

---

# Requirements Evolution Empirical Analyses

Massimo Felici

18 July 2001, ITC-IRST/ARS, Trento, Italy



# Overview

- Why Requirements Evolution?
- Empirical Requirements Evolution: Two Industrial Case Studies
- Discussion and Remarks
- An Empirical Framework for Requirements Evolution
- Conclusions and Further Work

# Requirements Evolution

- Fault recovery is *cost-effective* during requirements specification
- *Requirements changes* are due to many *stakeholders* and *environmental constraints*
- It is impossible to specify *right requirements* the first time
- *Environmental turbulence* gives rise to *requirements evolution*
- Requirements are *interdependent* on one another

# Requirements Evolution

- Requirements Evolution is due to stakeholder interaction
- Evolution may tackle system degradation
- Evolution is more general than maintenance
- Requirements Evolution is a orthogonal concept to dependability
- Process to product (requirements) engineering
- Requirements Evolution: tradeoff among Business, Process and Product viewpoints

# Empirical Requirements Evolution

- Empirical investigation of industrial case studies
  - Gathering
  - Measuring
  - Representing
- Linking Requirements Evolution to Dependability
- Requirements Evolution Engineering

# Research Rationale

- Software Engineering
  - ★ Iterative Development Processes
  - ★ Software Evolution
- Requirements Engineering
  - ★ Requirements Evolution
- Software Metrics
- Product Line Approaches

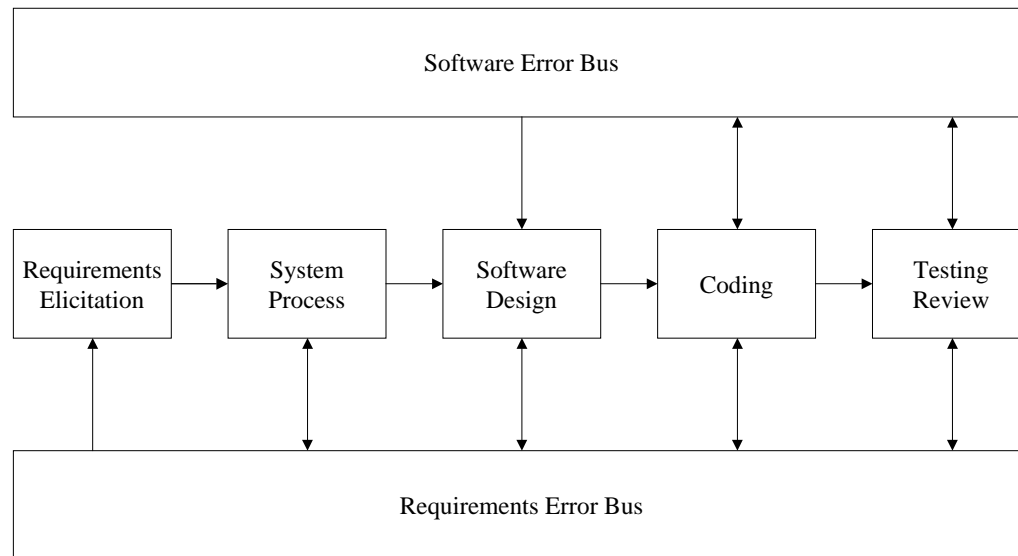
# Two Industrial Case Studies

- An **avionics** case study
- A **smart card** case study

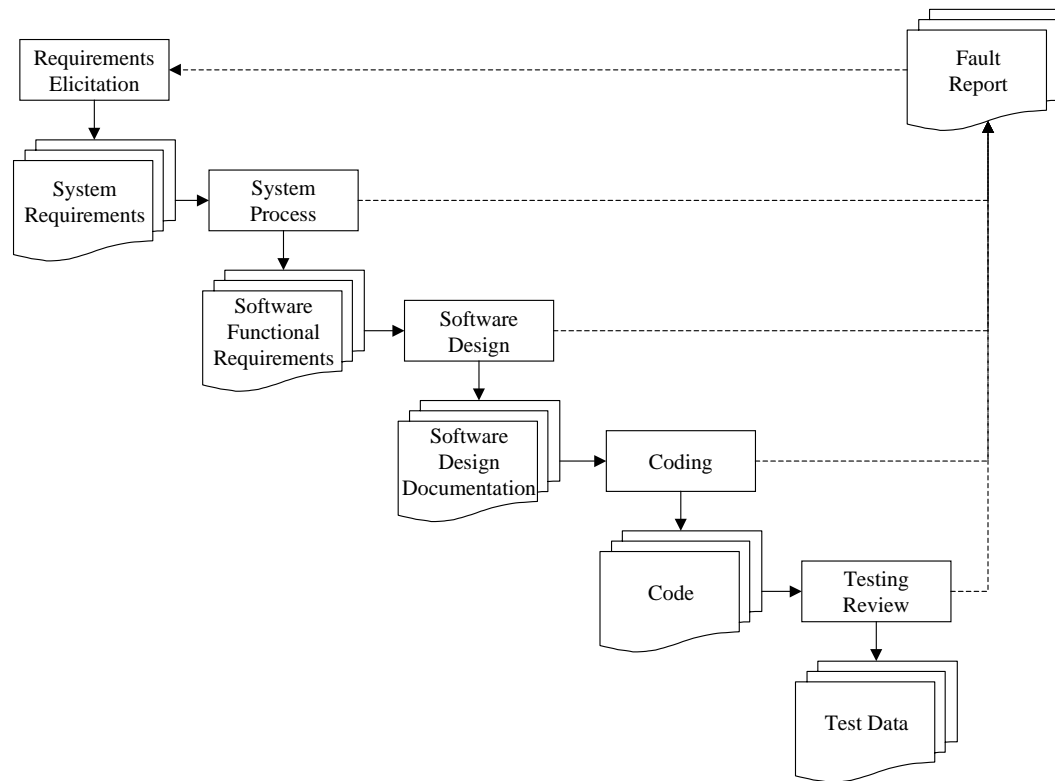
## An Avionics Case Study

- **Safety-critical** case study
- Requirements evolution over **22 software releases**
- DO-178B. Software Considerations in Airborne Systems and Equipment Certification. RTCA.
  - ★ Requirements **changes**: **added**, **deleted** and **modified**
  - ★ **Measure** of requirements evolution
  - ★ **Orthogonal analysis**: **types of changes**, **functional analysis**
  - ★ **Architecture stability**
  - ★ **Functional requirements dependency**
  - ★ **Graphical representation** of **Requirements Evolution**
  - ★ **Evolving paths**

