RiSE

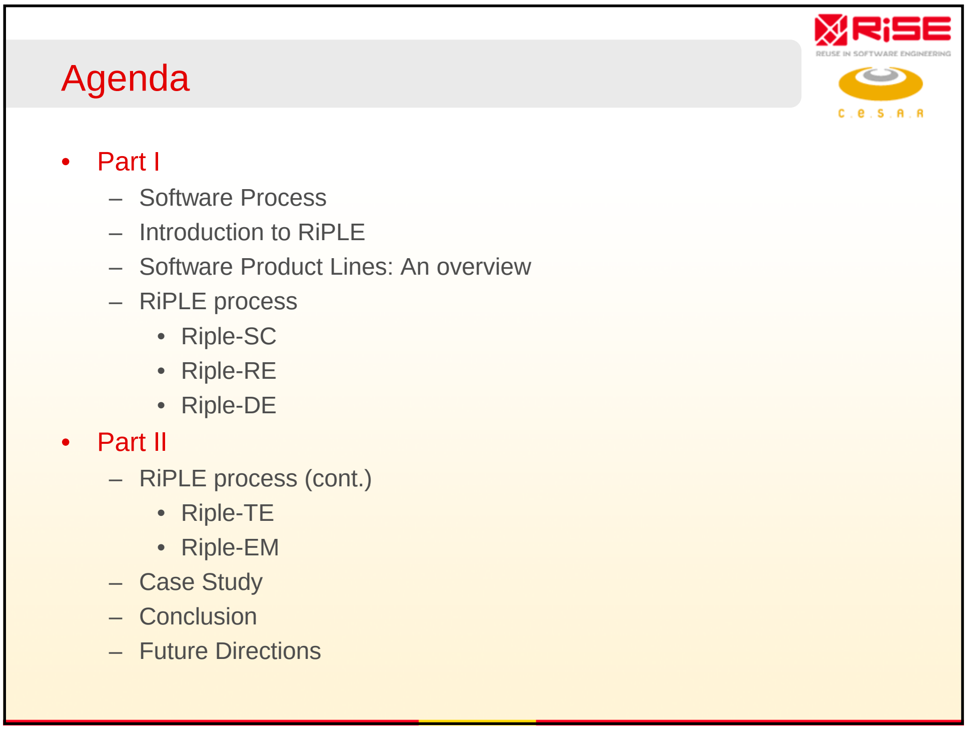
REUSE IN SOFTWARE ENGINEERING

DO MORE

www.rise.com.br

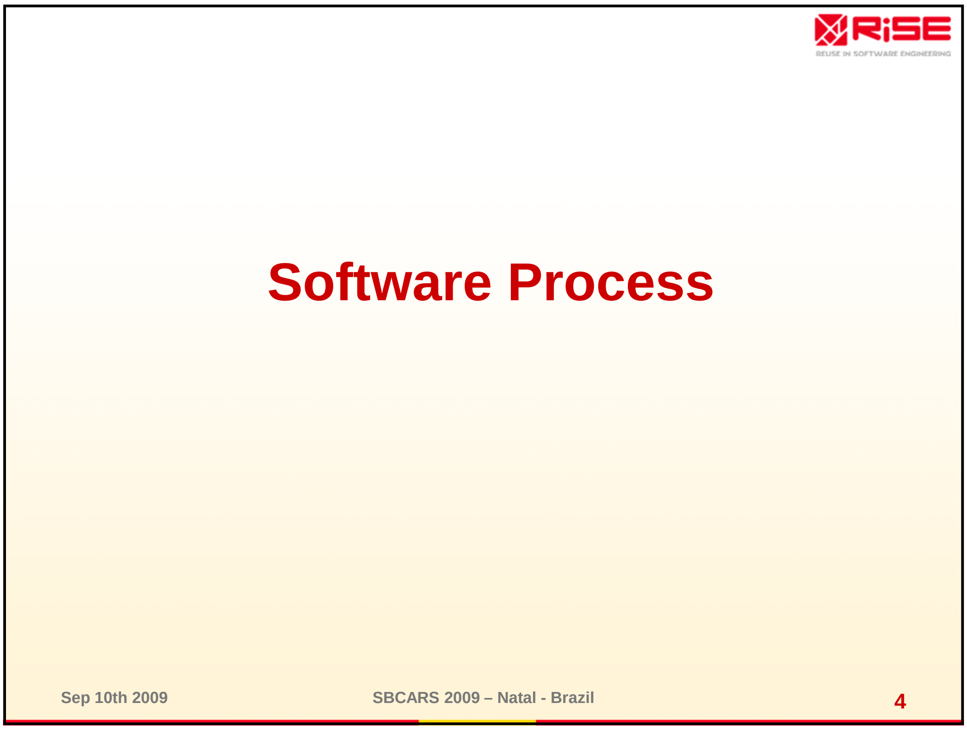
RiPLE: The RiSE Process for Product Line Engineering

Eduardo Almeida, Marcela Balbino, Danuza Neiva, Ednaldo Dilorenzo, Paulo Silveira, Ivan Machado, Thiago Burgos, Vanilson Burégio, Silvio Meira

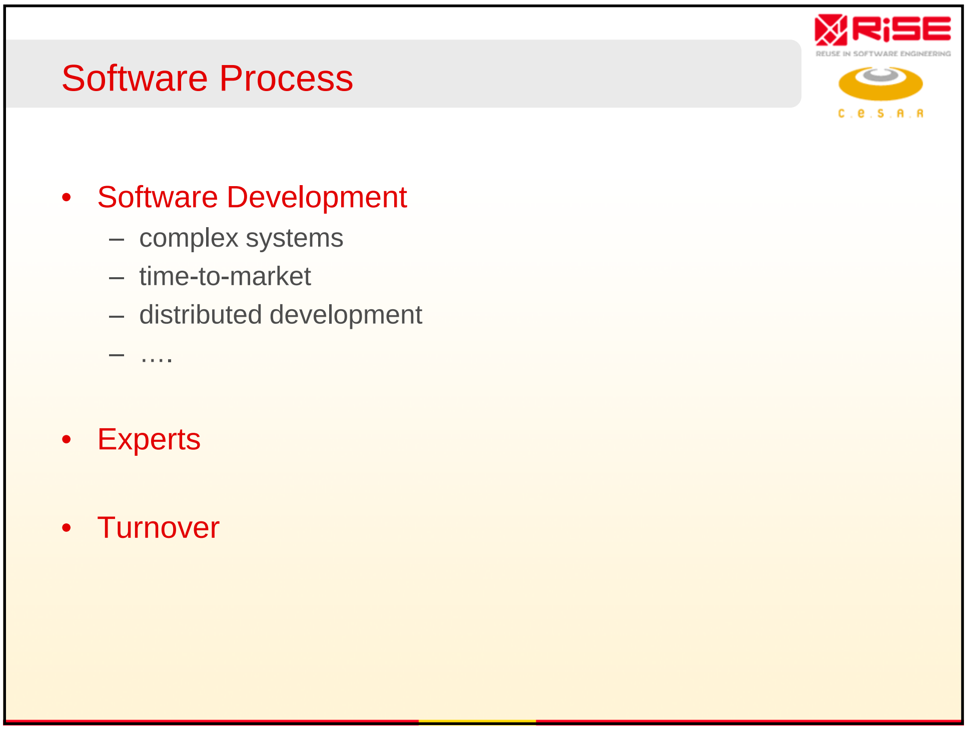


Agenda

- Part I
  - Software Process
  - Introduction to RiPLE
  - Software Product Lines: An overview
  - RiPLE process
    - Ripple-SC
    - Ripple-RE
    - Ripple-DE
- Part II
  - RiPLE process (cont.)
    - Ripple-TE
    - Ripple-EM
  - Case Study
  - Conclusion
  - Future Directions

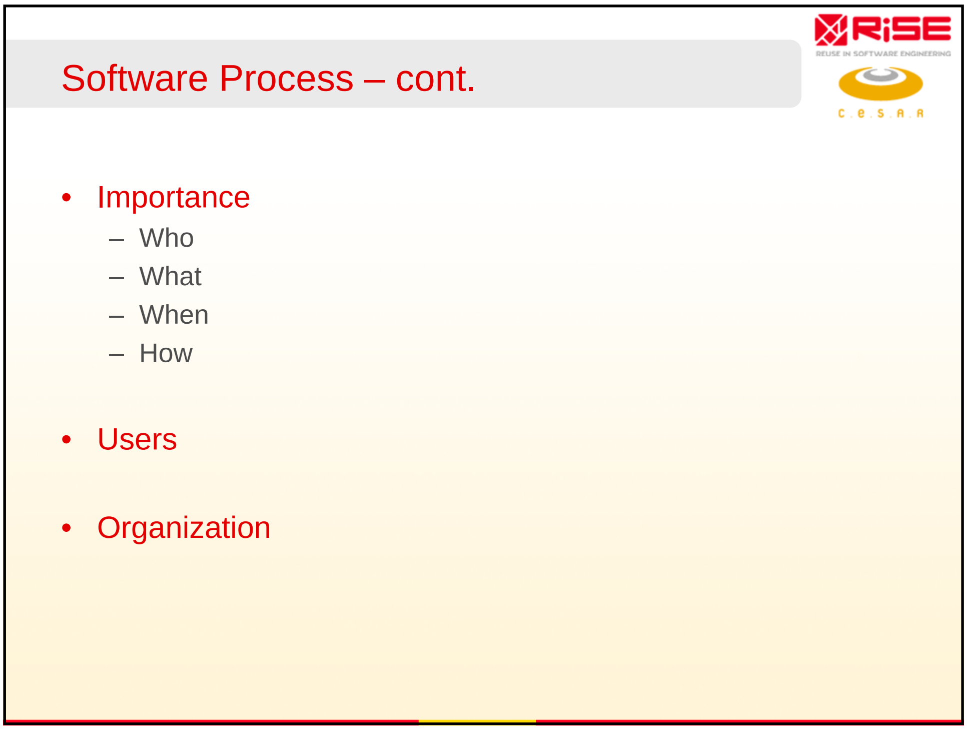


Software Process



Software Process

- Software Development
  - complex systems
  - time-to-market
  - distributed development
  - ….
- Experts
- Turnover



Software Process – cont.

- Importance
  - Who
  - What
  - When
  - How
- Users
- Organization

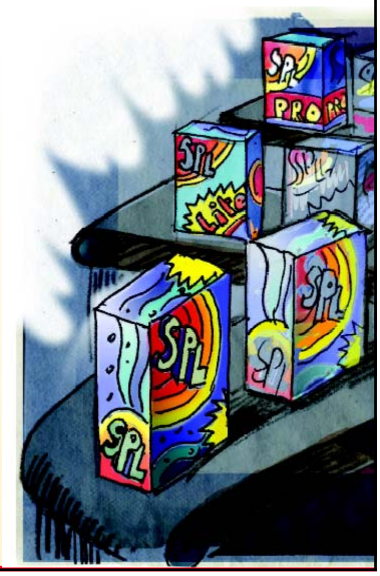
Software Process – cont.

- A software process is a set of activities that leads to the production of a software product
- Influences
  - Project | Methods | Tools
  - Knowledge
  - Cost
  - Contract
  - …..

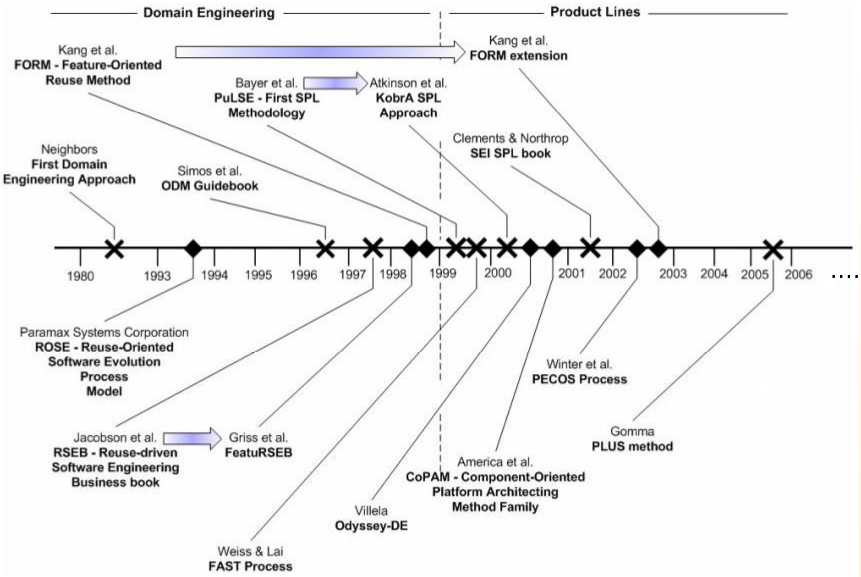
[Sommerville, 2008]

Software Process – cont.

- Process Model
  - Software Reuse
- Why a new process?



The State-of-the-Art [Almeida, 2007]



Domain Engineering

Product Lines

Kang et al.
FORM - Feature-Oriented Reuse Method

Kang et al.
FORM extension

Bayer et al.
PuLSE - First SPL Methodology

Atkinson et al.
KobrA SPL Approach

Clements & Northrop
SEI SPL book

Neighbors
First Domain Engineering Approach

Simos et al.
ODM Guidebook

1980 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 ….

Paramax Systems Corporation
ROSE - Reuse-Oriented Software Evolution Process Model

Winter et al.
PECOS Process

Gomma
PLUS method

Jacobson et al.
RSEB - Reuse-driven Software Engineering Business book

Griss et al.
FeatuRSEB

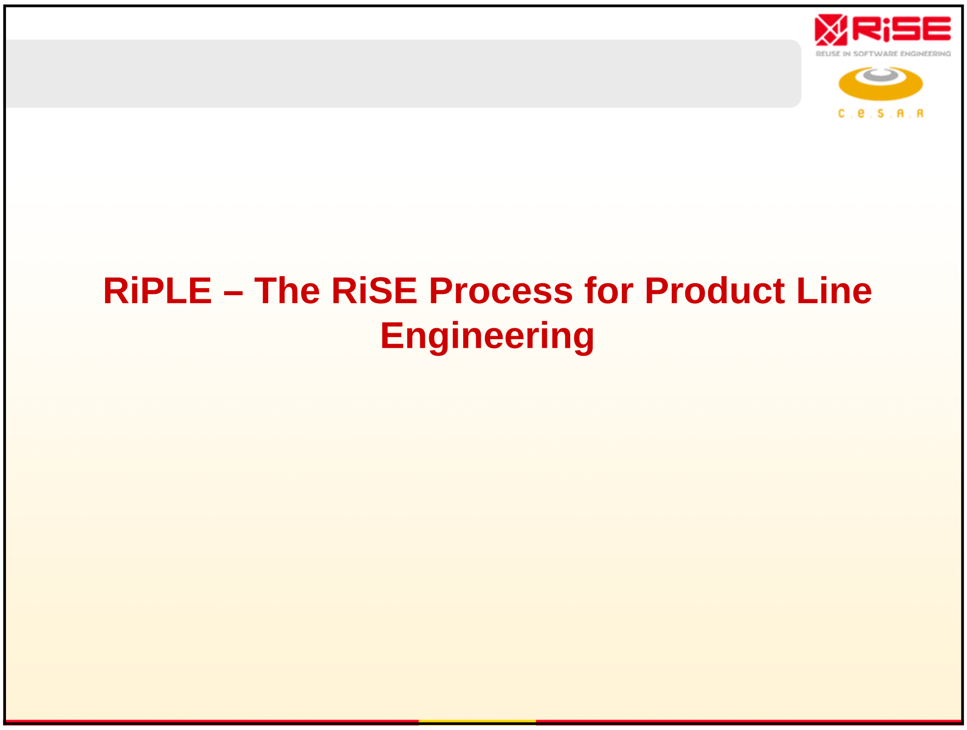
America et al.
CoPAM - Component-Oriented Platform Architecting Method Family

Villela
Odyssey-DE

Weiss & Lai
FAST Process

The State-of-the-Art – cont.

