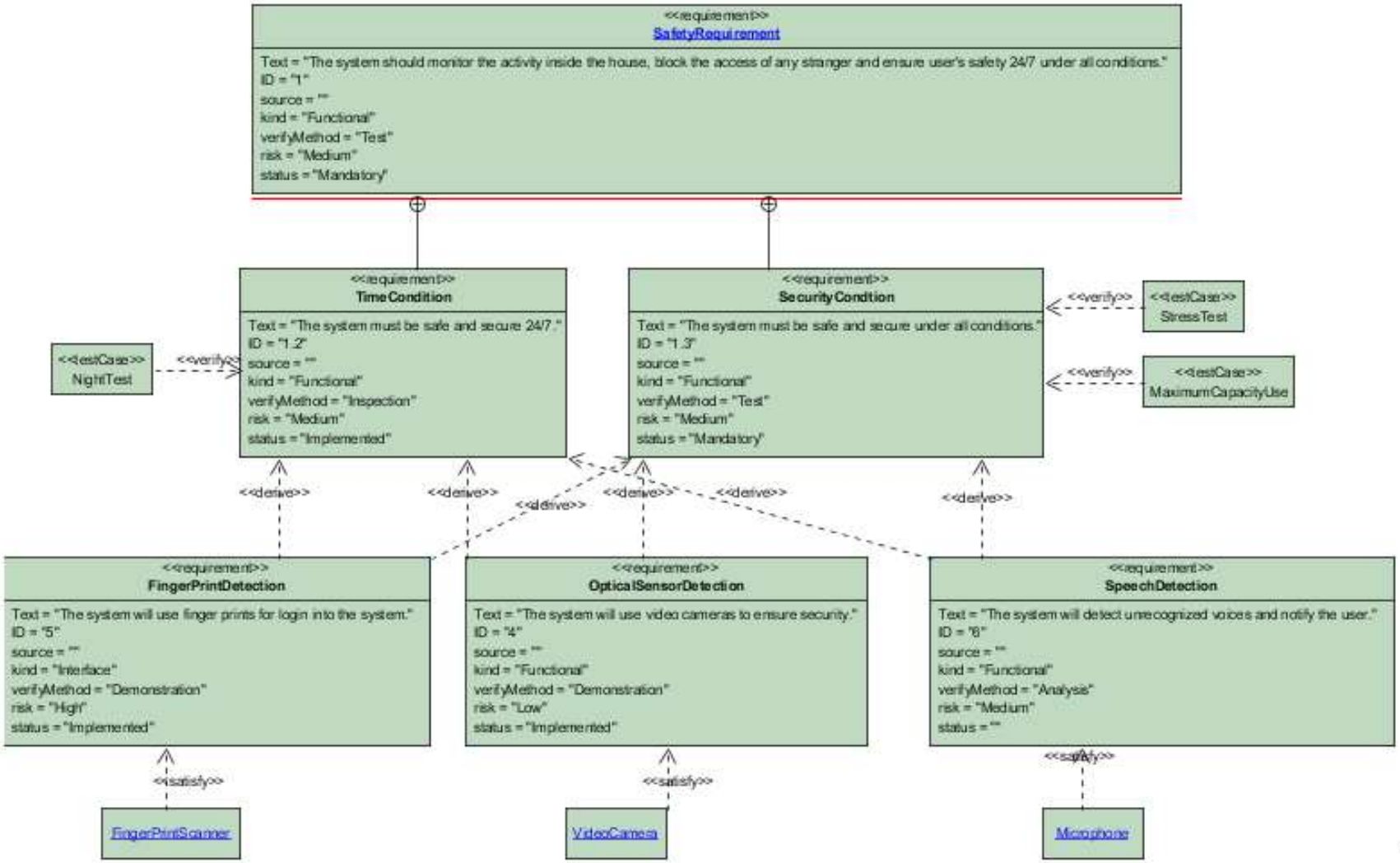
- Reqs Diagram
  - provides a bridge between the text-based requirements that may be maintained in a requirements management tool and the system model
  - keeps the requirements sync with the system model, assuring *traceability* through out the system lifecycle
  - header:

req [package or requirement] Model Element Name [diagram name]

- Requirements can be customized by adding additional properties such as verification status, criticality, risk, and requirements category.