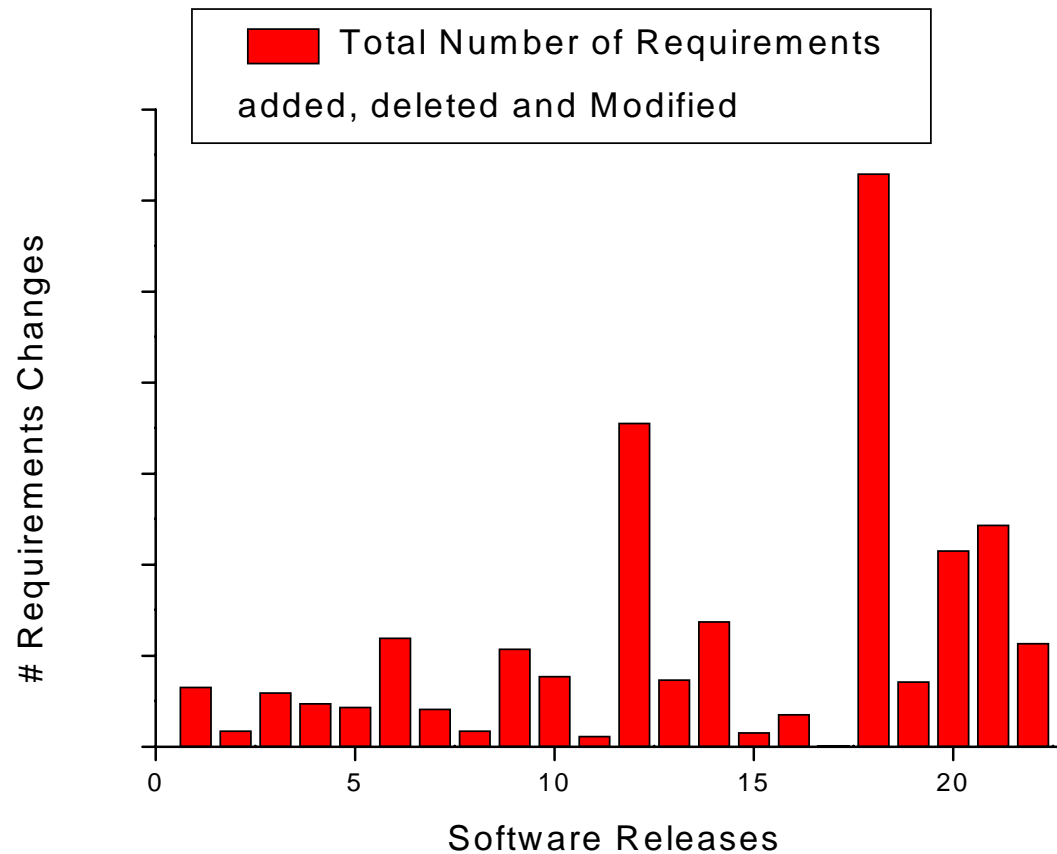
# The Safety-Critical Software Life Cycle



# Development Activities and Deliverables



# Requirements Evolution



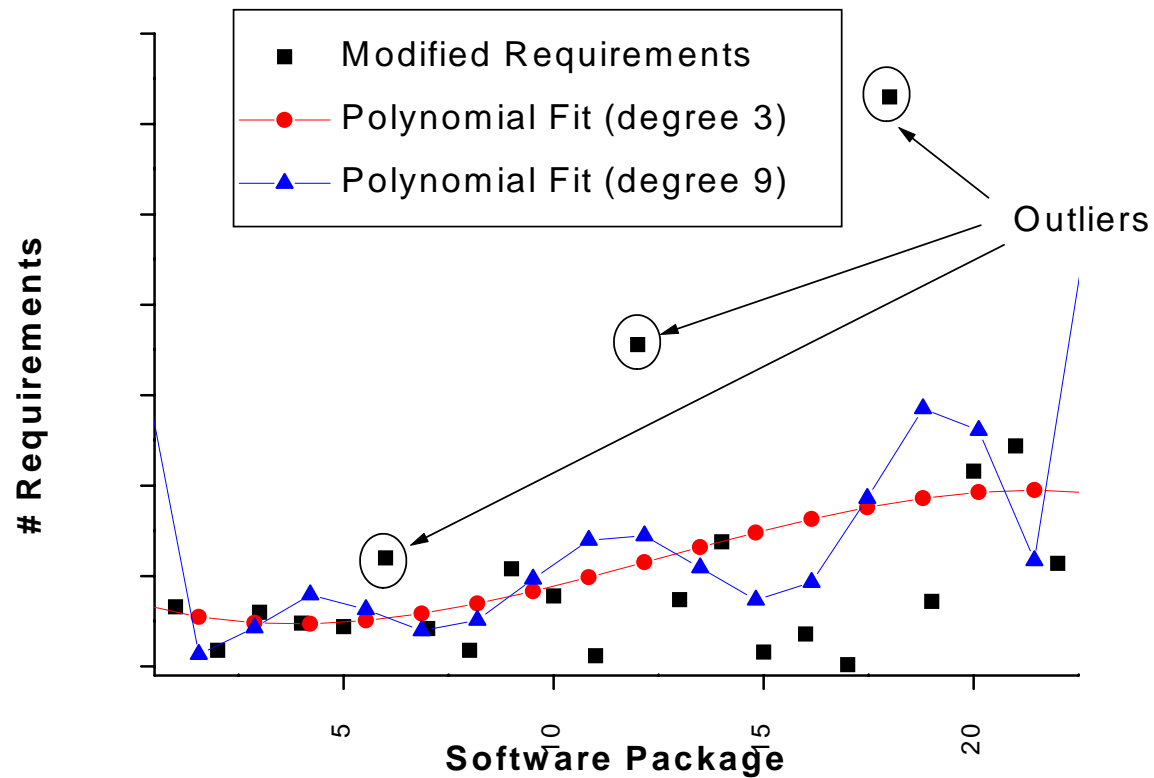
## Type of Changes

Type of Change	Change	Description
General	add, delete, modification of requirements	Requirements are modified due to the specification process maturity and knowledge.
	explanation	The paragraphs that refer to a specific requirement are changed for clarity.
	rewording	The requirements itself does not change, but it could be rewrote for clarity.
	traceability	The traces to other deliverables are changed.
Domain Specific	non-compliance	A requirement that is not applicable for a new software package. This is the case when the requirements specification is based on that one of a previous project.
	partial compliance	A requirement that is applicable partially for a new software package. This is the case when the requirements specification is based on that one of a previous project.
Product Line	hardware modification	Several changes are due to hardware modifications. This type of change applies usually to hardware dependent software requirements.
	range modification	The range of the variables within the scope of a specific requirements is modified.
	add, delete, rename parameters/variables	The variables/parameters to which a specific requirement refers can change.