- Systematic Reviews
  - Scoping
    - in evaluation
  - Requirements
    - in evaluation
  - Design
    - European Conference on Software Architecture (ECSA 2008)
  - Testing
    - in evaluation
  - Evolution
    - in evaluation
  - Tools
    - Journal of Information and Software technology 2009



RiPLE – The RiSE Process for Product Line Engineering

Integrated Effort

RiSE
REUSE IN SOFTWARE ENGINEERING

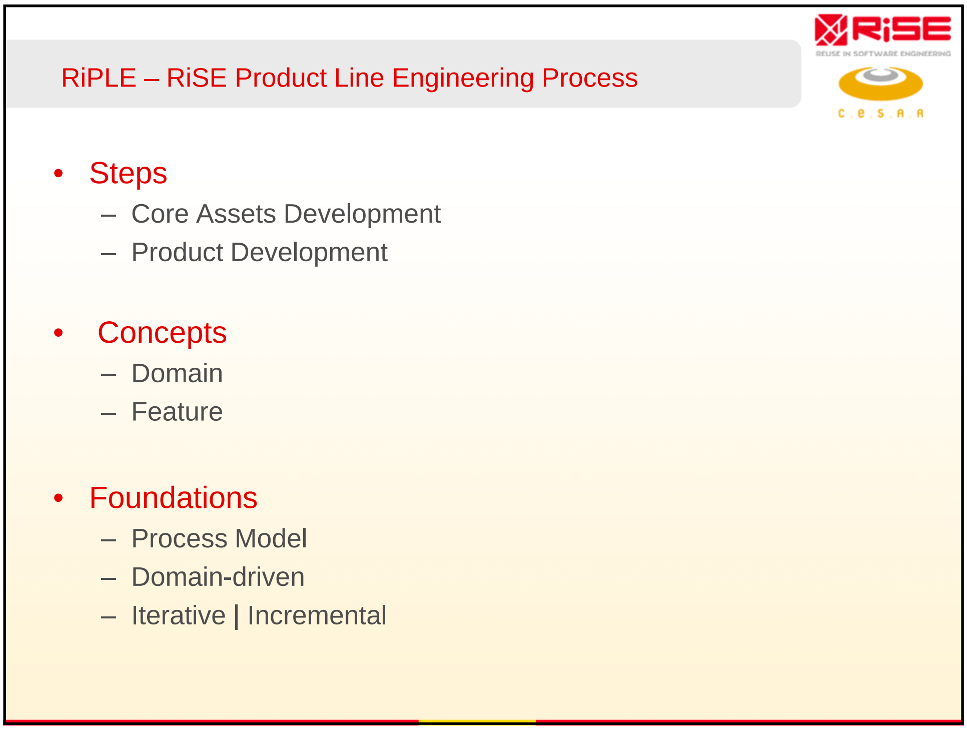
USP SÃO CARLOS

C.E.S.A.R

ines

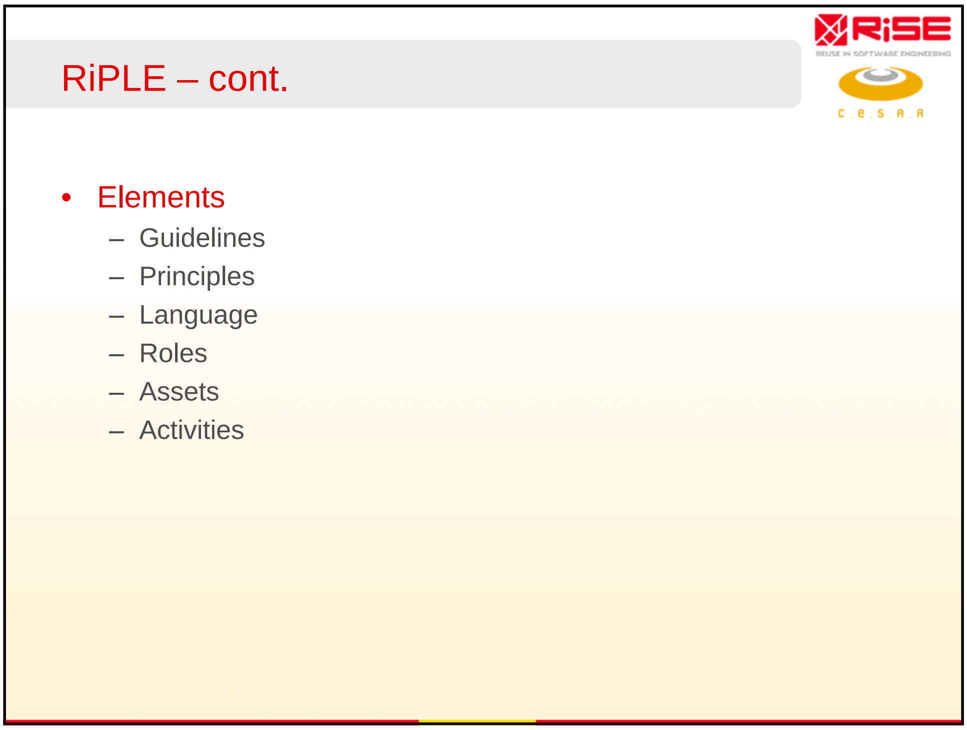
VIRTUTE SPIRITUS
UNIVERSIDADE FEDERAL
DA BAHIA - UFBA

VIRTUS IMPAVIDA
UFPE



RiPLE – RiSE Product Line Engineering Process

- Steps
  - Core Assets Development
  - Product Development
- Concepts
  - Domain
  - Feature
- Foundations
  - Process Model
  - Domain-driven
  - Iterative | Incremental

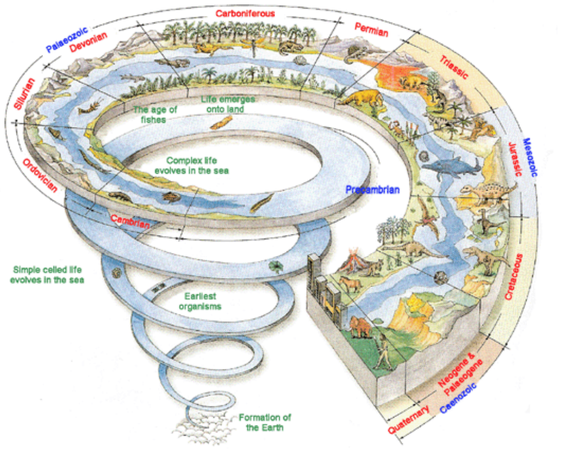


RiPLE – cont.

- Elements
  - Guidelines
  - Principles
  - Language
  - Roles
  - Assets
  - Activities

Software Product Lines (SPL)

A little bit of **history**…



The basis of Product Line Engineering....

- Henry Ford
  - “The father of **assembly-line automation”**

- **Model T** production (1908)
  - Main concept: **Interchangeble parts**
    - based on the ideas of Honoré Blanc and Eli Whitney
  - **Streamlined the production process**

The Economy of scale!

- Line of motor cars
  - affordable, built quickly, high quality

***“Any customer can have a car painted any colour that he wants so long as it is black*”** - Henry Ford

The Economies of scale!

- Line of motor cars
  - affordable, built quickly, high quality

"***Any customer can have a car painted any colour that he wants so long as it is black***" - Henry Ford

Certain choices where extremely limited!

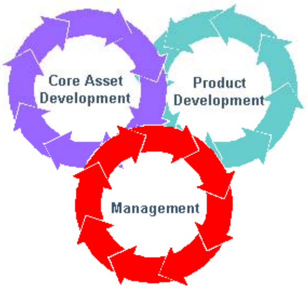
Product Line Engineering (PLE)

- **Economies of Scope**
  - **Mass customization**: producing goods and services to meet **individual customers needs**
  - Create an **underlying architecture** for an organization's **product platform**
  - **Core assets** can be reused to engineer new products from the **basic family**

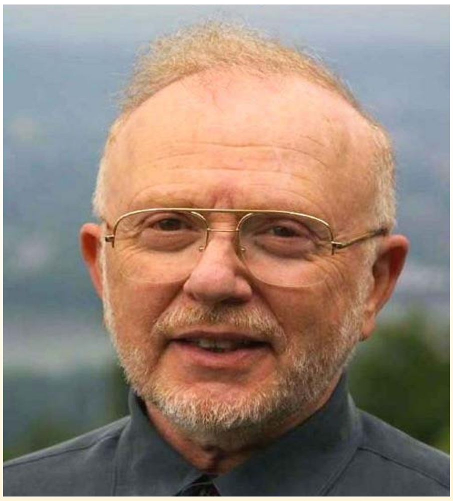
Economies of scale	*Economies of scope*
multiple identical instances *of a* ***single design*** *are produced collectively, rather than individually*	***multiple similar*** *but* ***distinct designs*** *and prototypes are produced collectively, rather than individually*

Software Product Lines (SPL)

- **Based on the ideas of PLE**
  - **Fundamental principle: variability management**
    - Allows to adapt a product to the **customer needs**
    - **Adaptation** is typically performed **during SPL development**



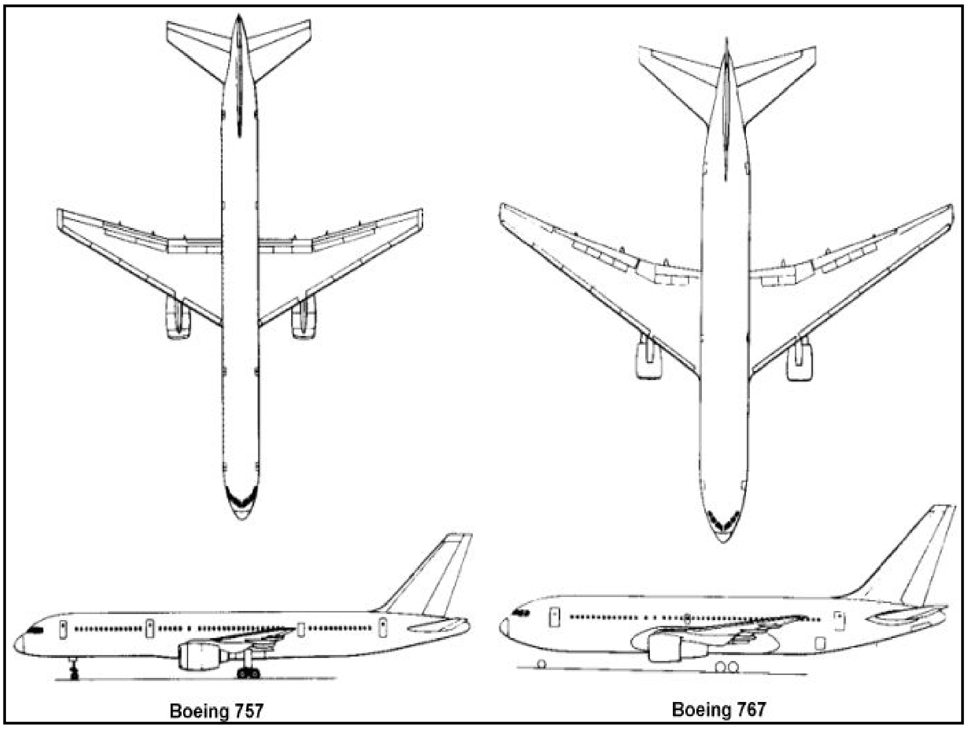
The roots....



David Lorge Parnas

- On the Design and Development of Program Families
  Parnas, D.L.;

  IEEE Transactions on Software Engineering, Vol. 02, Issue 01, March 1976, pp. 1 - 9


Boeing 757
Boeing 767

Aeroflite

BOEING
777
legend lives on . . .

RiSE
REUSE IN SOFTWARE ENGINEERING

Software Product Lines

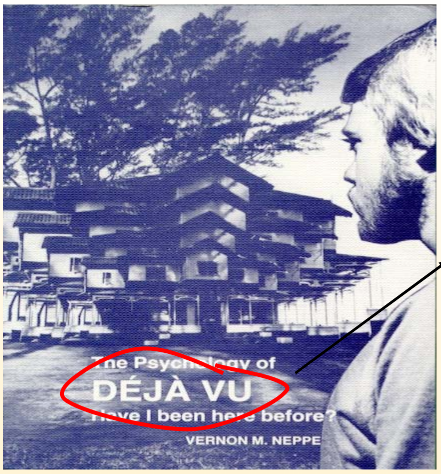
> *"A software product line is a set of software-intensive systems sharing a* ***common***, ***managed*** *set of features that satisfy the* ***specific needs*** *of a* ***particular market segment*** *or mission and that are developed from a* ***common*** *set of core assets in a* ***prescribed way.***
>
> **Paul Clements and Linda Northrop, 2002**

Essential Factors

- Investment
- Planning
- Direction
- Business Strategy

Management

Is Product Lines a new approach?



- Small-Grained Reuse
- Single-System Development with Reuse
- Component-Based Development
- Reconfigurable Architecture
- Release and versions of Single Products

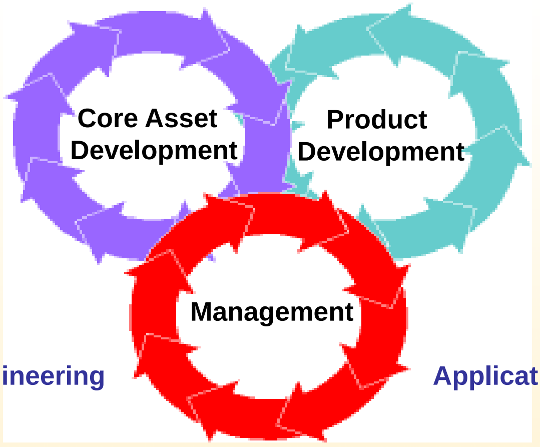
Organizational Benefits

- To achieve **large-scale productivity gains**
- To **improve time-to-market**
- To maintain market presence
- To **improve product quality**
- To increase customer satisfaction
- To **achieve reuse goals**
- To **enable mass customization**

Product Line asset repository Benefits

- Requirements
- Architecture
- Components
- Modeling and Analysis
- Testing
- Planning

Essential Activities



Core Asset Development

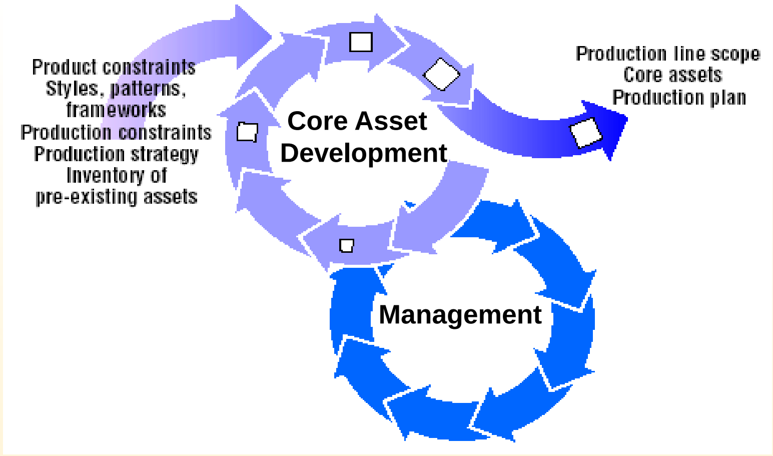
Product Development

Management

Domain Engineering

Application Engineering

Core Asset Development

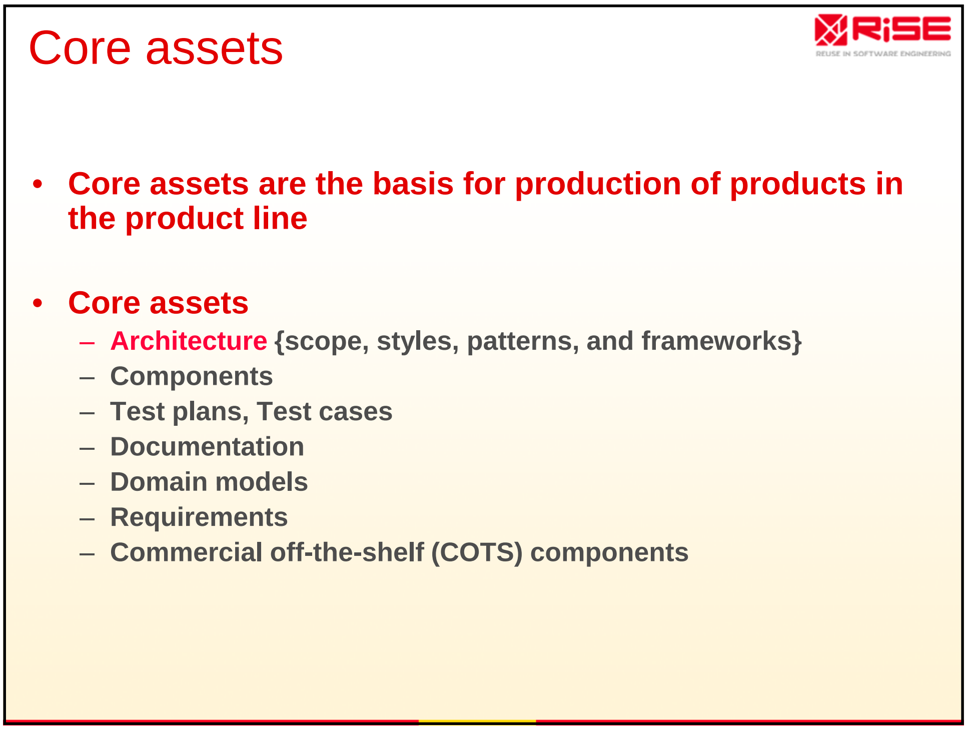


Product constraints
Styles, patterns, frameworks
Production constraints
Production strategy
Inventory of pre-existing assets