- An alternative method for creating requirements categories is to define additional subclasses of the requirement stereotype (e.g. <<extendedRequirement>>, <<functionalRequirement>>, <<performanceRequirement>>, <<physicalRequirement>>, <<designConstraint>> , <<interfaceRequirement>>)

Types of Requirements Relationships (1)



Types of Requirements Relationships (2)

Relationship name	Keyword depicted on relation	Req callout	Client callout
Satisfy	**«satisfy»**	Satisfied by « ***model element»***	**Satisfies «requirement»**
Verify	**«verify»**	Verified by **«model element»**	**Verifies «requirement»**
Refine	**«refine»**	Refined by **«model element»**	**Refines «requirement»**
Derive requirement	**«deriveReqt»**	Derived **«requirement»**	Derived from **«requirement»**
Copy	**«copy»**	(None)	**Master «requirement»**
Trace	**«trace»**	Traced «**model element**»	Traced from **«requirement»**

Modelling a Requirements Containment Hierarchy

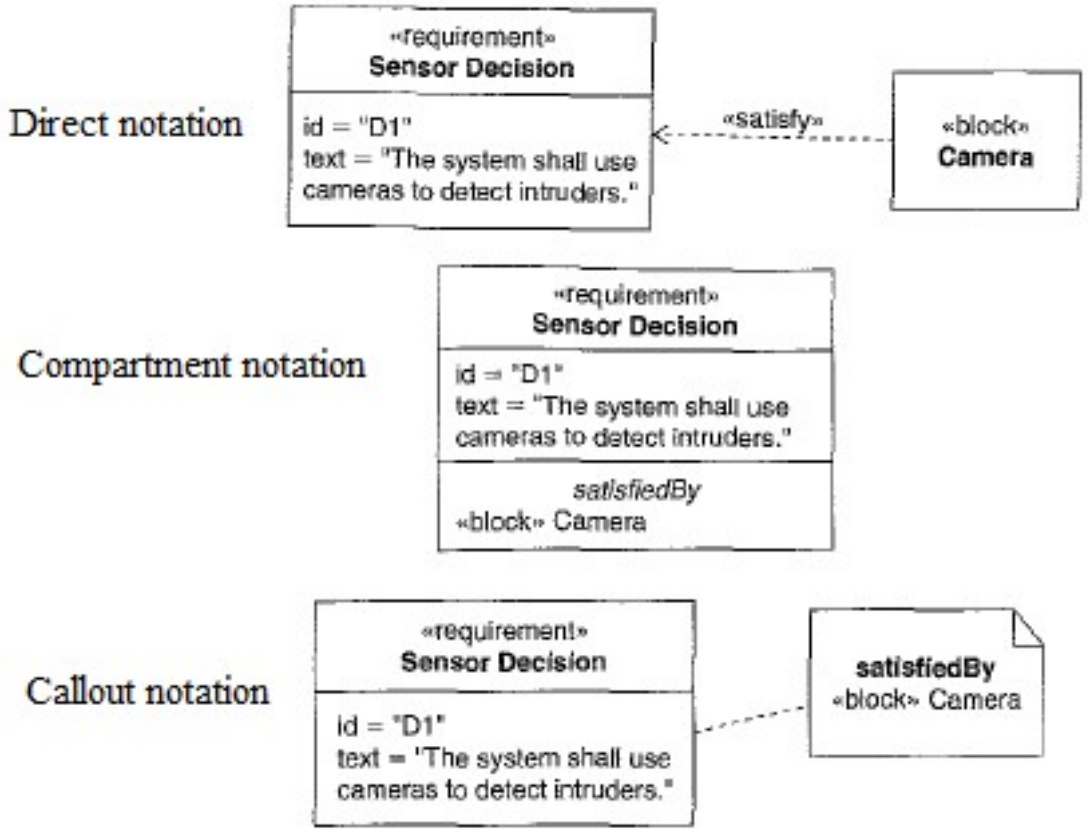
- The containment relationship is used to represent how a complex requirement can be partitioned into a set of simpler requirements without adding meaning or other implications.
- A containment relationship can be viewed as a logical and-ing (conjunction) of the contained requirements with the container requirement.
- The partitioning of complex requirements into simpler requirements is essential to establish full traceability and show how individual requirements are the basis for further derivation, and how they are satisfied and verified.