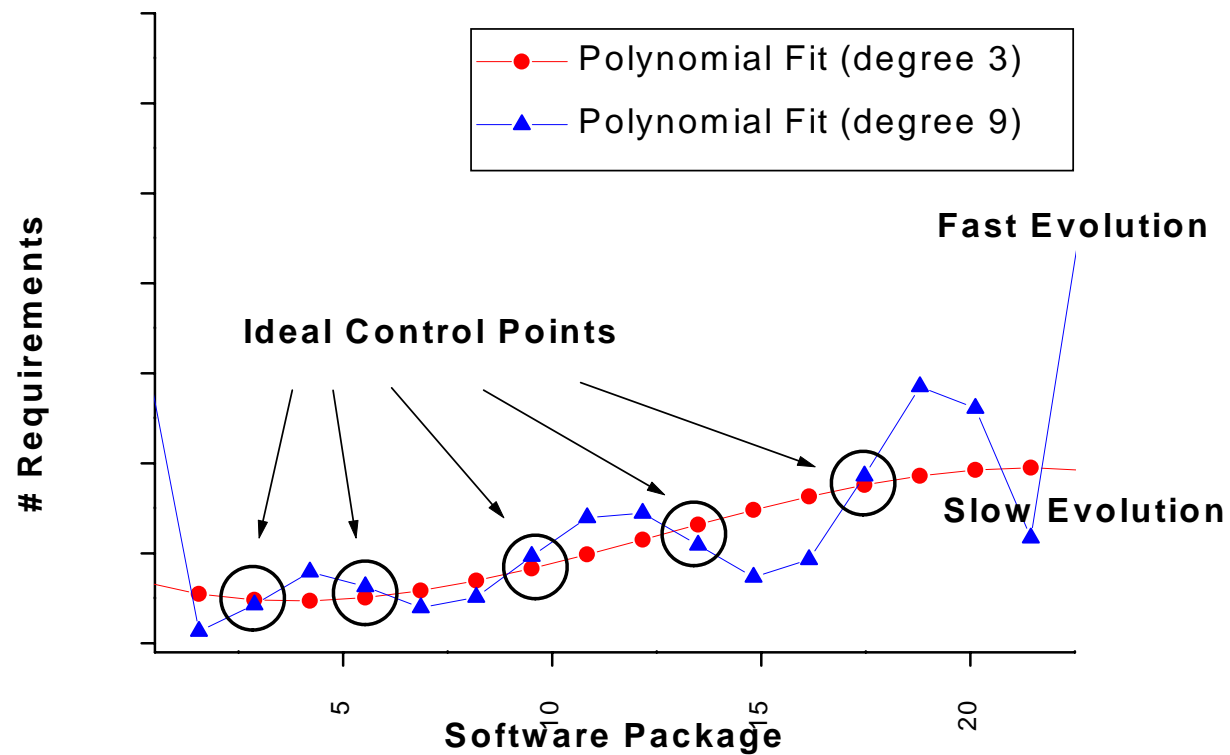
# Requirements Attributes

Attribute	Description
Parameter Variable	Requirements refer to specific parameters or variable on which the statement is based on.
Function Task	Requirements define functions or tasks, which have to be performed by the system, that are stated into specification.
Statement	Within the requirements specification there are statements to clarify the meaning of specific requirements.
Track	Each Requirement has tracks to the other project deliverables. These tracks identify parent origins within documents back in the design process and descendents within further documents (e.g., design specification and software code).
Dependence	Requirements can depend on hardware components, software components and other requirements them self. There could also be temporal (in terms of project stages) dependencies.