Core Asset Development

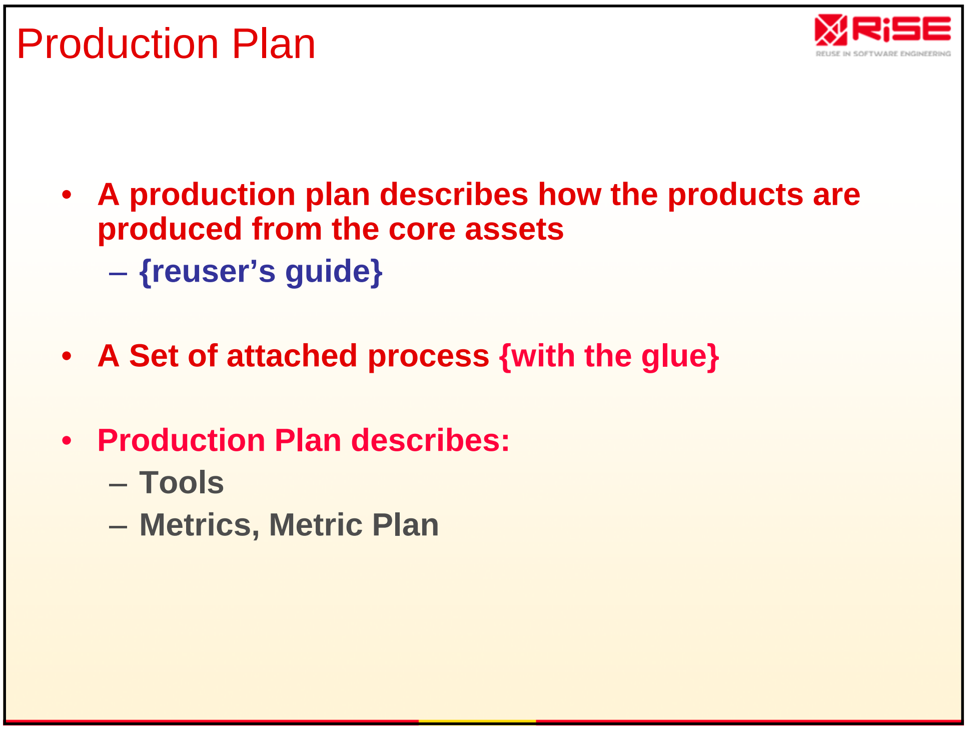
Management

Production line scope
Core assets
Production plan



Core assets

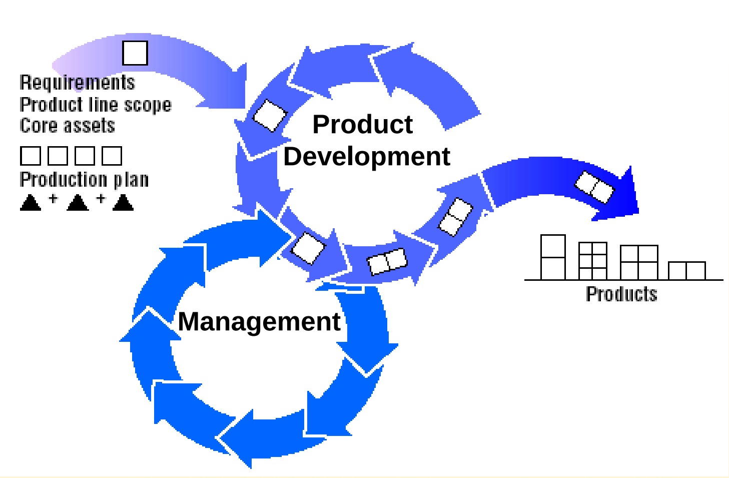
- Core assets are the basis for production of products in the product line
- Core assets
  - Architecture {scope, styles, patterns, and frameworks}
  - Components
  - Test plans, Test cases
  - Documentation
  - Domain models
  - Requirements
  - Commercial off-the-shelf (COTS) components



Production Plan

- A production plan describes how the products are produced from the core assets
  - {reuser's guide}
- A Set of attached process {with the glue}
- Production Plan describes:
  - Tools
  - Metrics, Metric Plan

Product Development



Management

- Critical role in the successful fielding of a product line
- Technical
  - Core asset development
  - Product development
- Organizational
  - Training
  - Funding
  - Risks

Some Successful Cases

Product Line Hall of Fame

http://www.sei.cmu.edu/productlines/plp_hof.html

RiSE
REUSE IN SOFTWARE ENGINEERING
NOKIA
Connecting People

Nokia: Mobile Phones

- Case study
  - Mobile phones
- Previous scenario
  - The initial software architecture for this product line addressed variations in hardware, communication standards, and user interfaces

Nokia: Mobile Phones

- Challenges
  - Language Challenge Abstract
  - The Hardware Challenge
  - The Feature Challenge
- Strategies
  - The current architecture is component based in the client-server style. It allows separate service providers to be plugged in or taken out without restarting the system

Nokia: Mobile Phones

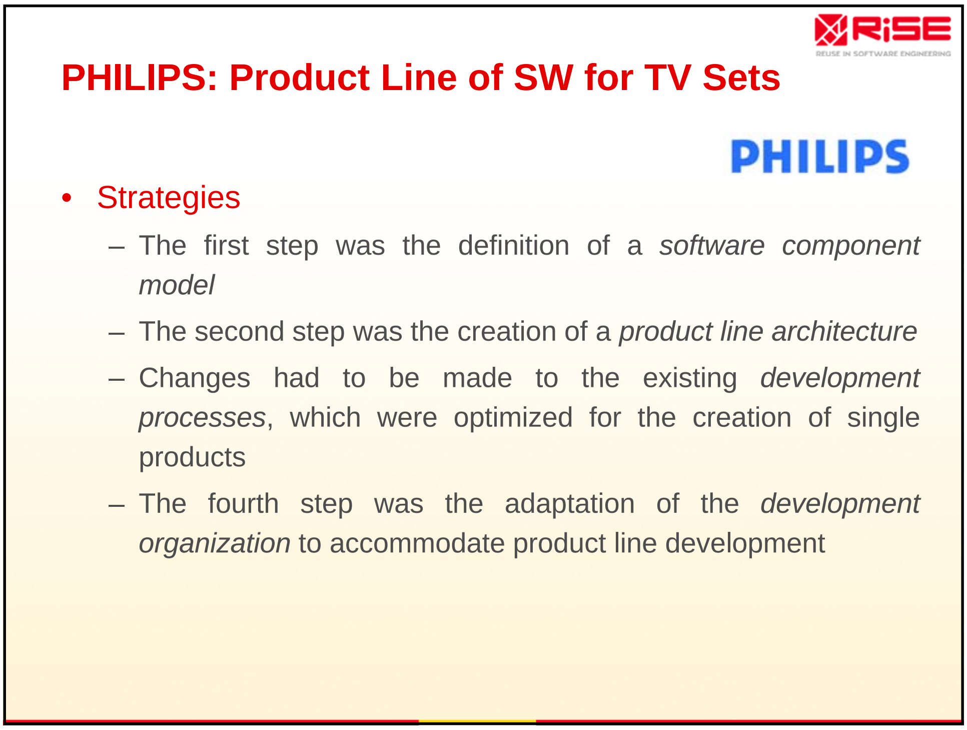
- Current Scenario
  - **32 different phones are manufactured covering six different protocol standards**, a wide variety of functional features and capabilities, different user interface designs, and many platforms and environments
- Results and Metrics
  - Nokia Mobile Phones is the world's largest mobile phone manufacturer, and they believe that software product line engineering has helped it to reach that position

PHILIPS: Product Line of SW for TV Sets

- Case study
  - Televisions sets
- Previous scenario
  - While initially solely consisting of hardware, TVs now contain fully equipped embedded computers to control the hardware and to implement extra features
  - These computers started small, with 1 kilobyte of code around 1980, resulting in many million lines of code today

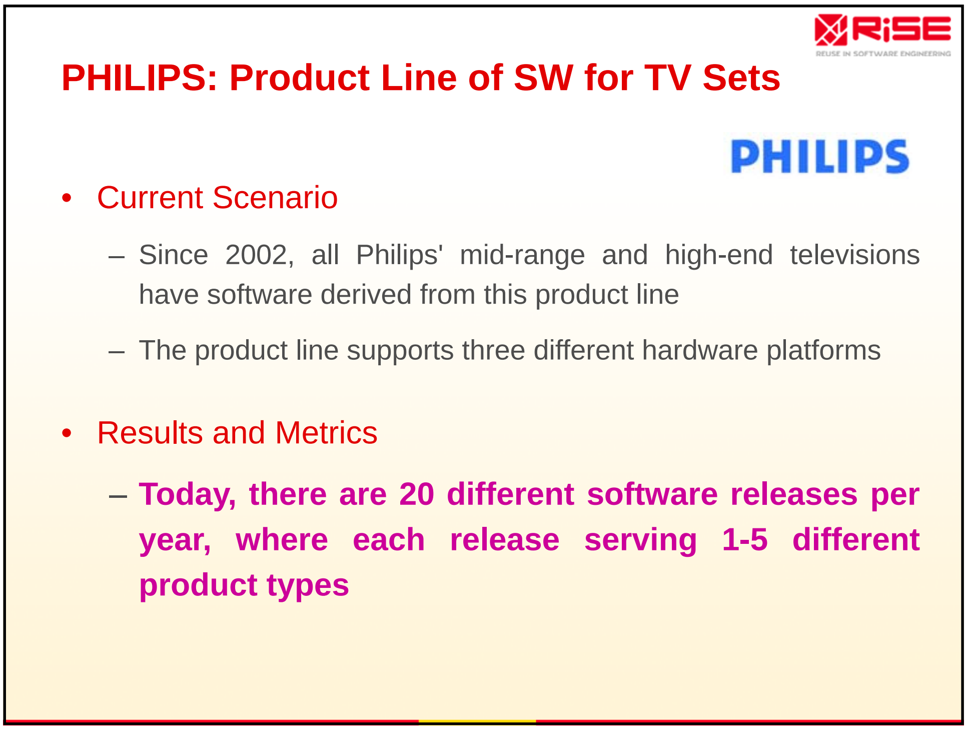
PHILIPS: Product Line of SW for TV Sets

- Challenges
  - Complexity
  - Diversity, since televisions are produced in many different variants



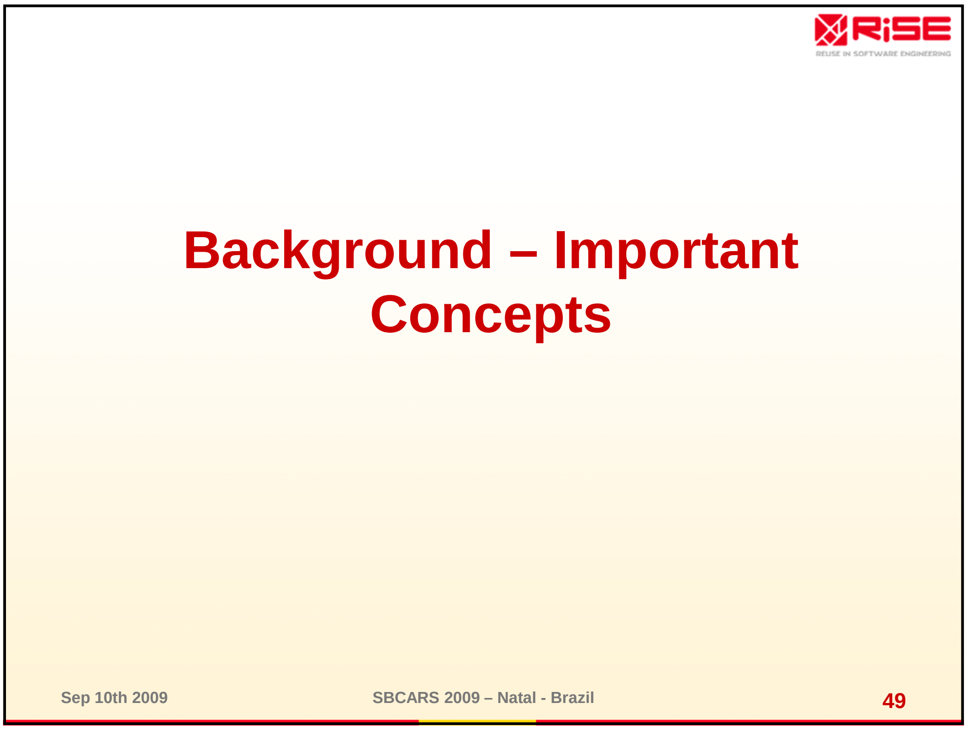
PHILIPS: Product Line of SW for TV Sets

- Strategies
  - The first step was the definition of a *software component model*
  - The second step was the creation of a *product line architecture*
  - Changes had to be made to the existing *development processes*, which were optimized for the creation of single products
  - The fourth step was the adaptation of the *development organization* to accommodate product line development

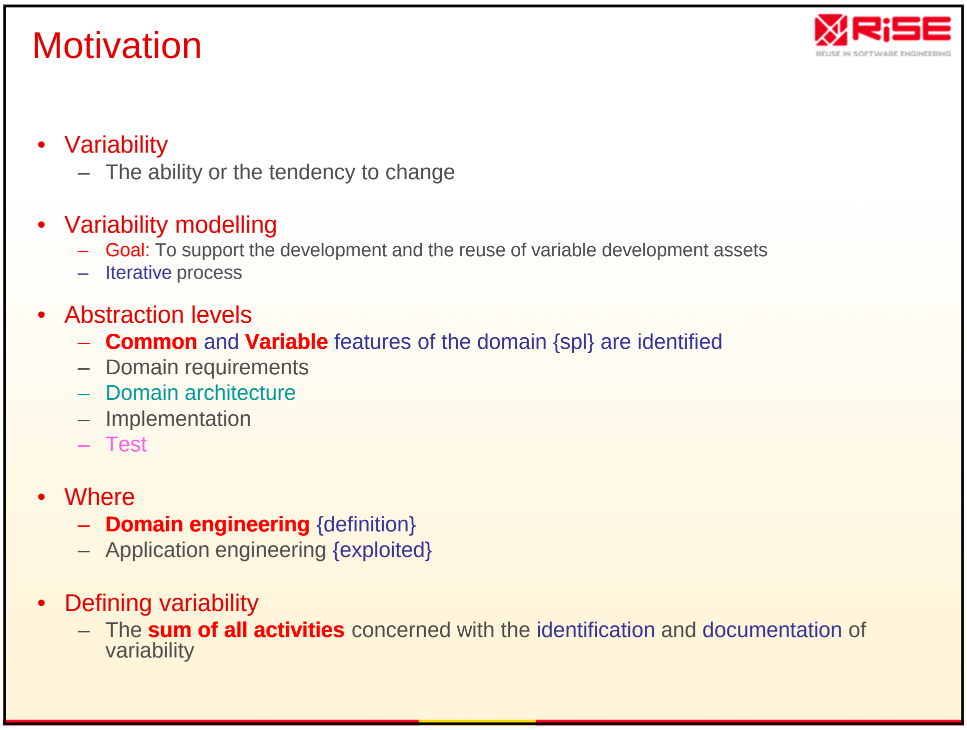


PHILIPS: Product Line of SW for TV Sets

- Current Scenario
  - Since 2002, all Philips' mid-range and high-end televisions have software derived from this product line
  - The product line supports three different hardware platforms
- Results and Metrics
  - **Today, there are 20 different software releases per year, where each release serving 1-5 different product types**