The Browser View of a Containment Hierarchy

table [SafetyRequirement] Relation [Requirements Tree]					
Id	Name	Relation	Id	Name	Rationale
4	OpticalSensorDetection	Derived From	1.2	TimeCondition	A video camera is a cost-effective way to ensure security.
		Derived From	1.3	SecurityCondition	A video camera is a cost-effective way to ensure security.
5	FingerPrintDetection	Derived From	1.2	TimeCondition	A finger print scanner is an innovative and optimal method to ensure security.
		Derived From	1.3	SecurityCondition	A finger print scanner is an innovative and optimal method to ensure security.

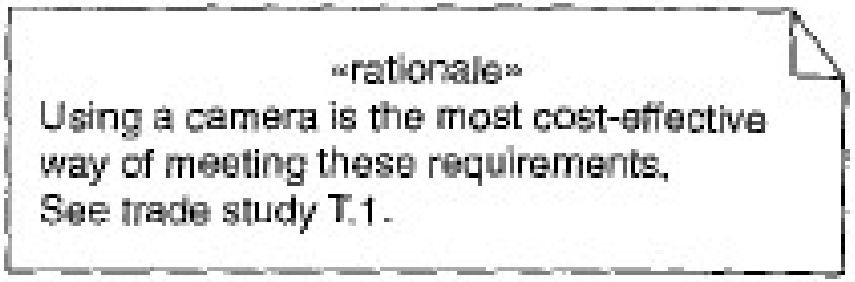
Relations between Reqs and Other SysML Elements

- shown directly (if all the elements are in the same diagram)
- shown by using the compartment or callout notation (if the elements are not in the same diagram)
- the arrow points from the dependent model element (called the client) to the independent model element (called the supplier): the camera design is dependent on the requirement, meaning that if the requirement changes, the design must change



Depicting Rationale for Requirements Relationships

- A rationale is a SysML model element that can be associated with either a requirement or a relationship between requirements.
- The rationale is intended to capture the reason for a particular design decision.



Depicting Requirements and their Relationships in Tables

table [SafetyRequirement] System Specifications [Decomposition of system's specification]						
ID	**Name**	**Kind**	**Text**	**Verify Method**	**Risk**	**Status**
1	SafetyRequirement	Functional	The system should monitor the activity inside the house, block the access of any stranger and ensure user's safety 24/7 under all conditions.	Test	Medium	Mandatory
1.2	TimeCondition	Functional	The system must be safe and secure 24/7.	Inspection	Medium	Implemented
1.3	SecurityCondtion	Functional	The system must be safe and secure under all conditions.	Test	Medium	Mandatory
4	OpticalSensorDetection	Functional	The system will use video cameras to ensure security.	Demonstration	Low	Implemented
5	FingerPrintDetection	Interface	The system will use finger prints for login into the system.	Demonstration	High	Implemented
6	SpeechDetection	Functional	The system will detect unrecognized voices and notify the user.	Analysis	Medium	Implemented