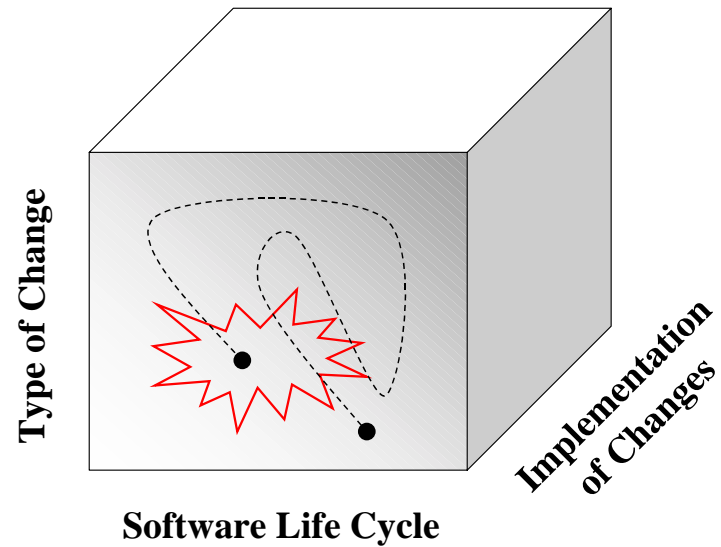
# Requirements Evolution



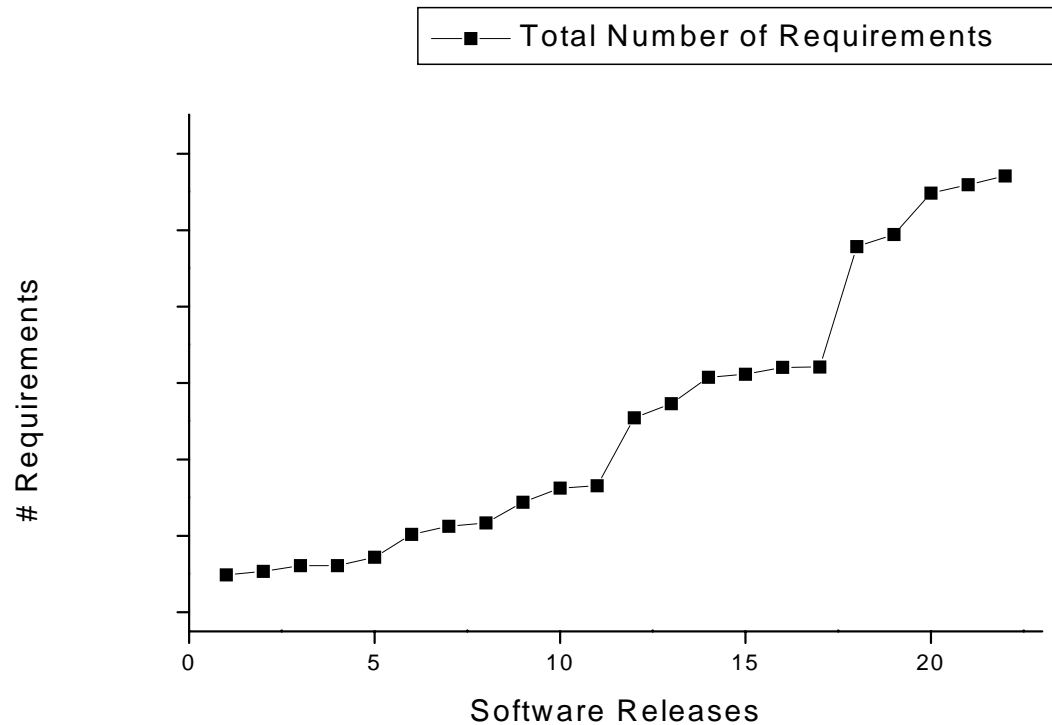
# Requirements Evolution



# A Requirements Evolution Framework



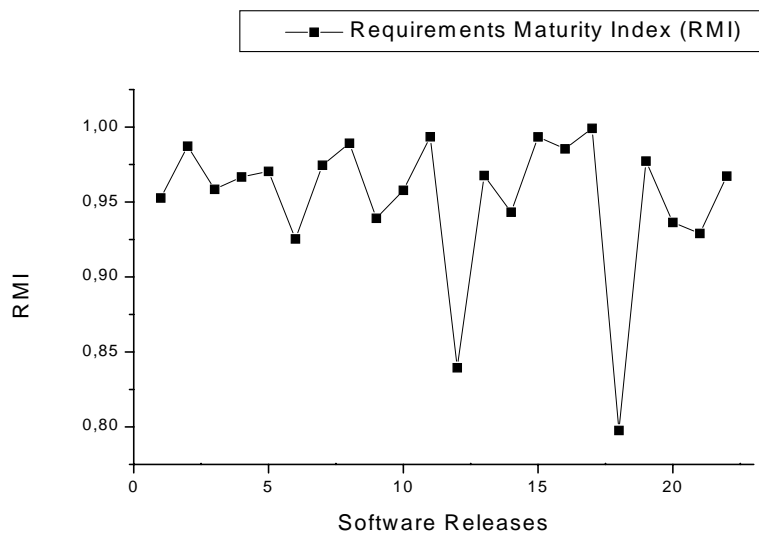
# Size of Requirements



## Remark 1 (Size of Requirements)

*The number of requirements tends to grow over the software releases.*

# Requirements Maturity Index



## Requirements Maturity Index

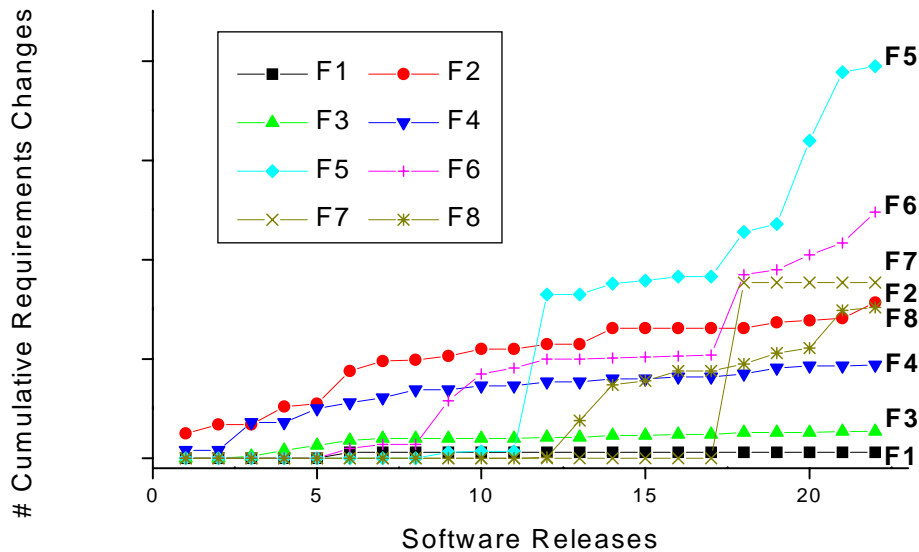
$$RMI^1 = \frac{R_T - R_C}{R_T}$$

<sup>1</sup> $R_T$  = number of software requirements in the current delivery ;  $R_C$  = number of software requirements in the current delivery that are added, deleted or modified from a previous delivery.  
References: IEEE Std 982.1 - 1988, IEEE Std 982.2 - 1988

## Remark 2 (Requirements Maturity Index)

*The Requirements Maturity Index can be misleading in assessing the readiness of requirements. The metrics user should always assess its applicability in the specific context.*

# Requirements Evolution - Functions View



- F1 is not likely to change. F1 defines the requirements for the system architecture.
- Functions that are likely to change during early software releases change less during later releases, and vice versa.

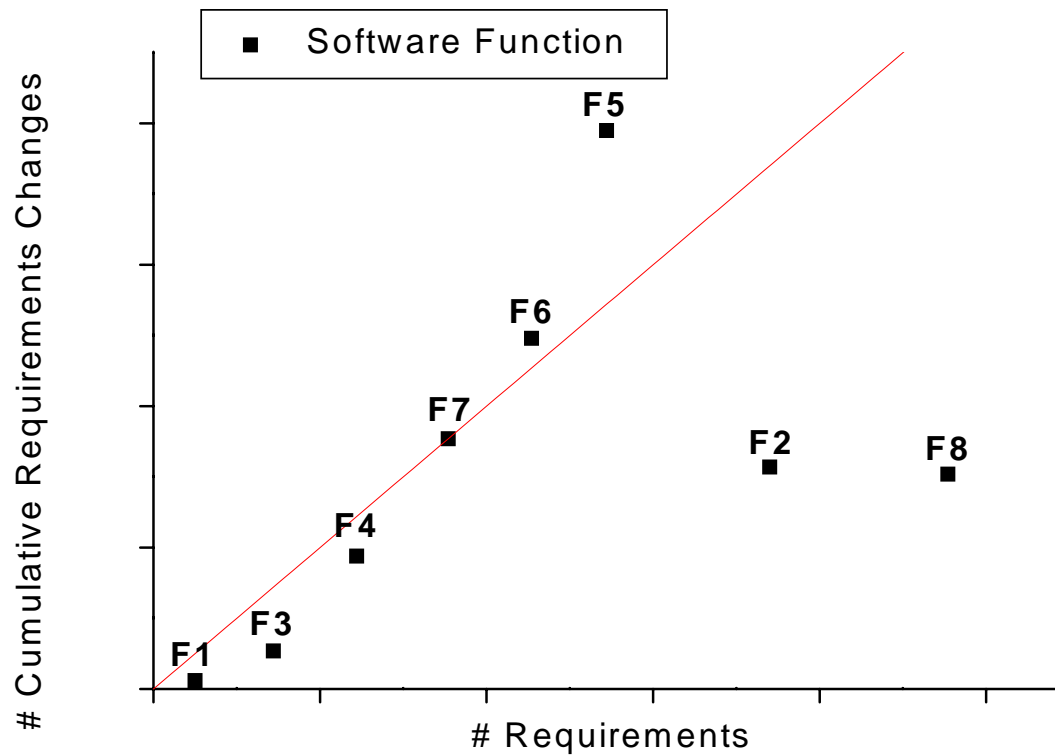
## Remark 3 (Architecture)

*The software requirements of the system architecture represent a stable part of the requirements. The likelihood of architecture changes is low and changes usually occur in early releases of the software, when they can still be accommodated with affordable costs.*

## Remark 4 (Requirements Dependencies)

*Functions that are likely to change during early software releases change less during later releases, and vice versa. The evolution trends show some dependencies between requirements.*

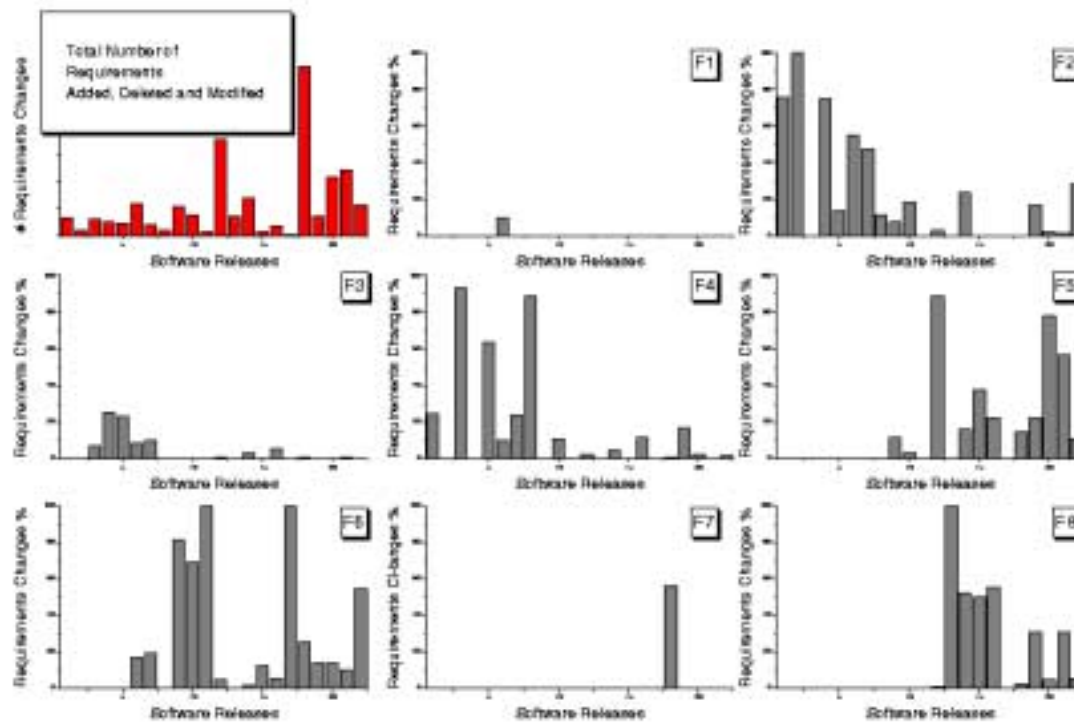
# Requirements Evolution - Functions View