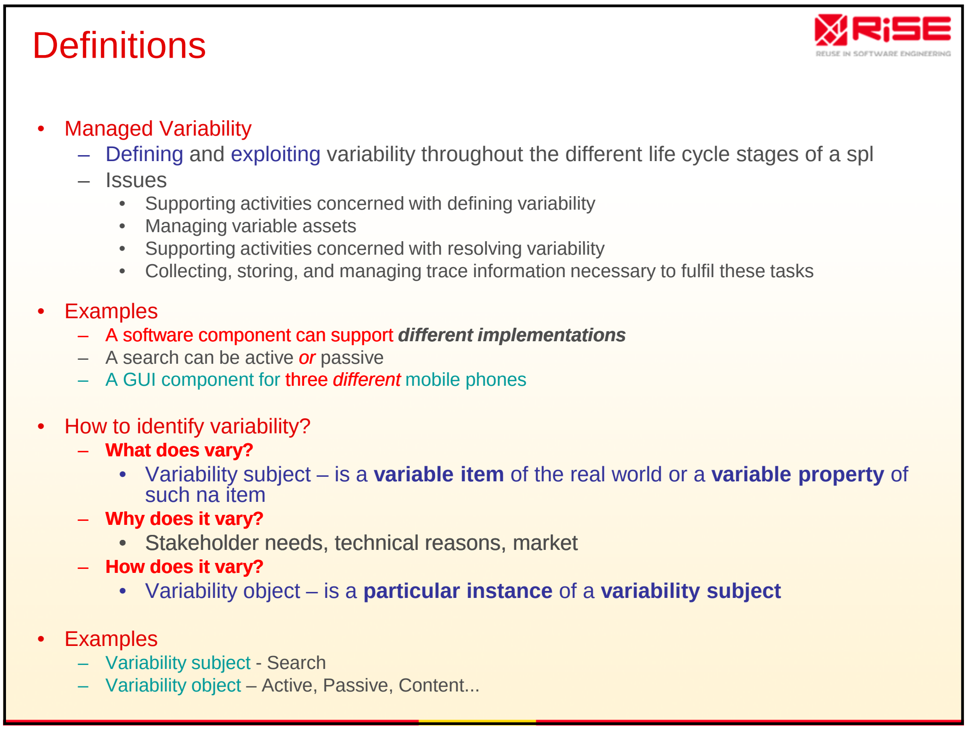


Background – Important Concepts



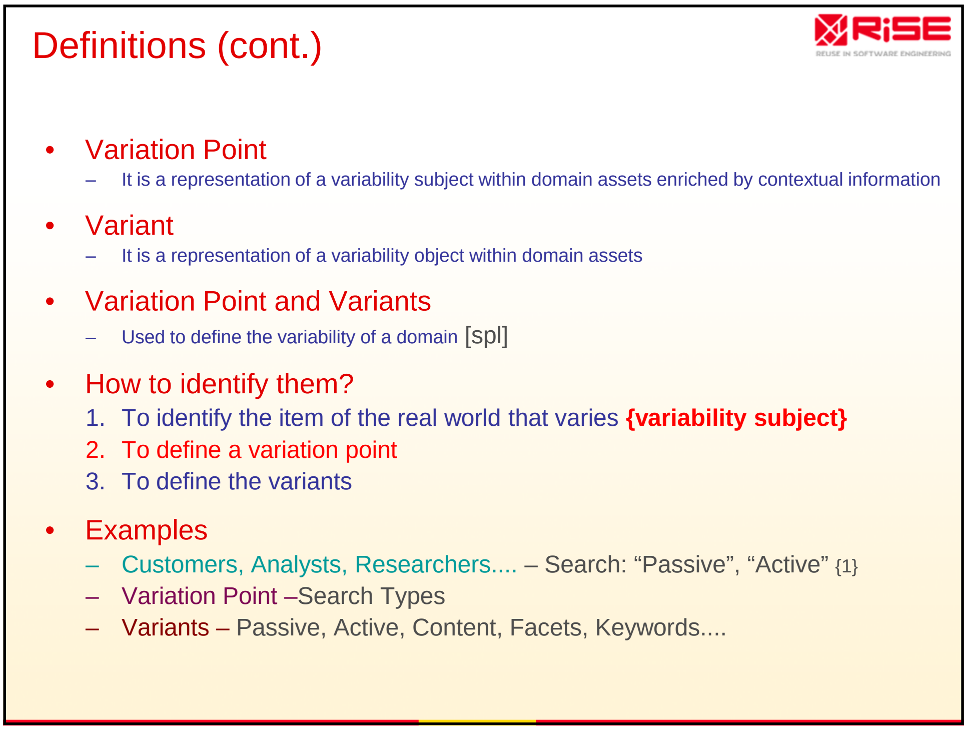
Motivation

- Variability
  - The ability or the tendency to change
- Variability modelling
  - Goal: To support the development and the reuse of variable development assets
  - Iterative process
- Abstraction levels
  - **Common** and **Variable** features of the domain {spl} are identified
  - Domain requirements
  - Domain architecture
  - Implementation
  - Test
- Where
  - **Domain engineering** {definition}
  - Application engineering {exploited}
- Defining variability
  - The **sum of all activities** concerned with the identification and documentation of variability



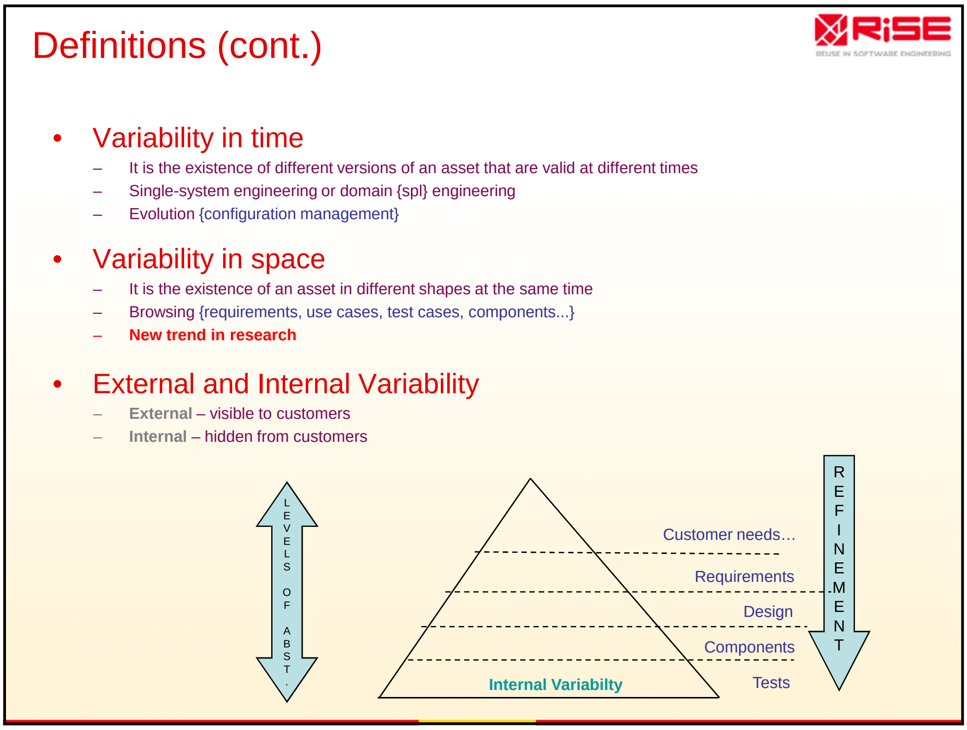
Definitions

- Managed Variability
  - Defining and exploiting variability throughout the different life cycle stages of a spl
  - Issues
    - Supporting activities concerned with defining variability
    - Managing variable assets
    - Supporting activities concerned with resolving variability
    - Collecting, storing, and managing trace information necessary to fulfil these tasks
- Examples
  - A software component can support ***different implementations***
  - A search can be active *or* passive
  - A GUI component for three *different* mobile phones
- How to identify variability?
  - **What does vary?**
    - Variability subject – is a **variable item** of the real world or a **variable property** of such na item
  - **Why does it vary?**
    - Stakeholder needs, technical reasons, market
  - **How does it vary?**
    - Variability object – is a **particular instance** of a **variability subject**
- Examples
  - Variability subject - Search
  - Variability object – Active, Passive, Content...



Definitions (cont.)

- Variation Point
  - It is a representation of a variability subject within domain assets enriched by contextual information
- Variant
  - It is a representation of a variability object within domain assets
- Variation Point and Variants
  - Used to define the variability of a domain [spl]
- How to identify them?
  1. To identify the item of the real world that varies **{variability subject}**
  2. To define a variation point
  3. To define the variants
- Examples
  - Customers, Analysts, Researchers.... – Search: “Passive”, “Active” {1}
  - Variation Point –Search Types
  - Variants – Passive, Active, Content, Facets, Keywords....



Definitions (cont.)

- Variability in time
  - It is the existence of different versions of an asset that are valid at different times
  - Single-system engineering or domain {spl} engineering
  - Evolution {configuration management}
- Variability in space
  - It is the existence of an asset in different shapes at the same time
  - Browsing {requirements, use cases, test cases, components...}
  - **New trend in research**
- External and Internal Variability
  - **External** – visible to customers
  - **Internal** – hidden from customers

LEVELS OF ABST.

Customer needs…

Requirements

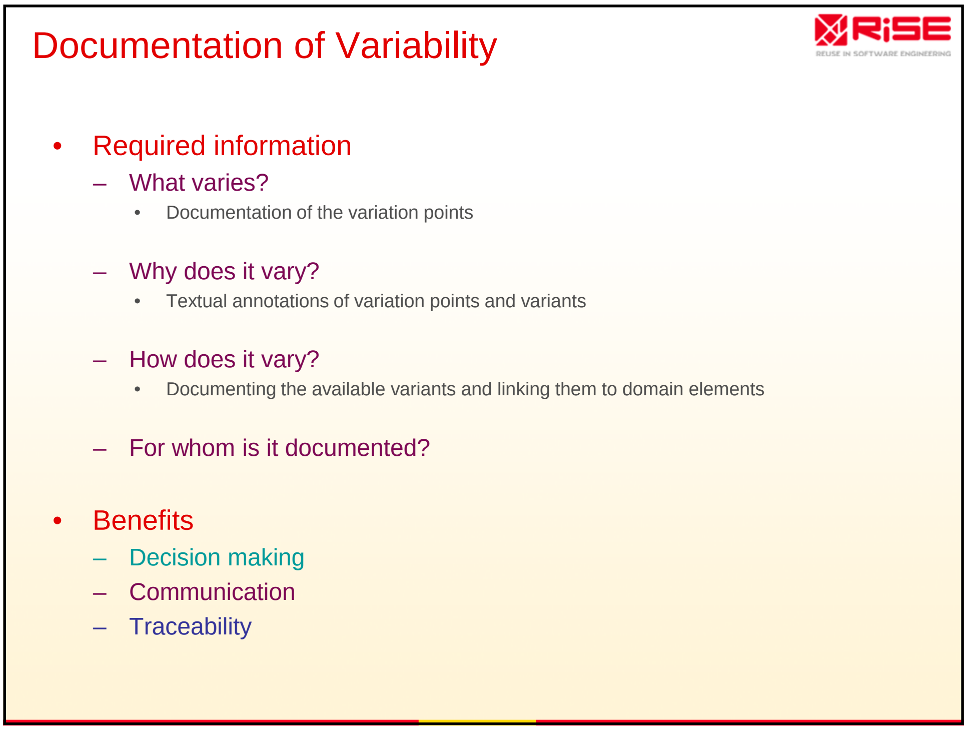
Design

Components

Tests

Internal Variabilty

REFINEMENT



Documentation of Variability

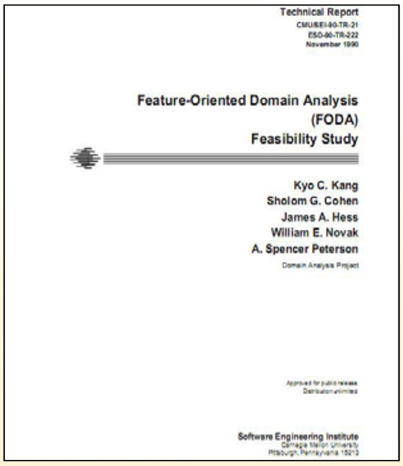
- Required information
  - What varies?
    - Documentation of the variation points
  - Why does it vary?
    - Textual annotations of variation points and variants
  - How does it vary?
    - Documenting the available variants and linking them to domain elements
  - For whom is it documented?
- Benefits
  - Decision making
  - Communication
  - Traceability

Variability constraints

- Variant constraint dependency
  - Variant *requires* variant
    - Notification x Interest
  - Variant *excludes* variant
- Variant to Variation Point constraint dependency
  - Variant *requires* variation point
    - Asset publish x Access control
  - Variant *excludes* variation point
- Variation Point constraint dependency
  - Variation Point *requires* Variation Point
    - Publish x Search
  - Variation Point *excludes* Variation Point

Features and Feature Model

- Feature
  - An end-user-visible characteristic of a system
  - A ***distinguishable characteristic*** of a concept that is relevant to some stakeholder of the concept
- Elements
  - Feature diagram
  - Feature definitions
  - Composition rules
  - Rationale for features



Technical Report
CMU/SEI-90-TR-21
ESD-90-TR-222
November 1990

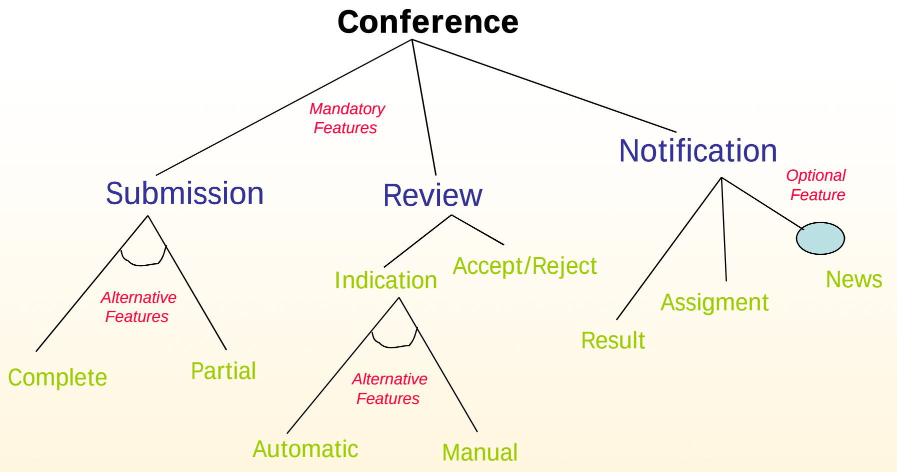
Feature-Oriented Domain Analysis
(FODA)
Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson
Domain Analysis Project

Approved for public release
Distribution unlimited

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Example



Conference

Mandatory Features

Submission

Review

Notification

Optional Feature

Alternative Features

Complete

Partial

Indication

Accept/Reject

Alternative Features

Automatic

Manual

Result

Assigment

News

Feature Modeling: The importance

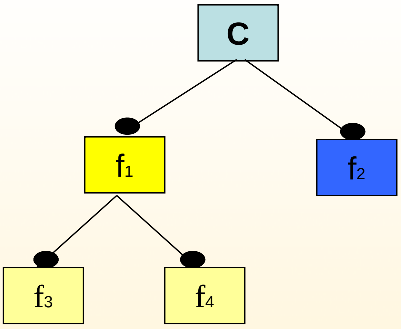
- Reusable software
  - Variability
- Key technique
  - To Identify and capture variability
- To avoid
  - Relevant features and variations points are not included in the reusable software
  - Many features and variations points are included but never used {complexity, costs}

Feature Models

- Represents the common and the variable features of concept instances and ...
- Dependencies between the variable features
- Elements
  - Feature Diagram
  - Semantic descriptions of each features
  - Client programs
  - Exemplar systems
  - Constraints
  - Priorities

Feature Diagrams (cont.)

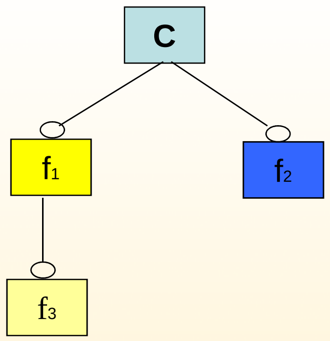
Mandatory Features



Feature set $\{C, f_1, f_2, f_3, f_4\}$

Feature Diagrams (cont.)