Exam Question

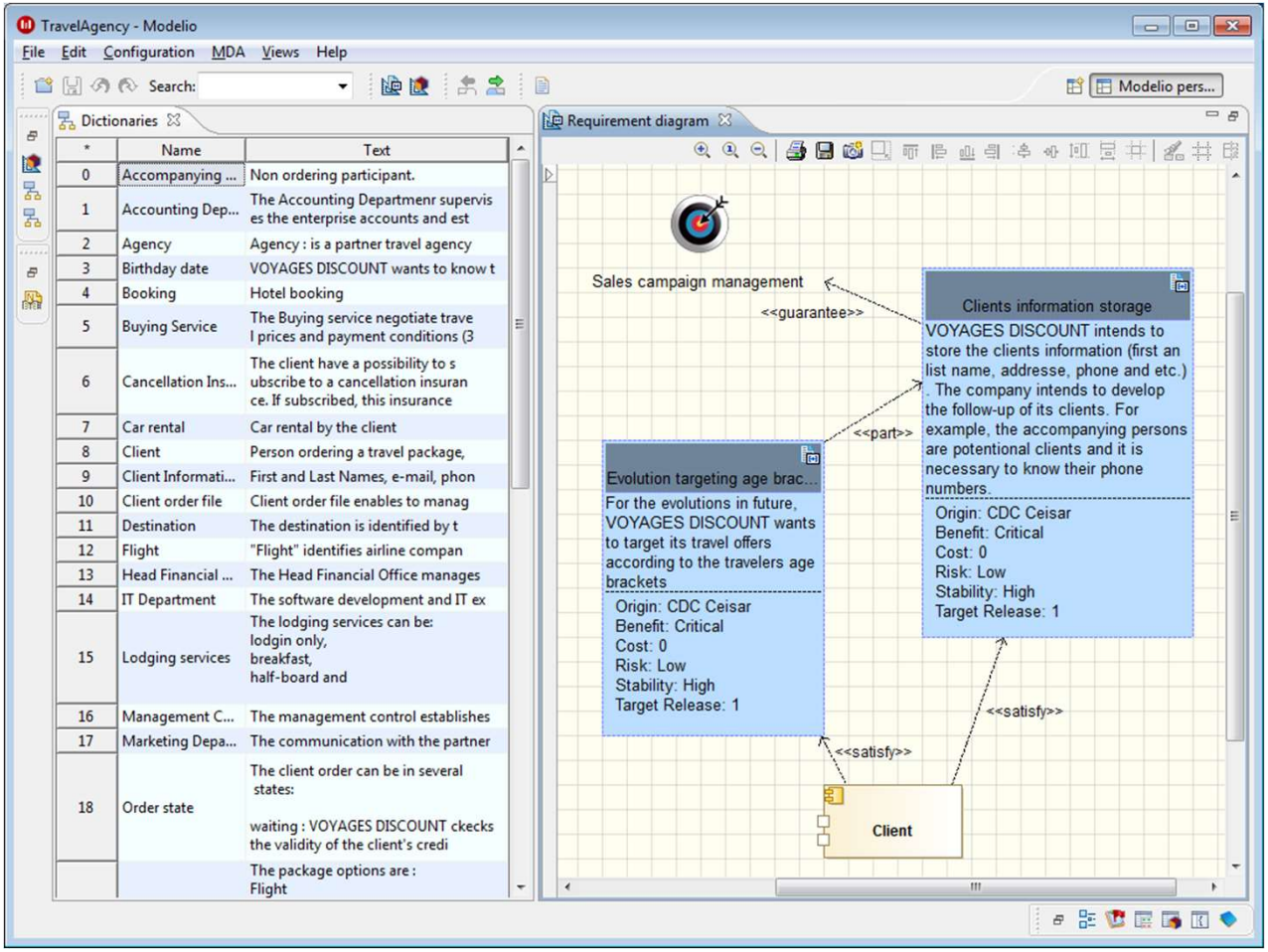
- What aren't requirement relationships in SysML?
  - Implement (check)
  - Refine
  - Satisfy
  - Extend (check)

SysML Tools

- Open source:
  - Modelio
  - Papyrus
- Comercial:
  - Magic Draw
  - Rational Rhapsody
  - Visual Paradigm
  - Astah
  - Sparx System Enterprise Architect