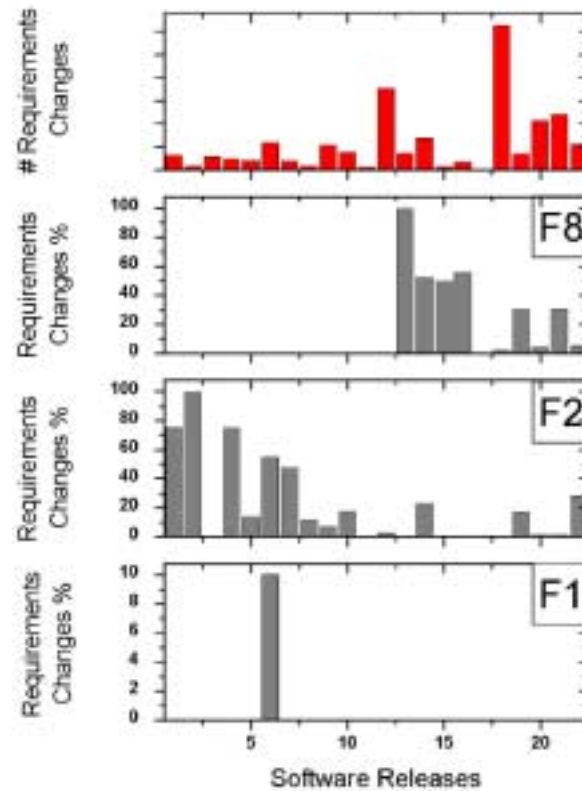
## Remark 5 (Product Oriented Perspective)

*Some functions are more likely to experience requirements change than others. For most of the functions there exists a linear relation between cumulative number of requirements changes and final number of requirements. The management of requirements changes could be improved by a product oriented perspective.*

# Functional Requirements Evolution



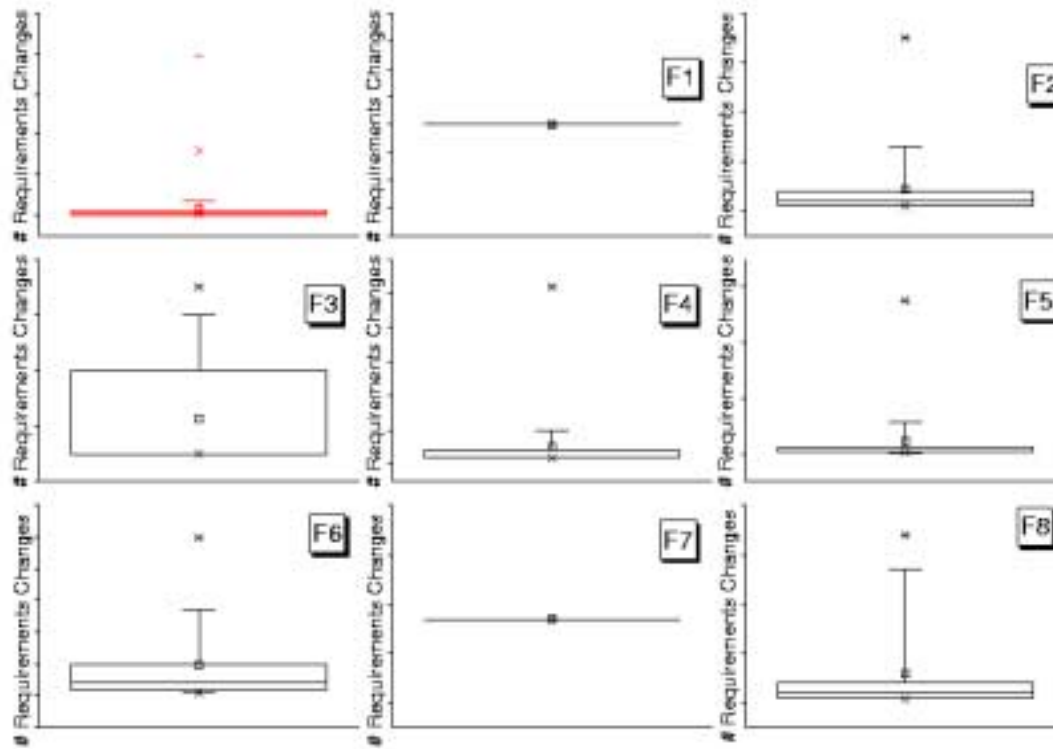
# Functional Requirements Evolution



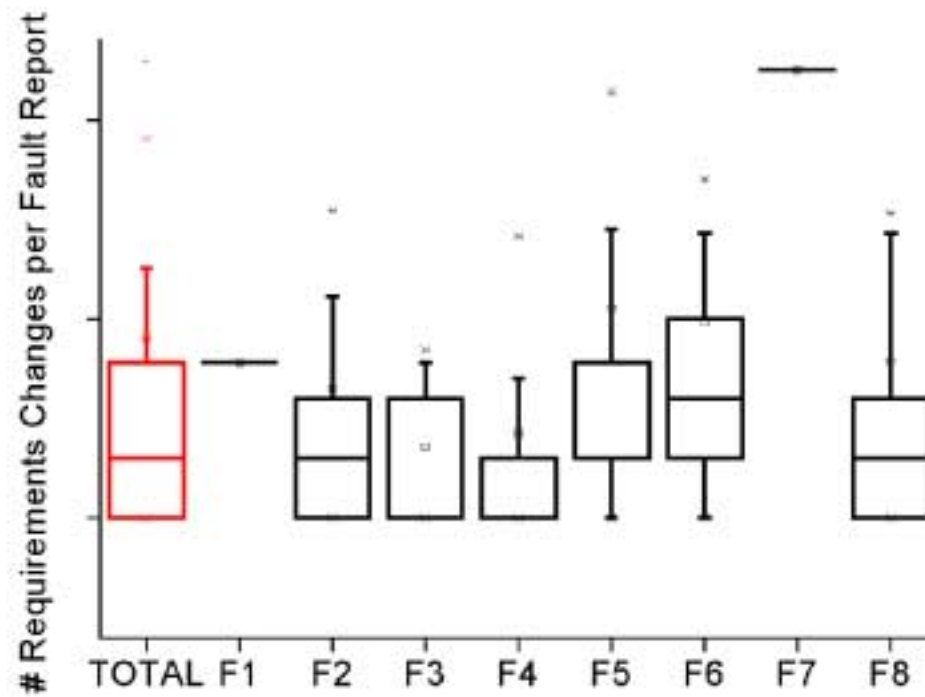
# A Classification of Requirements Evolution