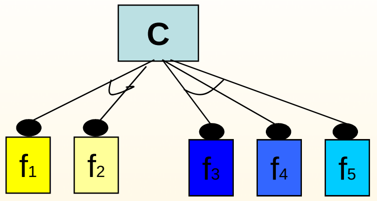
Optional Features



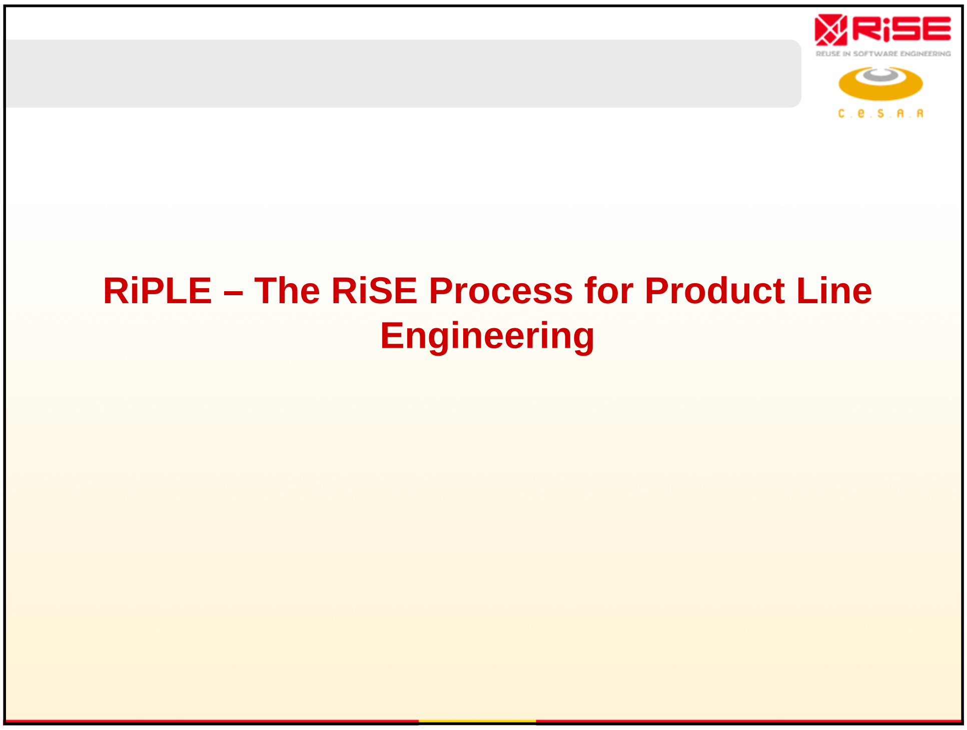
Feature set *{C} , {C, f_1}, {C, $f_{1,} f_3$}, {C, f_2}, {C, f_1, f_2}, {C, $f_{1,}$ f_3, f_2 }*

Feature Diagrams (cont.)

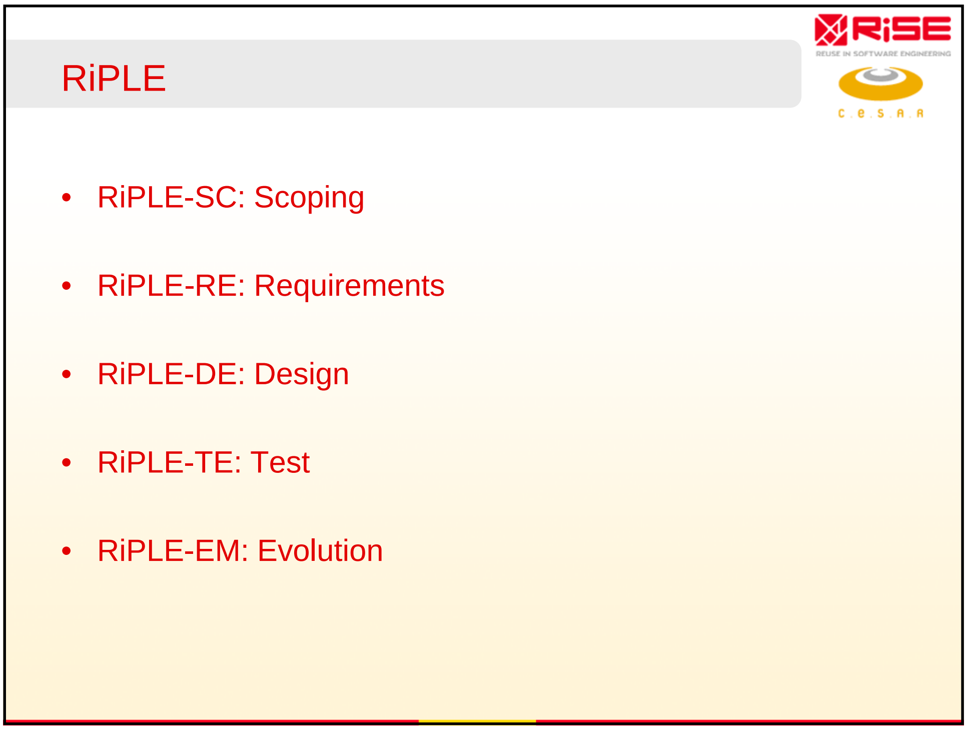
Alternative Features



Feature set *{C, $f_{1,} f_3$} , {C, $f_{1,} f_4$}, {C, $f_{1,} f_5$}, {C, $f_{2,} f_3$}, {C, f_2, f_4}, {C, f_2, f_5 }*



RiPLE – The RiSE Process for Product Line Engineering



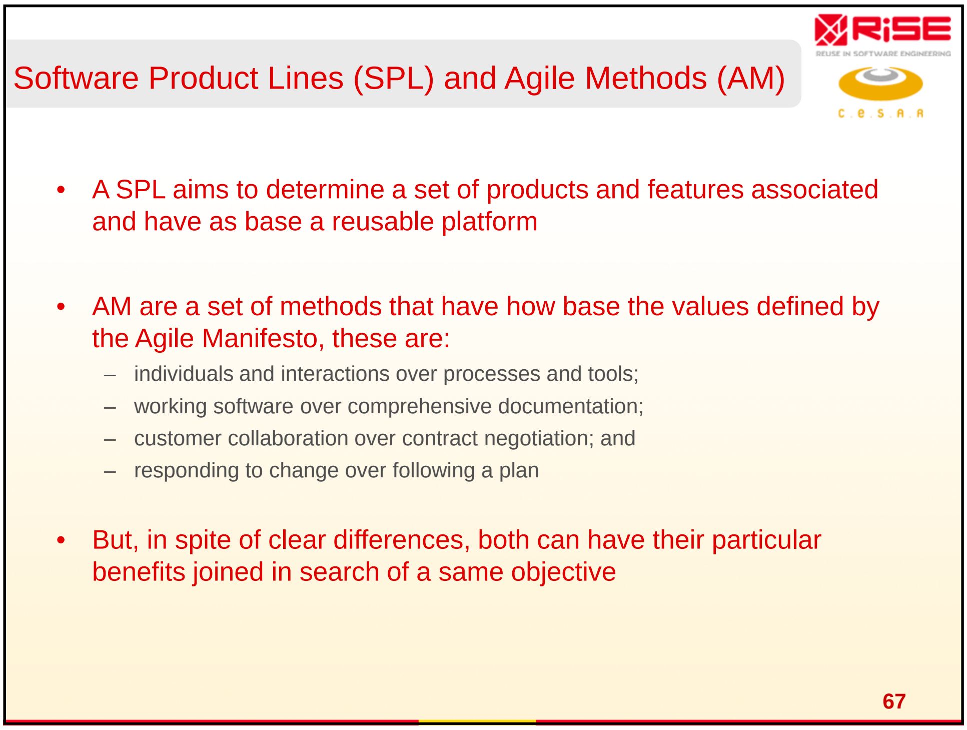
RiPLE

- RiPLE-SC: Scoping
- RiPLE-RE: Requirements
- RiPLE-DE: Design
- RiPLE-TE: Test
- RiPLE-EM: Evolution

RiPLE-SC: Scoping Process

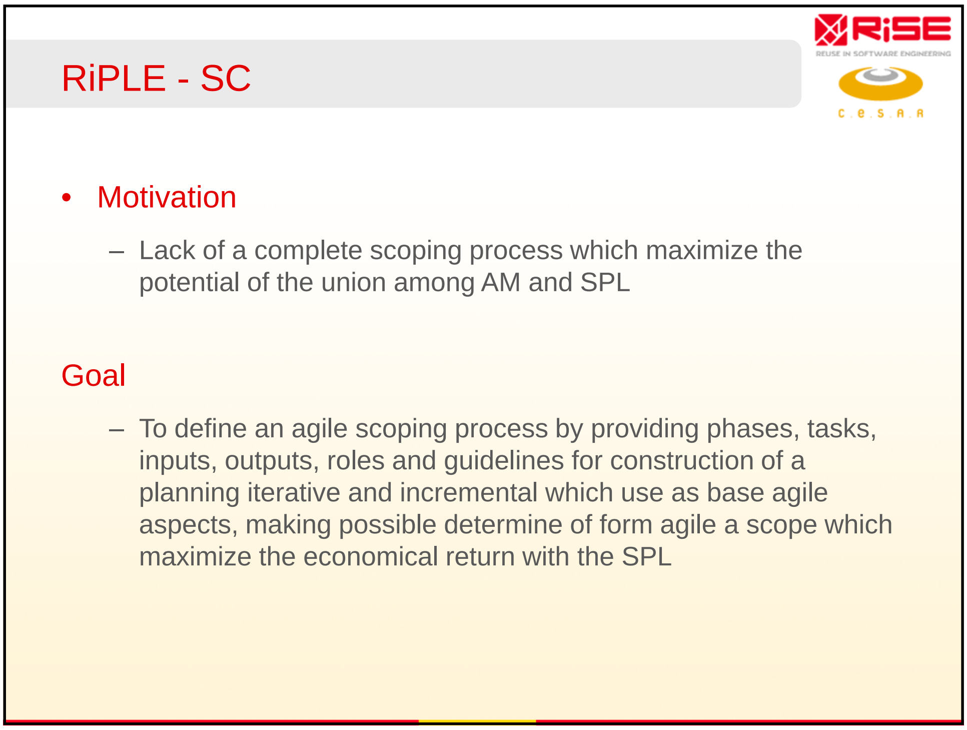
Scoping on SPL

- What is Scoping?
  - It is the initial phase of a SPL
  - It aims to identify products, features, potential of the domain and reusable assets
- Why Scoping?
  - It determine the viability of the SPL
  - It maximizes the economical value of the SPL



Software Product Lines (SPL) and Agile Methods (AM)

- A SPL aims to determine a set of products and features associated and have as base a reusable platform
- AM are a set of methods that have how base the values defined by the Agile Manifesto, these are:
  - individuals and interactions over processes and tools;
  - working software over comprehensive documentation;
  - customer collaboration over contract negotiation; and
  - responding to change over following a plan
- But, in spite of clear differences, both can have their particular benefits joined in search of a same objective



RiPLE - SC

- Motivation
  - Lack of a complete scoping process which maximize the potential of the union among AM and SPL

Goal

- To define an agile scoping process by providing phases, tasks, inputs, outputs, roles and guidelines for construction of a planning iterative and incremental which use as base agile aspects, making possible determine of form agile a scope which maximize the economical return with the SPL

RiPLE - SC :: Overview

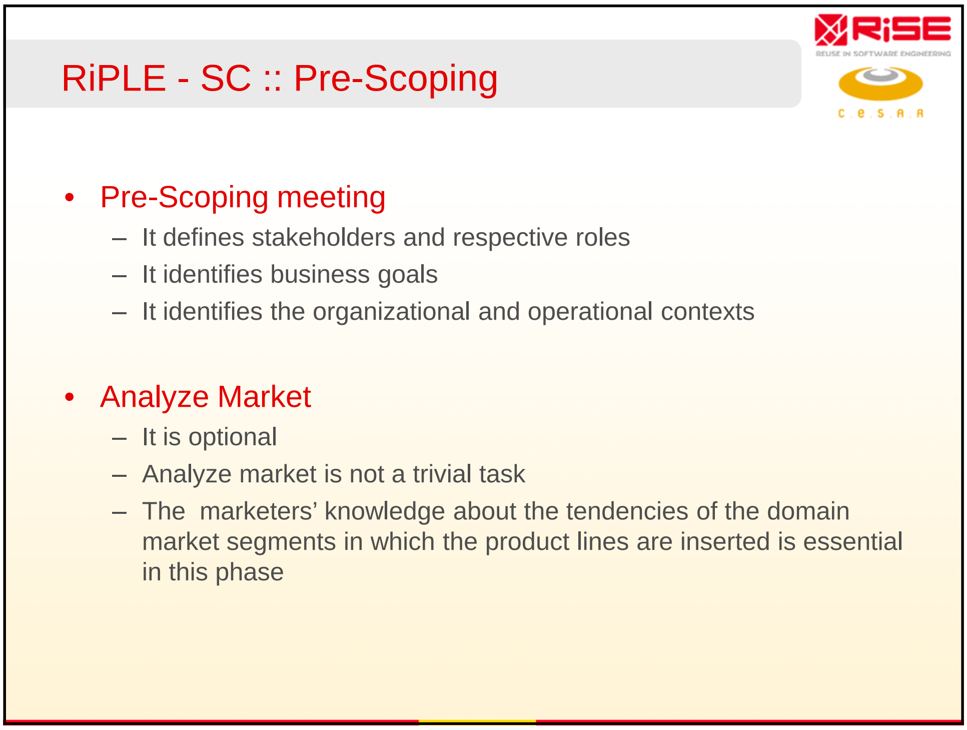
- It is performed in an iterative and incremental way using agile values, principles and techniques

- It is defined in a systematic way

RiPLE - SC :: Overview

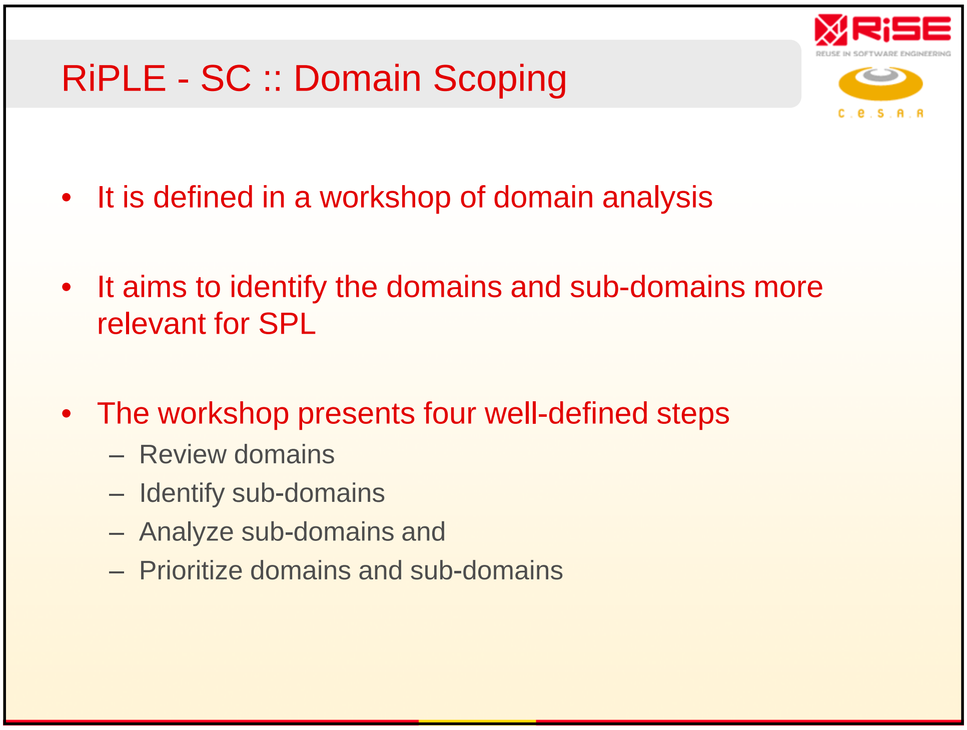
- The RiPLE-SC consists of four main phases
  - Pre-Scoping
  - Domain Scoping
  - Product Scoping
  - Assets Scoping

- The roles are relevant in these phases
  - Scoping expert
  - Customer
  - Domain expert
  - Marketer
  - Developer
  - Architect
  - SPL manager



RiPLE - SC :: Pre-Scoping

- Pre-Scoping meeting
  - It defines stakeholders and respective roles
  - It identifies business goals
  - It identifies the organizational and operational contexts
- Analyze Market
  - It is optional
  - Analyze market is not a trivial task
  - The marketers' knowledge about the tendencies of the domain market segments in which the product lines are inserted is essential in this phase



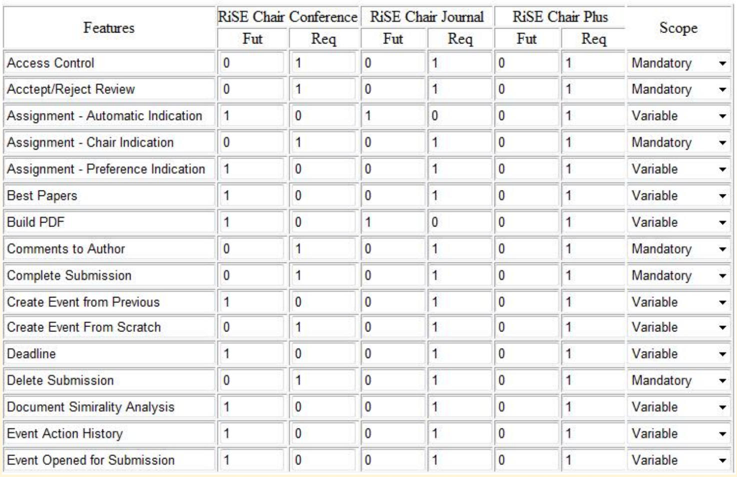
RiPLE - SC :: Domain Scoping

- It is defined in a workshop of domain analysis
- It aims to identify the domains and sub-domains more relevant for SPL
- The workshop presents four well-defined steps
  - Review domains
  - Identify sub-domains
  - Analyze sub-domains and
  - Prioritize domains and sub-domains