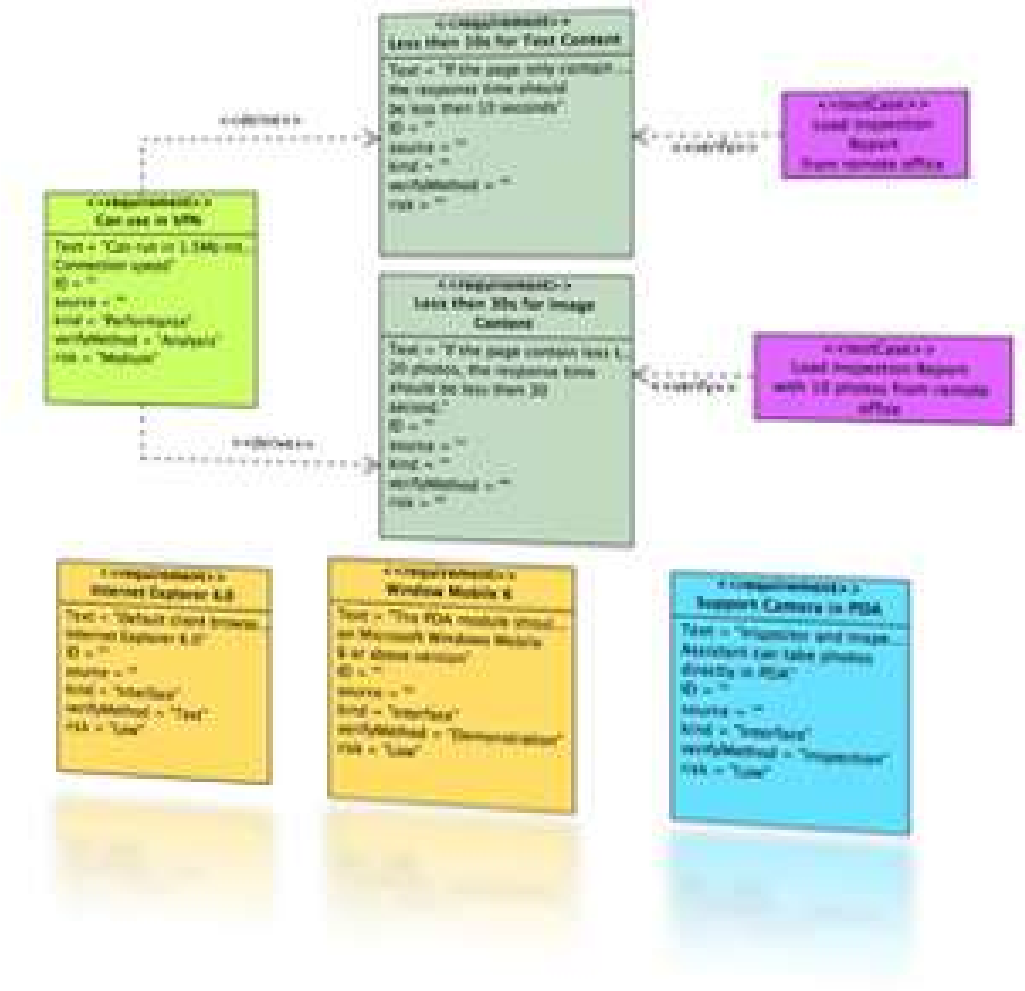
Modelio (with SysML Architect Module):