Property	Description	Function
<b>single changes</b>	Requirements changes are concentrated over a single software release	F1, F7
<b>late changes</b>	Requirements changes are likely to occur over late software releases	F5, F7, F8
<b>early changes</b>	Requirements changes are likely to occur over early software releases	F1, F2, F3, F4
<b>spread changes</b>	Requirements changes are likely to occur over any software releases	F2, F4, F6

# Fault Report - Requirements Changes



# Fault Report - Requirements Changes



## Fault Report - Requirements Changes

	<b>single function</b>	<b>multiple functions</b>
<b>single release</b>	A fault report that requires changes into a single function and all changes are allocated to a single software release.	A fault report that requires changes into different functions and all changes are allocated to a single software release.
<b>multiple releases</b>	A fault report that requires changes into a single function and all changes are allocated to subsequent software releases.	A fault report that requires changes into different functions and all changes are allocated to subsequent software releases.

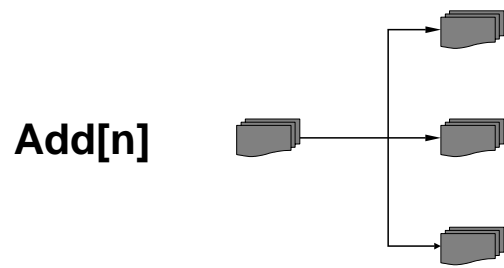
# Requirements Dependencies Matrix

<b>F1</b>	<b>F1</b>							
<b>F2</b>	2	<b>F2</b>						
<b>F3</b>		3	<b>F3</b>					
<b>F4</b>	3	1	1	<b>F4</b>				
<b>F5</b>	1	4	2	6	<b>F5</b>			
<b>F6</b>				1	1	<b>F6</b>		
<b>F7</b>				1	1	1	<b>F7</b>	
<b>F8</b>	1	4	3	5	9	2		<b>F8</b>

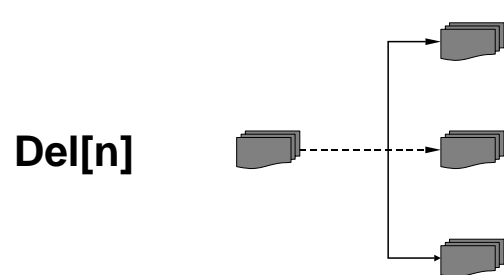
# Modelling Requirements Evolution

- Graphical Representation of Requirements Evolution
  - ★ easy to understand
  - ★ easy to identify similarities and trends
- Capturing evolution throughout the life cycle
- Identification of patterns
  - ★ trade-off among business, process and product viewpoints

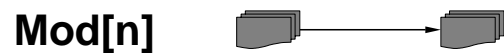
# Requirements Changes



**Add:**  $Add[n]$  introduces a subset of  $n$  new contiguous requirements ordered according to their index into the current requirements document.