RiPLE - SC :: Product Scoping

- The goal is to identify the products more relevant for SPL and their features
- Five tasks
  - Identify products
  - Construct user stories
  - Identify features
  - Features review meeting and
  - Construct product map

RiPLE - SC :: Product Scoping – Product Map



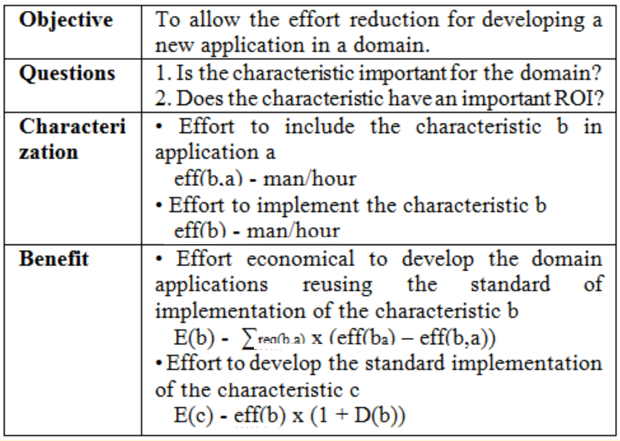
Features	RiSE Chair Conference		RiSE Chair Journal		RiSE Chair Plus		Scope
	Fut	Req	Fut	Req	Fut	Req	
Access Control	0	1	0	1	0	1	Mandatory
Acctept/Reject Review	0	1	0	1	0	1	Mandatory
Assignment - Automatic Indication	1	0	1	0	0	1	Variable
Assignment - Chair Indication	0	1	0	1	0	1	Mandatory
Assignment - Preference Indication	1	0	0	1	0	1	Variable
Best Papers	1	0	0	1	0	1	Variable
Build PDF	1	0	1	0	0	1	Variable
Comments to Author	0	1	0	1	0	1	Mandatory
Complete Submission	0	1	0	1	0	1	Mandatory
Create Event from Previous	1	0	0	1	0	1	Variable
Create Event From Scratch	0	1	0	1	0	1	Variable
Deadline	1	0	0	1	0	1	Variable
Delete Submission	0	1	0	1	0	1	Mandatory
Document Simirality Analysis	1	0	0	1	0	1	Variable
Event Action History	1	0	0	1	0	1	Variable
Event Opened for Submission	1	0	0	1	0	1	Variable

RiPLE - SC :: Assets Scoping

- The goal of the assets scoping is to determine an appropriate set of assets for product line
- Three tasks:
  - Create metrics of characterization and benefit
  - Apply metrics
  - Prioritize product map

RiPLE - SC :: Assets Scoping

- Example of characterization and benefit metrics in a generic domain:



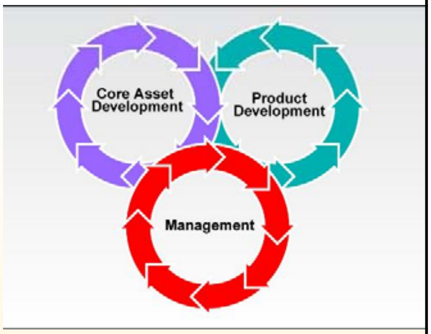
Objective	To allow the effort reduction for developing a new application in a domain.
Questions	1. Is the characteristic important for the domain? 2. Does the characteristic have an important ROI?
Characterization	• Effort to include the characteristic b in application a eff(b.a) - man/hour • Effort to implement the characteristic b eff(b) - man/hour

Benefit	• Effort economical to develop the domain applications reusing the standard of implementation of the characteristic b
	E(b) - $\sum_{req(b.a)}$ x (eff(b_a) – eff(b,a))
	• Effort to develop the standard implementation of the characteristic c
	E(c) - eff(b) x (1 + D(b))

RiPLE-RE: Requirements Engineering

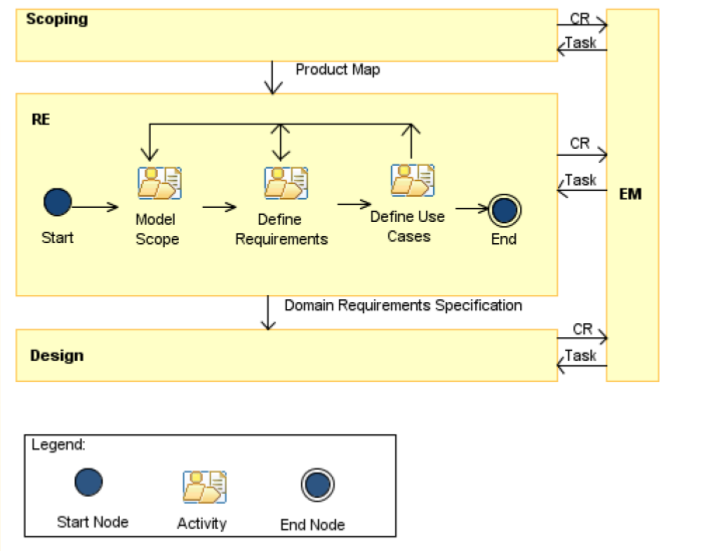
Introduction

- **Software Product Lines**
  - Activities
- **Requirements Engineering**
  - More products and stakeholders
  - Attention to **variabilities** and **commonalities**
  - Lack of a systematic process



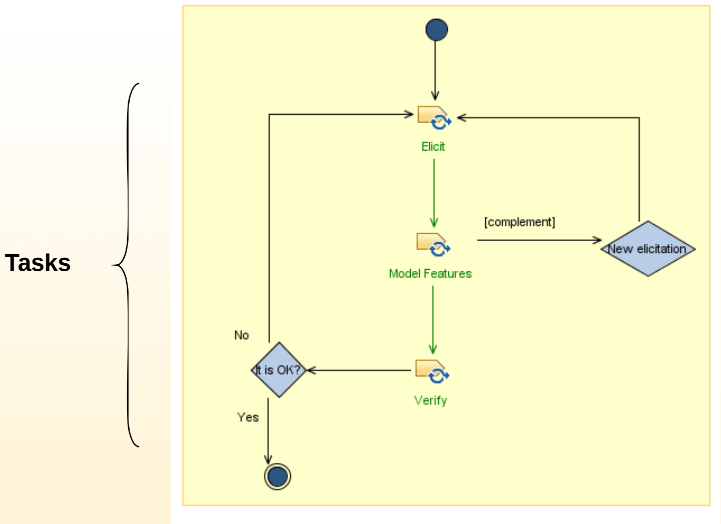
(Clements and Northrop, 2001)

RiPLE-RE :: Overview



79

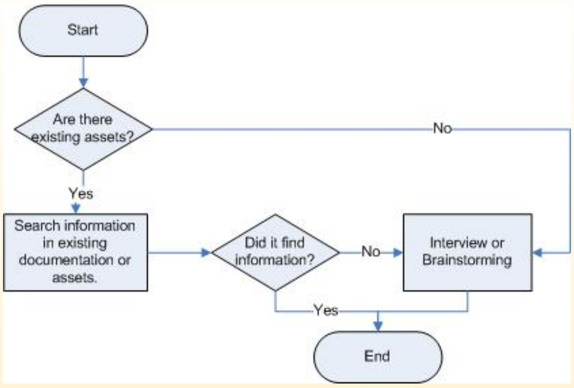
RiPLE-RE :: Activity Model Scope



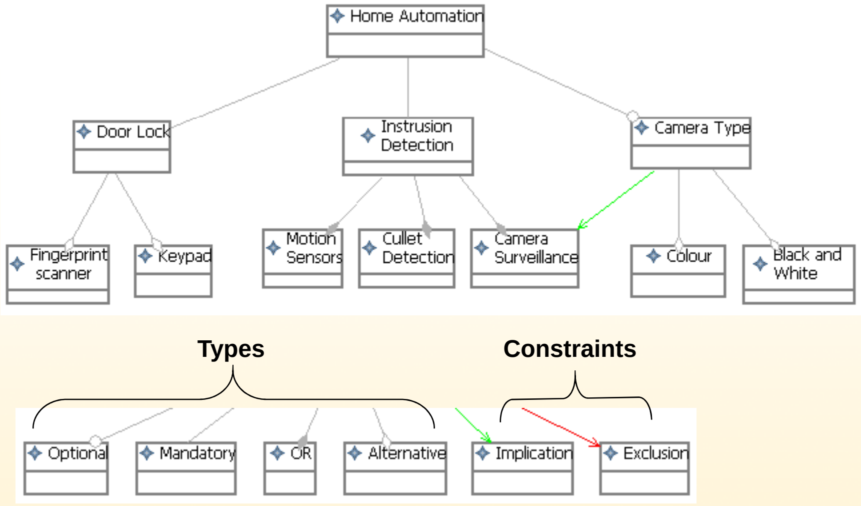
80

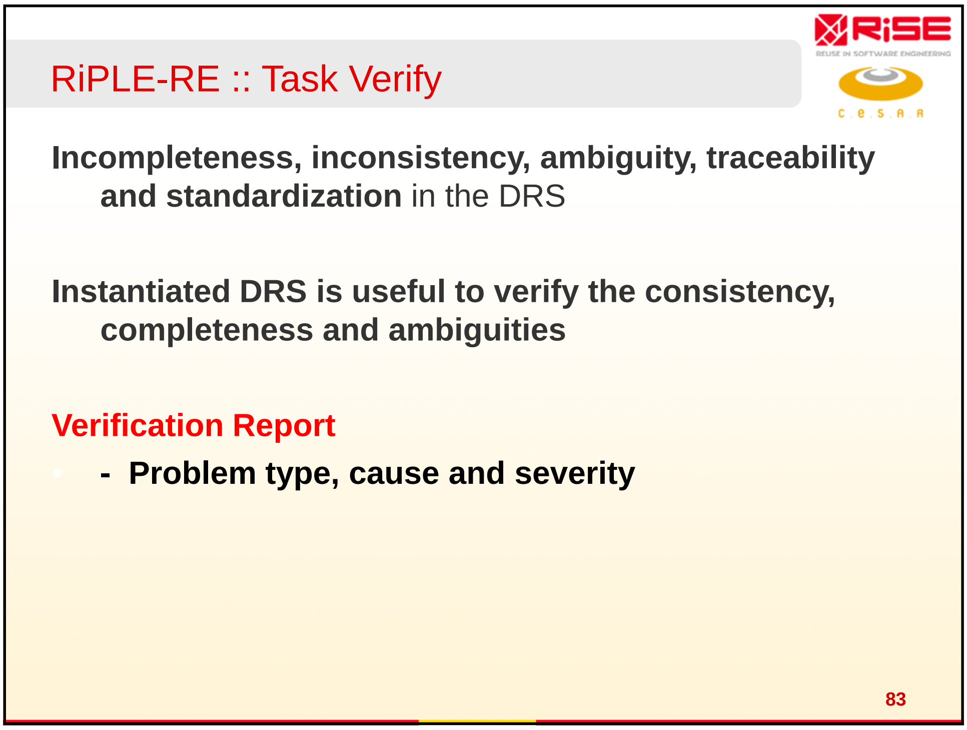
RiPLE-RE :: Task Elicit

- Identifying commonalities and variabilities
- Identifying future needs
- Information source x Context


Start
Are there existing assets?
No
Yes
Search information in existing documentation or assets.
Did it find information?
No
Interview or Brainstorming
Yes
End

RiPLE-RE :: Task Model Feature





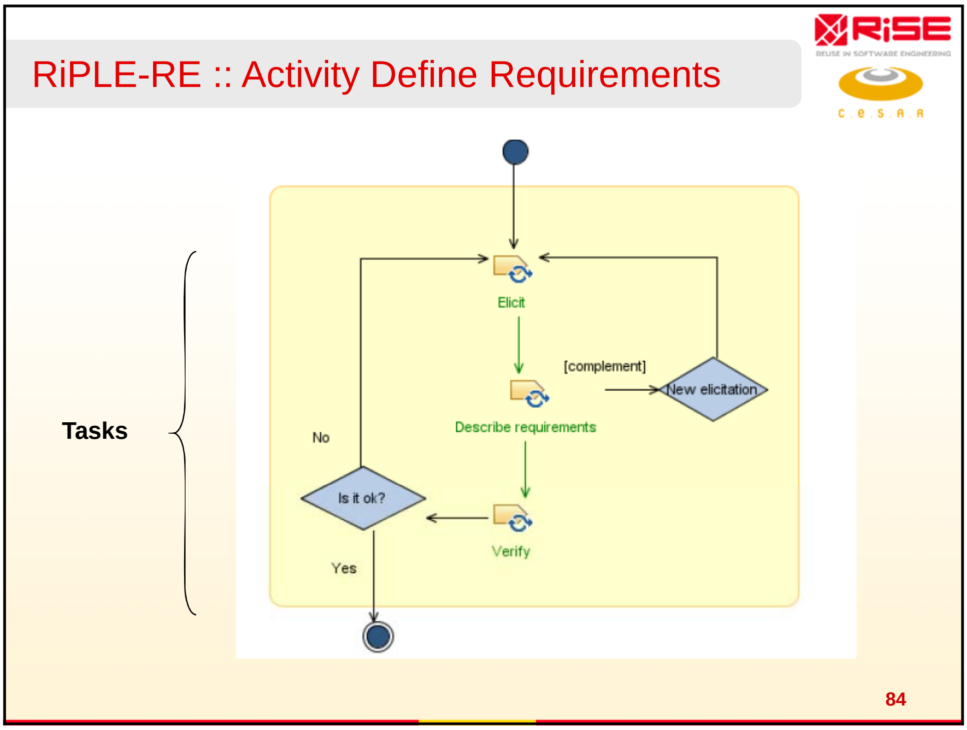
RiPLE-RE :: Task Verify

Incompleteness, inconsistency, ambiguity, traceability and standardization in the DRS

Instantiated DRS is useful to verify the consistency, completeness and ambiguities

Verification Report

- **Problem type, cause and severity**



RiPLE-RE :: Activity Define Requirements

RiPLE-RE :: Task Describe Requirements [1]

Variability Scope

- Whole requirement (Mandatory and variant)
- Requirement text fragment (Variation Point)

Requirement Attributes

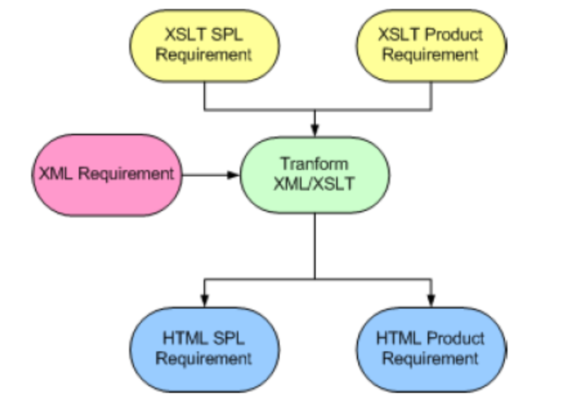
- Id
- Type
- Name
- **Variability Type**
- **Binding Time**
- Priority
- Rationale
- Description
- **Implication**
- **Exclusion**

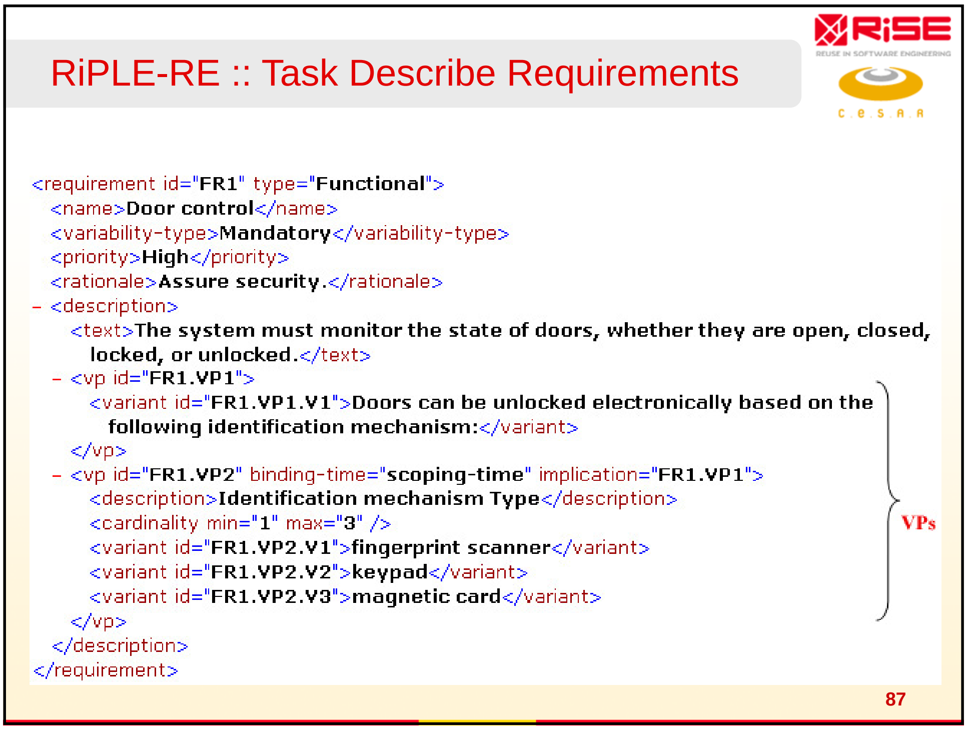
Point Variation Attributes

- Id
- Description
- Variants
- Cardinality
- Binding Time
- Implication
- Exclusion