http://www.modelio.org/

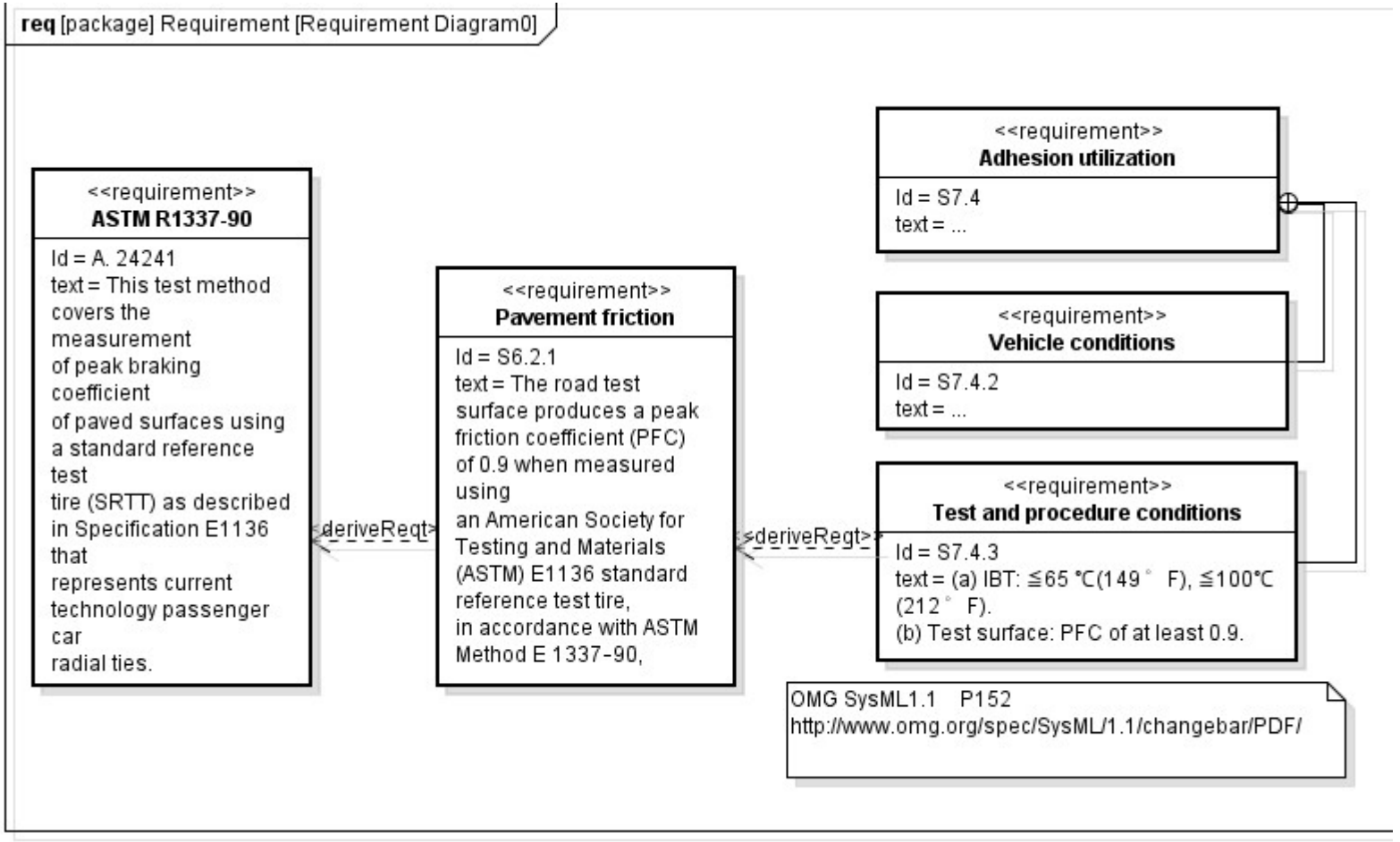


*	Name	Text
0	Accompanying ...	Non ordering participant.
1	Accounting Dep...	The Accounting Departmenr supervis es the enterprise accounts and est
2	Agency	Agency : is a partner travel agency
3	Birthday date	VOYAGES DISCOUNT wants to know t
4	Booking	Hotel booking
5	Buying Service	The Buying service negotiate trave l prices and payment conditions (3
6	Cancellation Ins...	The client have a possibility to s ubscribe to a cancellation insuran ce. If subscribed, this insurance
7	Car rental	Car rental by the client
8	Client	Person ordering a travel package,
9	Client Informati...	First and Last Names, e-mail, phon
10	Client order file	Client order file enables to manag
11	Destination	The destination is identified by t
12	Flight	"Flight" identifies airline compan
13	Head Financial ...	The Head Financial Office manages
14	IT Department	The software development and IT ex
15	Lodging services	The lodging services can be: lodgin only, breakfast, half-board and
16	Management C...	The management control establishes
17	Marketing Depa...	The communication with the partner
18	Order state	The client order can be in several states: waiting : VOYAGES DISCOUNT ckecks the validity of the client's credi
		The package options are : Flight

Visual Paradigm: http://www.visual-paradigm.com/ (trial version: http://www.visual-paradigm.com/download/vpuml.jsp)



AstahSysML: http://astah.net/editions/sysml



Overview of AstahSysML:

http://astah.net/tutorials/sysml/astah-window