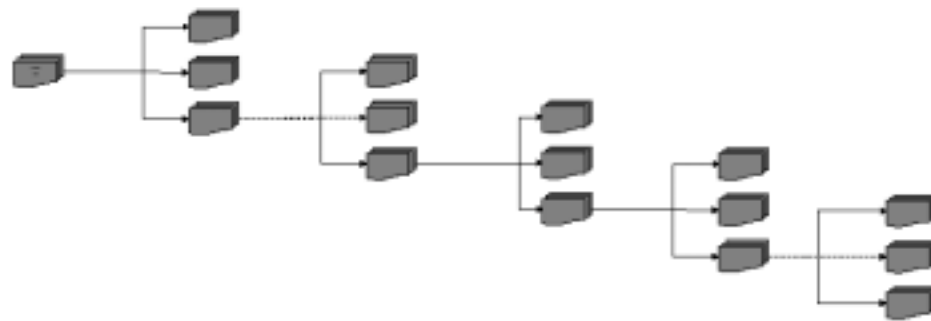


**Delete:**  $Del[n]$  deletes a subset of  $n$  contiguous requirements from the current requirements document.

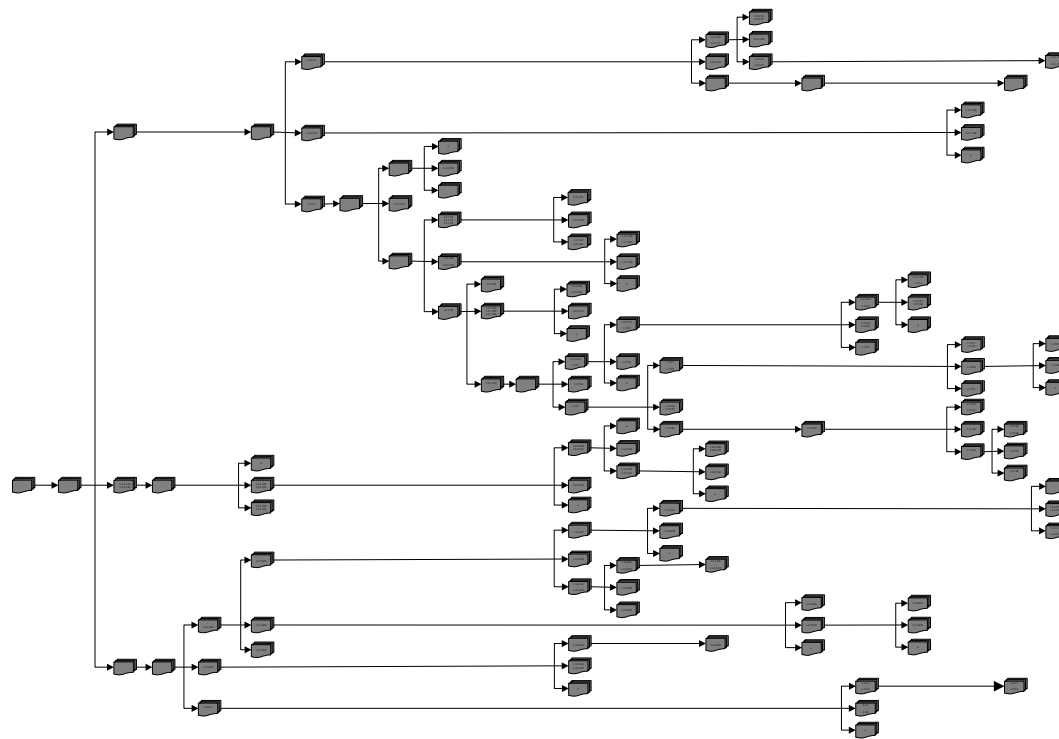


**Modify:**  $Mod[n]$  modifies a subset of  $n$  contiguous requirements into the current requirements document.

# Evolution 1 - F1



# Evolution 2 - F4



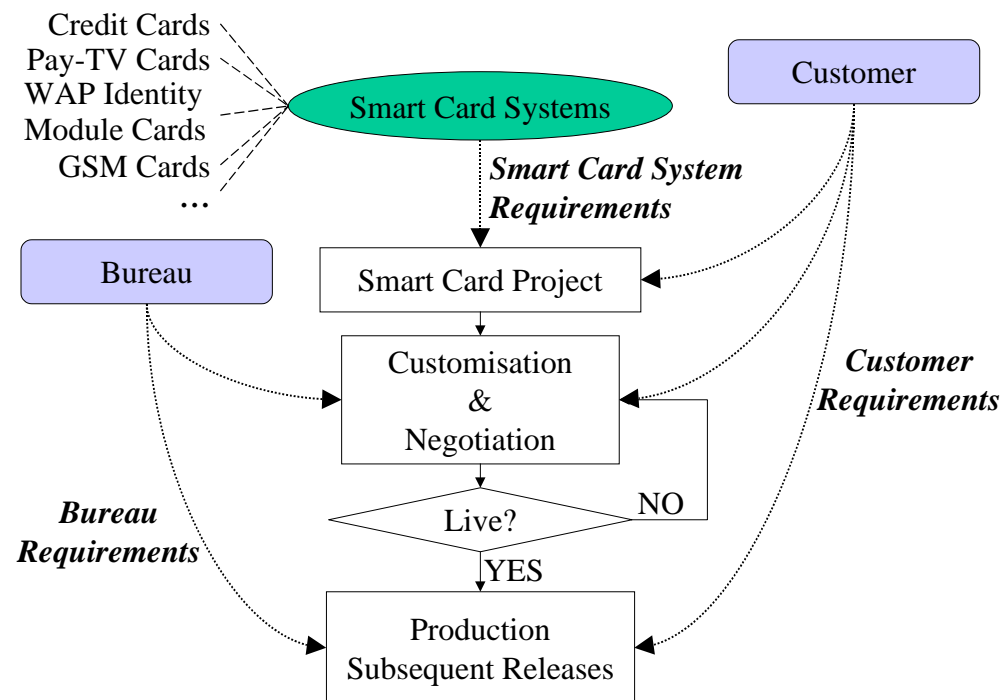
## Formal Requirements Evolution

- Formal specification of the graphical model for Requirements Evolution
- Formal reasoning on Requirements Evolution
- Tool to support the methodology

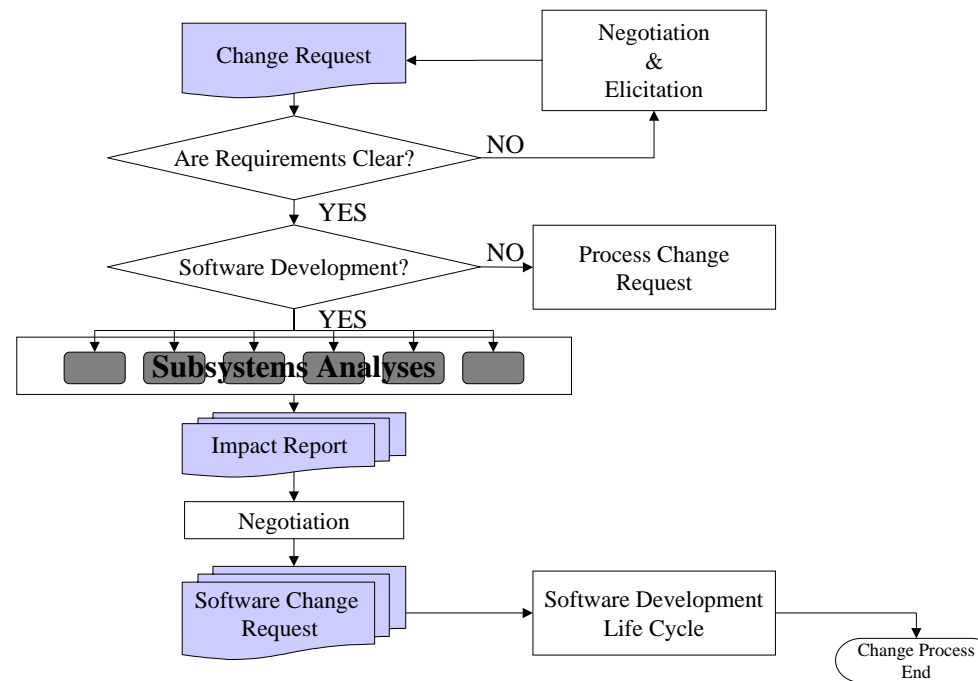
## A Smart Card Case Study

- **Security** context
- **Knowledge distribution** and **understanding** within the organisation
  - **Interviews** of people with different **responsibilities**
  - Requirements Engineering **Questionnaire**
  - **Viewpoints** analysis: **Business**, **Process** and **Product**
  - Requirements Evolution **Viewpoints Management**
- ★ **Architecture stability**
- ★ **Architecture** implements **security** requirements
- ★ **Issues**: project **visibility**, **measuring requirements evolution** and **software functions allocation**

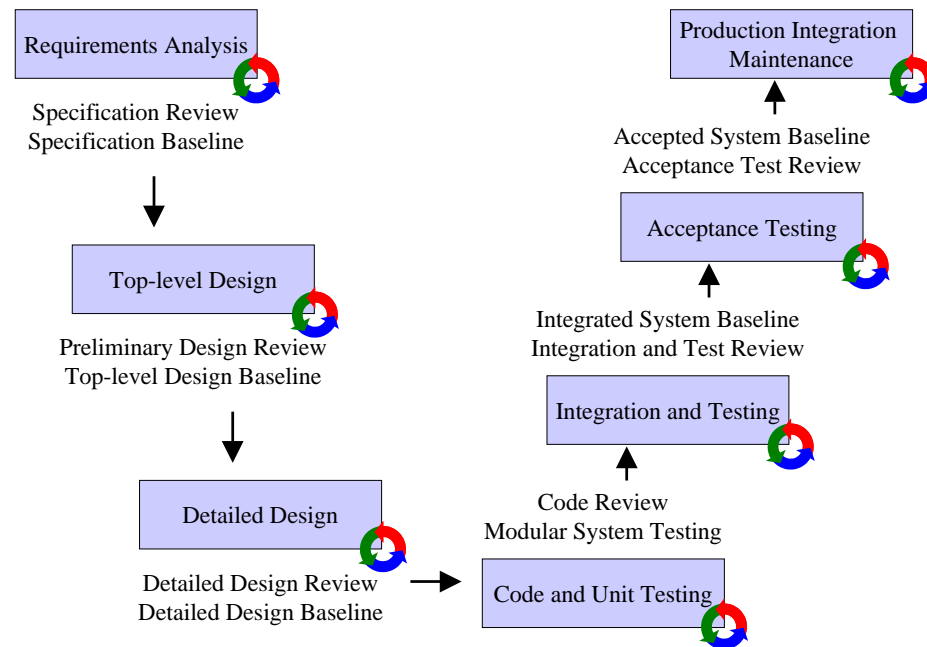
# Business Viewpoint



# Process Viewpoint



# Product Viewpoint



## Lessons Learned

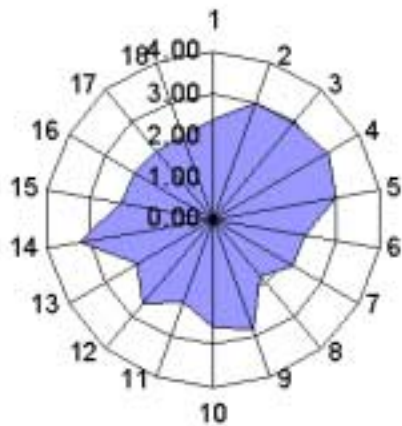
- Requirements viewpoints should capture the hierarchical organisation and its granularity.
- Viewpoints need to be supported by different abilities (e.g., visibility, measurability and functionality).
- Engineering methodologies need to be calibrated in order to set a trade-off supporting good practice management and design.
- Function allocation is important to support reuse, compositionality and cross-projects orthogonal classification.
- The architecture represents a stable part of a system. It implements those requirements that find origin in the technical core of the business.