RiPLE-RE :: Task Describe Requirements [3]

- Template in XML

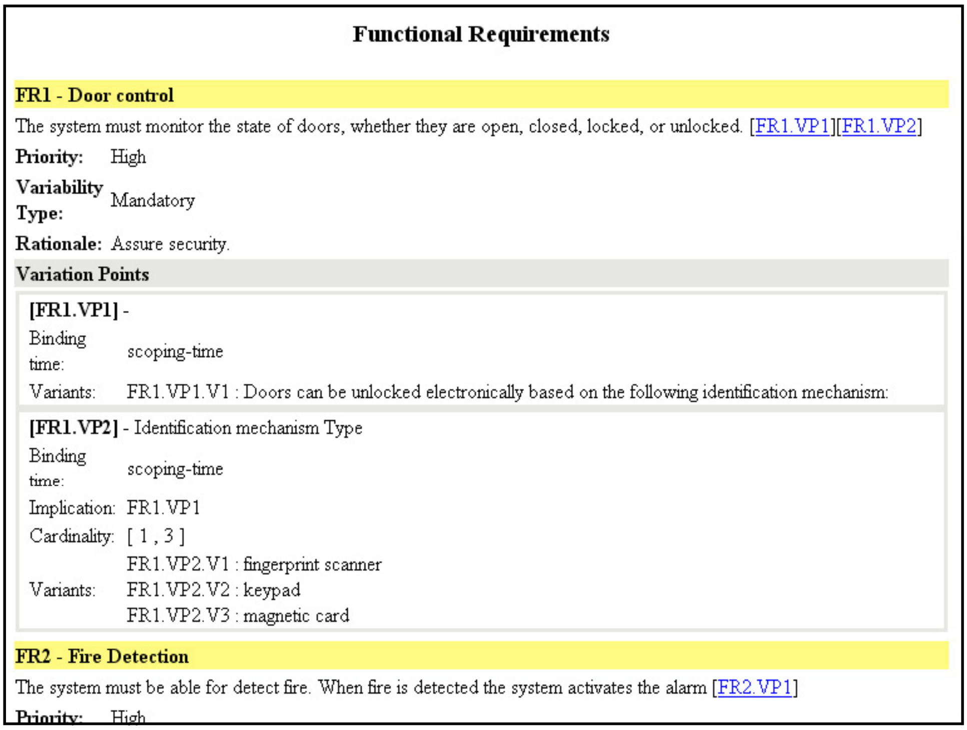




RiPLE-RE :: Task Describe Requirements

```
<requirement id="FR1" type="Functional">
  <name>Door control</name>
  <variability-type>Mandatory</variability-type>
  <priority>High</priority>
  <rationale>Assure security.</rationale>
- <description>
    <text>The system must monitor the state of doors, whether they are open, closed,
      locked, or unlocked.</text>
  - <vp id="FR1.VP1">
      <variant id="FR1.VP1.V1">Doors can be unlocked electronically based on the
        following identification mechanism:</variant>
    </vp>
  - <vp id="FR1.VP2" binding-time="scoping-time" implication="FR1.VP1">
      <description>Identification mechanism Type</description>
      <cardinality min="1" max="3" />
      <variant id="FR1.VP2.V1">fingerprint scanner</variant>
      <variant id="FR1.VP2.V2">keypad</variant>
      <variant id="FR1.VP2.V3">magnetic card</variant>
    </vp>
  </description>
</requirement>
```

87



Functional Requirements

FR1 - Door control

The system must monitor the state of doors, whether they are open, closed, locked, or unlocked. [FR1.VP1][FR1.VP2]

Priority:	High
Variability Type:	Mandatory
Rationale:	Assure security.

Variation Points

[FR1.VP1] -

Binding time:	scoping-time
Variants:	FR1.VP1.V1 : Doors can be unlocked electronically based on the following identification mechanism:

[FR1.VP2] - Identification mechanism Type

Binding time:	scoping-time
Implication:	FR1.VP1
Cardinality:	[1 , 3]
Variants:	FR1.VP2.V1 : fingerprint scanner FR1.VP2.V2 : keypad FR1.VP2.V3 : magnetic card

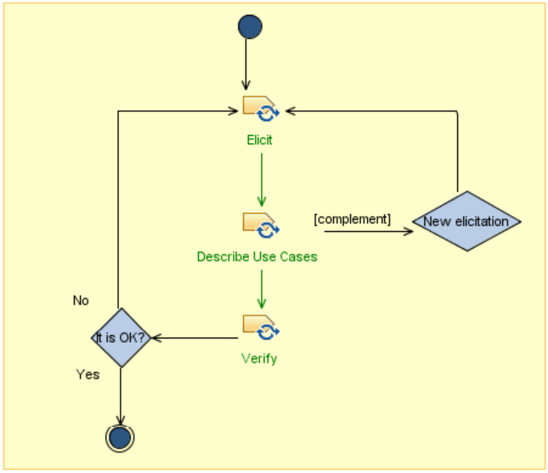
FR2 - Fire Detection

The system must be able for detect fire. When fire is detected the system activates the alarm [FR2.VP1]

Priority: High

RiPLE-RE :: Activity Define Use Cases

Tasks



RiPLE-RE :: Task Describe Use Cases [1]

- **Variability Scope**
  - Whole use case (Mandatory and variant)
  - Use case part (Variation Point)

Use Case Attributes

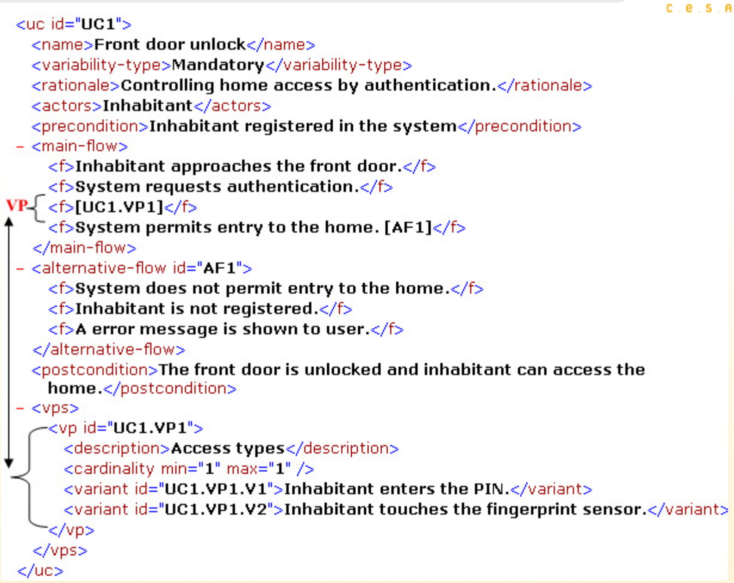
- Id
- Name
- **Variability Type**
- **Binding Time**
- Rationale
- Actors
- **Dependency**
- Preconditions
- Main Flow
- Alternative Flow
- Exception Flow
- Post Conditions
- **Implication**
- **Exclusion**

Point Variation Attributes

- Id
- Description
- Cardinality
- Variants
- Binding Time
- Implication
- Exclusion

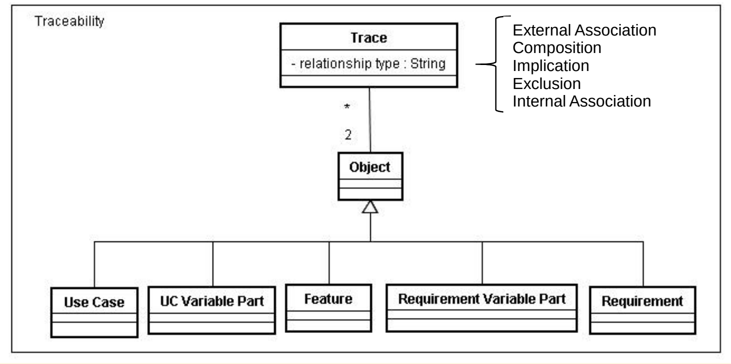
RiPLE-RE :: Task Describe Use Case

Variation Point
in small variations



```
<uc id="UC1">
  <name>Front door unlock</name>
  <variability-type>Mandatory</variability-type>
  <rationale>Controlling home access by authentication.</rationale>
  <actors>Inhabitant</actors>
  <precondition>Inhabitant registered in the system</precondition>
- <main-flow>
    <f>Inhabitant approaches the front door.</f>
    <f>System requests authentication.</f>
    <f>[UC1.VP1]</f>
    <f>System permits entry to the home. [AF1]</f>
  </main-flow>
- <alternative-flow id="AF1">
    <f>System does not permit entry to the home.</f>
    <f>Inhabitant is not registered.</f>
    <f>A error message is shown to user.</f>
  </alternative-flow>
  <postcondition>The front door is unlocked and inhabitant can access the
    home.</postcondition>
- <vps>
    <vp id="UC1.VP1">
      <description>Access types</description>
      <cardinality min="1" max="1" />
      <variant id="UC1.VP1.V1">Inhabitant enters the PIN.</variant>
      <variant id="UC1.VP1.V2">Inhabitant touches the fingerprint sensor.</variant>
    </vp>
  </vps>
</uc>
```

RiPLE-RE :: Traceability

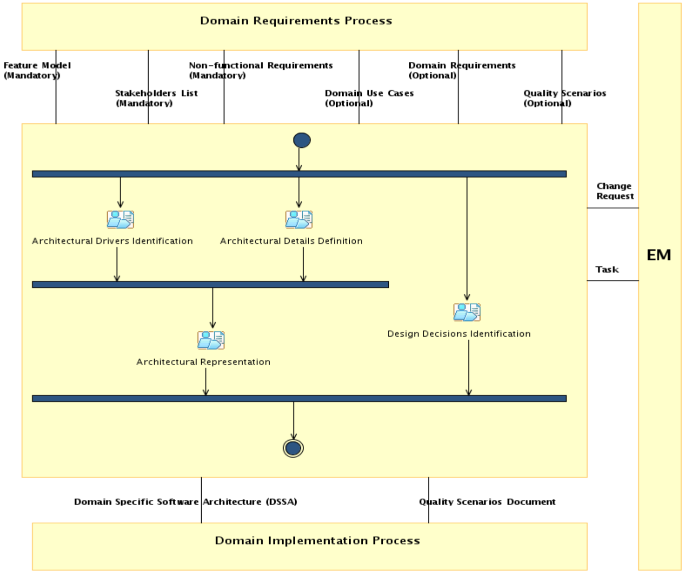


SPLiTT

Software Product Line Traceability Tool

RiPLE-DE - Design

RiPLE-DE Overview



Domain Requirements Process

Feature Model (Mandatory)

Stakeholders List (Mandatory)

Non-functional Requirements (Mandatory)

Domain Use Cases (Optional)

Domain Requirements (Optional)

Quality Scenarios (Optional)

Architectural Drivers Identification

Architectural Details Definition

Architectural Representation

Design Decisions Identification

Change Request

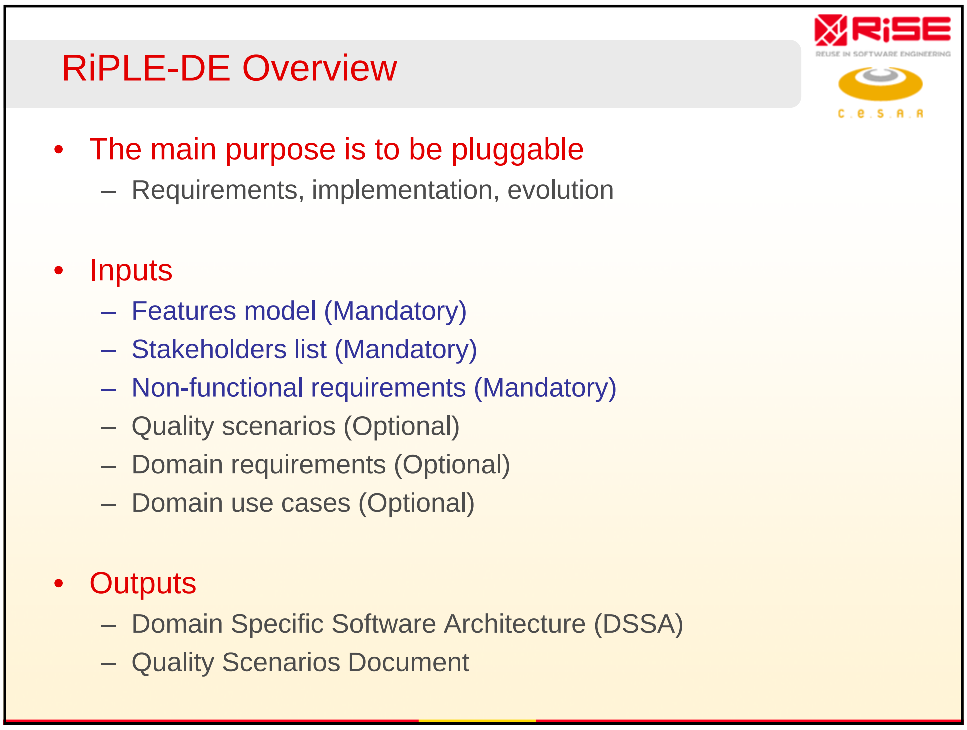
Task

EM

Domain Specific Software Architecture (DSSA)

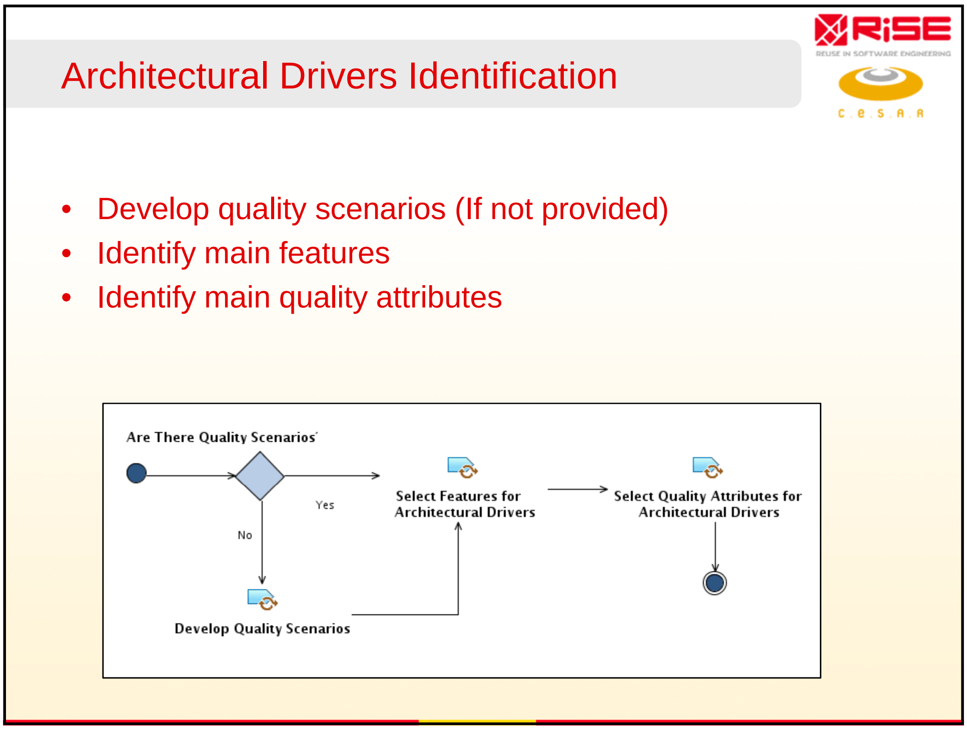
Quality Scenarios Document

Domain Implementation Process



RiPLE-DE Overview

- The main purpose is to be pluggable
  - Requirements, implementation, evolution
- Inputs
  - Features model (Mandatory)
  - Stakeholders list (Mandatory)
  - Non-functional requirements (Mandatory)
  - Quality scenarios (Optional)
  - Domain requirements (Optional)
  - Domain use cases (Optional)
- Outputs
  - Domain Specific Software Architecture (DSSA)
  - Quality Scenarios Document