# Requirements Engineering Questionnaire

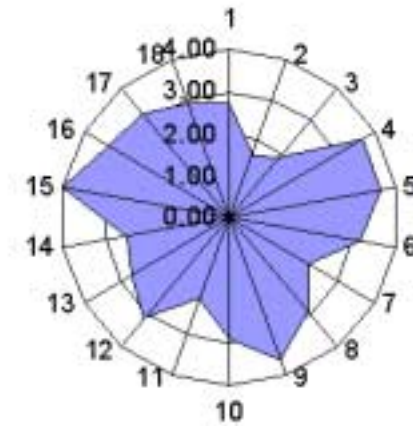
1. Requirements Management Compliance
2. Business Tolerance Requirements
3. Business Performance Requirements
4. Requirements Elicitation
5. Requirements Analysis Negotiation
6. Requirements Validation
7. Requirements Management
8. Requirements Evolution & Maintenance
9. Requirements Process Deliverables
10. Requirements Description
11. System Modelling
12. Functional Requirements
13. Non Functional Requirements
14. Portability Requirements
15. System Interface
16. Requirements Viewpoints
17. Product-Line Requirements
18. Failure Impact Requirements

# Requirements Viewpoints

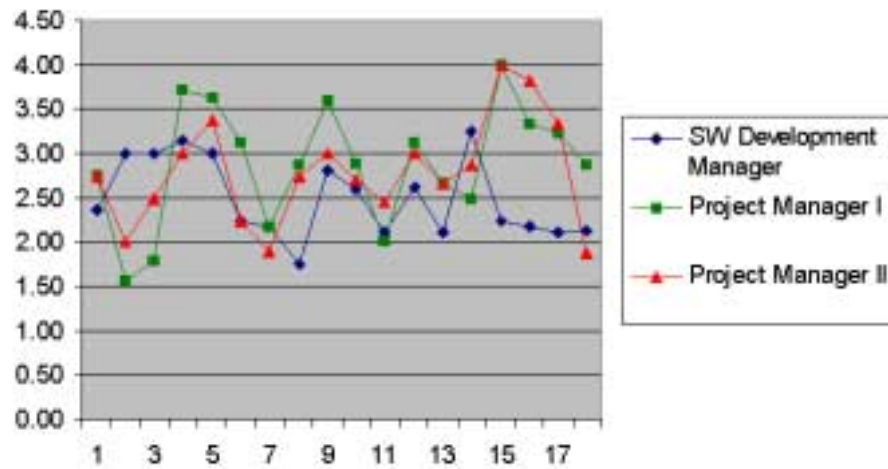
SW Development Manager



Project Manager I



# Requirements Viewpoints



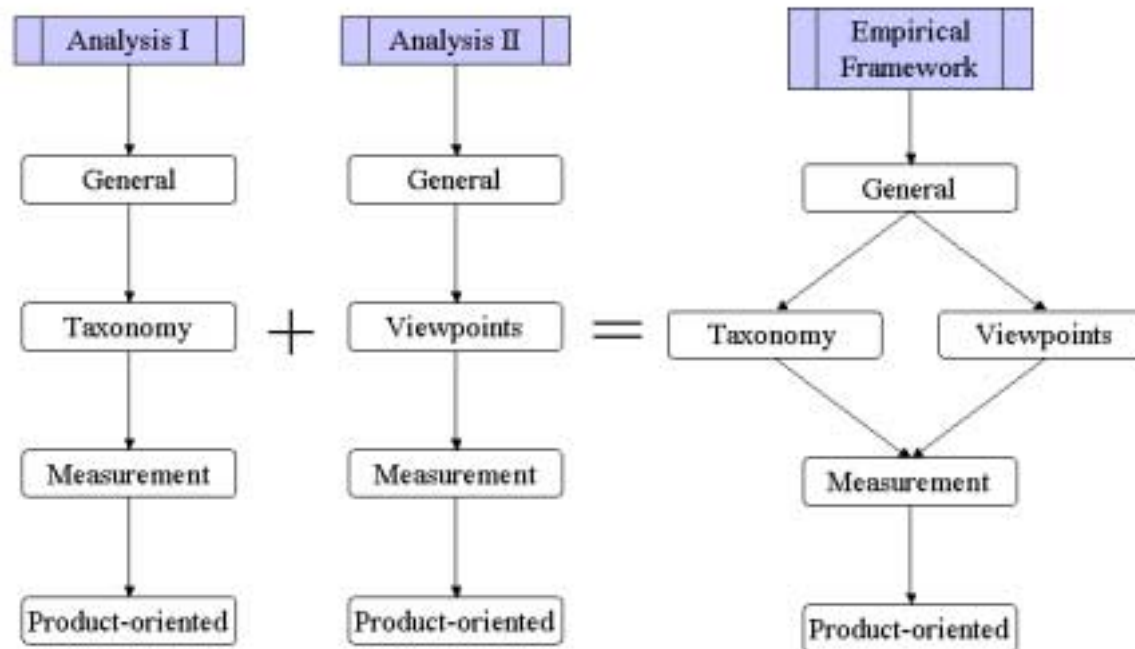
## Discussion and Remarks [1/2]

- **Process** to **product** engineering
  - ★ **Requirements Evolution**: tradeoff among **Business**, **Process** and **Product** viewpoints
  - ★ Classification of industrial contexts
  - ★ The extent to which organisations have flexible business depends on their ability to change their own **B-P-P** tradeoff
- **Measuring** Requirements Evolution
  - ★ Gathering requirements evolution
  - ★ Orthogonal Analysis
  - ★ Functional requirements dependency

## Discussion and Remarks [2/2]

- **Architecture** stability
  - ★ Critical requirements should be associated to stable requirements
  - ★ Stable requirements find origin in the business core
- **Requirements Evolution** representation
  - ★ Structured requirements viewpoints
  - ★ Requirements granularity
  - ★ Extending traceability information

# Empirical Framework



## Conclusions and Further Work

- Requirements Evolution: lecture key of industrial contexts
- Product features may enhance our ability in managing and specifying requirements
- Product-line Requirements Evolution
- Measuring Requirements Evolution
- Modelling Requirements Evolution
- Formal Reasoning on Requirements Evolution
- Dependable Requirements Evolution

# Evolutionary Deliverables

- Quantitative and Qualitative Empirical Framework for Requirements Evolution
- Formal Requirements Evolution
  - ★ Reasoning on Requirements Evolution
- Tools Implementation
- Dependable Requirements Evolution