Architectural Drivers Identification

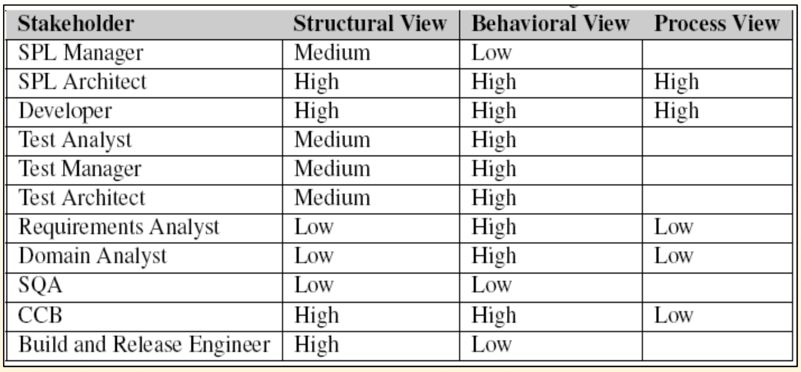
- Develop quality scenarios (If not provided)
- Identify main features
- Identify main quality attributes

Are There Quality Scenarios'
Yes
No
Select Features for Architectural Drivers
Select Quality Attributes for Architectural Drivers
Develop Quality Scenarios

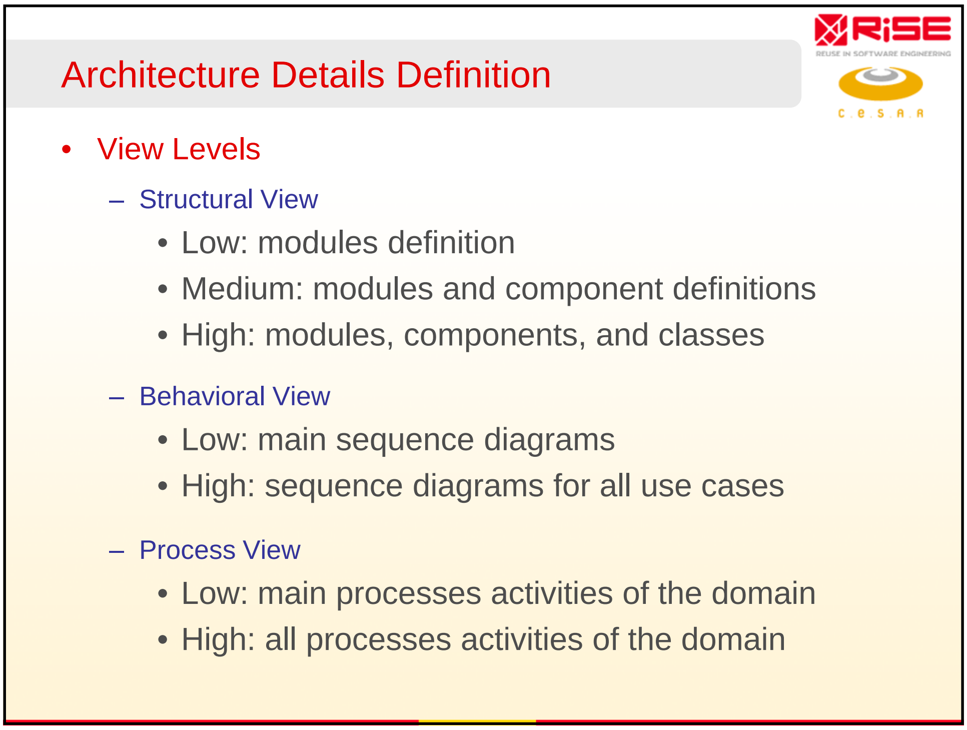
Architecture Details Definition

- Define which architectural views will be documented and in which level of details based on stakeholders list
  - Structural view (Mandatory)
  - Behavioral view (Mandatory)
  - Process view (Optional)

Architecture Details Definition

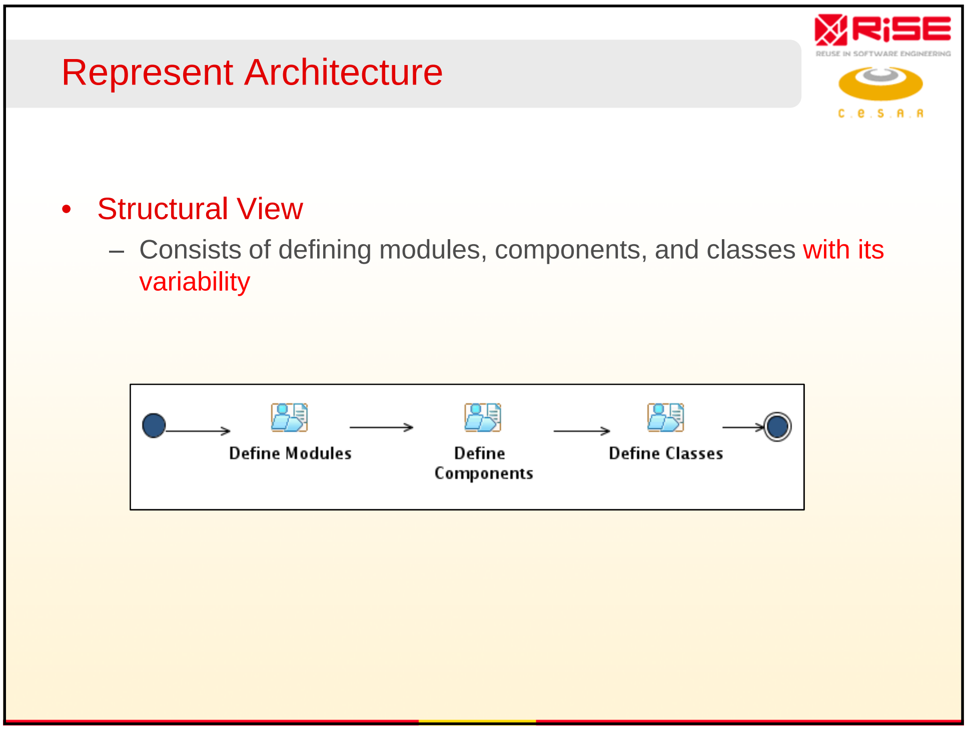


Stakeholder	Structural View	Behavioral View	Process View
SPL Manager	Medium	Low	
SPL Architect	High	High	High
Developer	High	High	High
Test Analyst	Medium	High	
Test Manager	Medium	High	
Test Architect	Medium	High	
Requirements Analyst	Low	High	Low
Domain Analyst	Low	High	Low
SQA	Low	Low	
CCB	High	High	Low
Build and Release Engineer	High	Low	



Architecture Details Definition

- View Levels
  - Structural View
    - Low: modules definition
    - Medium: modules and component definitions
    - High: modules, components, and classes
  - Behavioral View
    - Low: main sequence diagrams
    - High: sequence diagrams for all use cases
  - Process View
    - Low: main processes activities of the domain
    - High: all processes activities of the domain



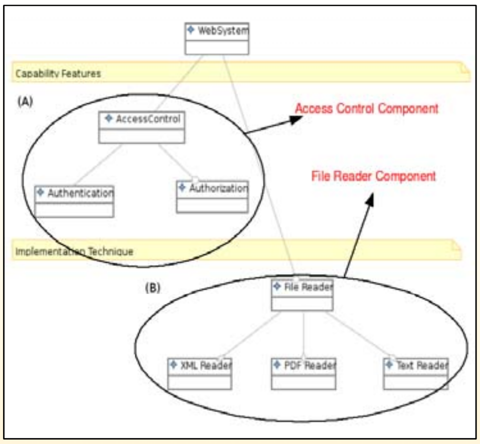
Represent Architecture

- Structural View
  - Consists of defining modules, components, and classes with its variability

Define Modules → Define Components → Define Classes

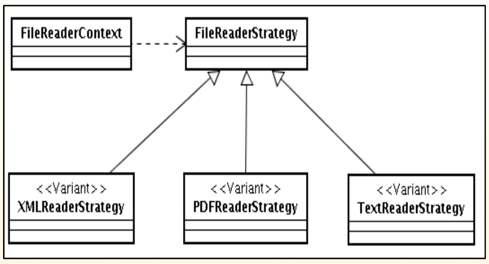
Structural View

- Components are defined based on the feature model
- Groups of features are defined to form a component
- The group top level feature defines the component variability



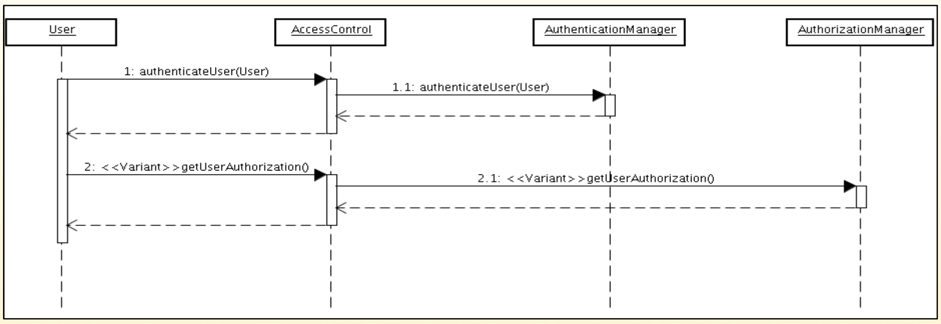
Structural View

- Classes are defined according to the features relationship using the guideline proposed by [Almeida2007]



Behavioral View

- Define sequence diagrams based on classes defined in structural view and the messages variability
- Variant messages are represented with the variant id stereotype.


User
AccessControl
AuthenticationManager
AuthorizationManager
1: authenticateUser(User)
1.1: authenticateUser(User)
2: <<Variant>>getUserAuthorization()
2.1: <<Variant>>getUserAuthorization()

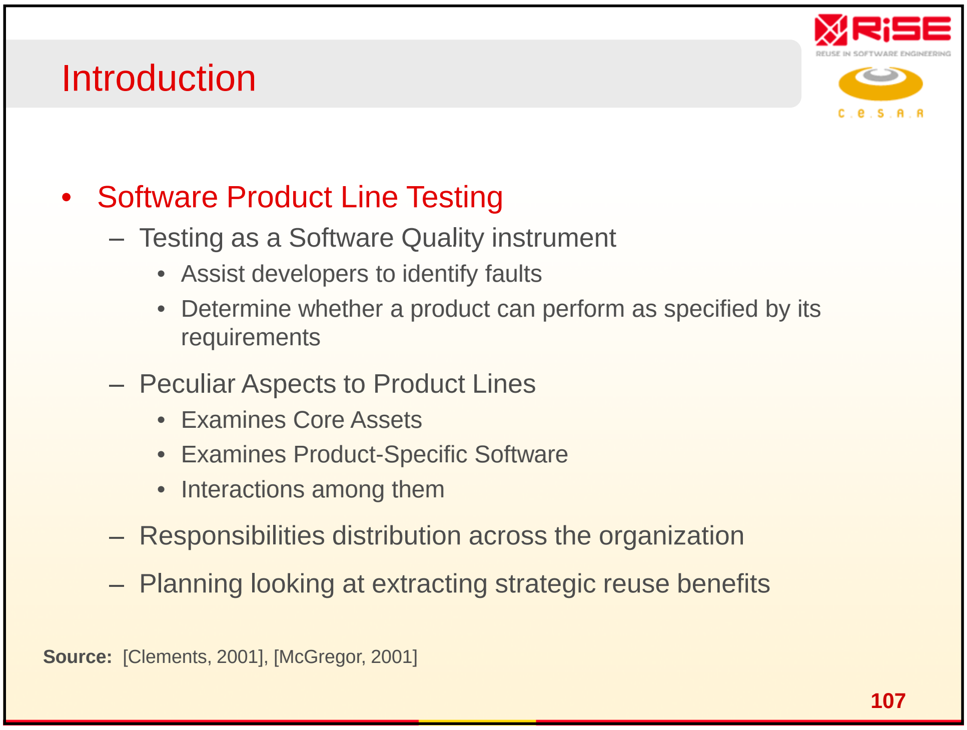
Process View

- Define the main process for the domain
- Define process variabilities
- It is not mandatory if the domain does not have complex processes

Identify Design Decisions

- Consists of identifying and recording the main architectural decisions for the domain
  - Ex.: Programming language
  - Application server
- Useful for avoiding architecture deprecation
- It can be defined during all the process life cycle

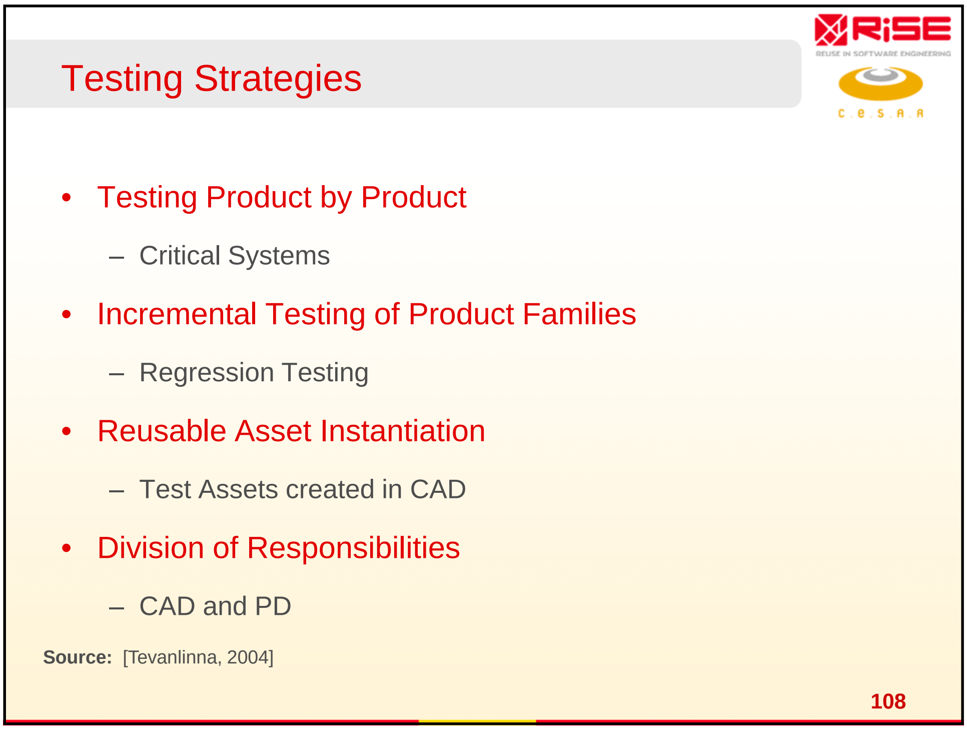
RiPLE-TE - Testing



Introduction

- Software Product Line Testing
  - Testing as a Software Quality instrument
    - Assist developers to identify faults
    - Determine whether a product can perform as specified by its requirements
  - Peculiar Aspects to Product Lines
    - Examines Core Assets
    - Examines Product-Specific Software
    - Interactions among them
  - Responsibilities distribution across the organization
  - Planning looking at extracting strategic reuse benefits

Source: [Clements, 2001], [McGregor, 2001]



Testing Strategies

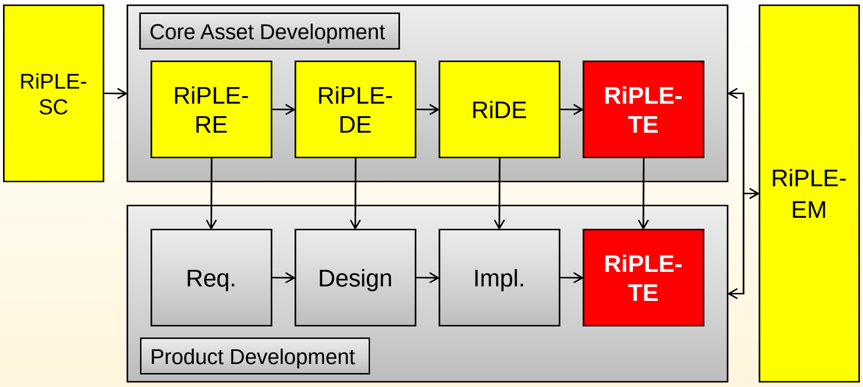
- Testing Product by Product
  - Critical Systems
- Incremental Testing of Product Families
  - Regression Testing
- Reusable Asset Instantiation
  - Test Assets created in CAD
- Division of Responsibilities
  - CAD and PD

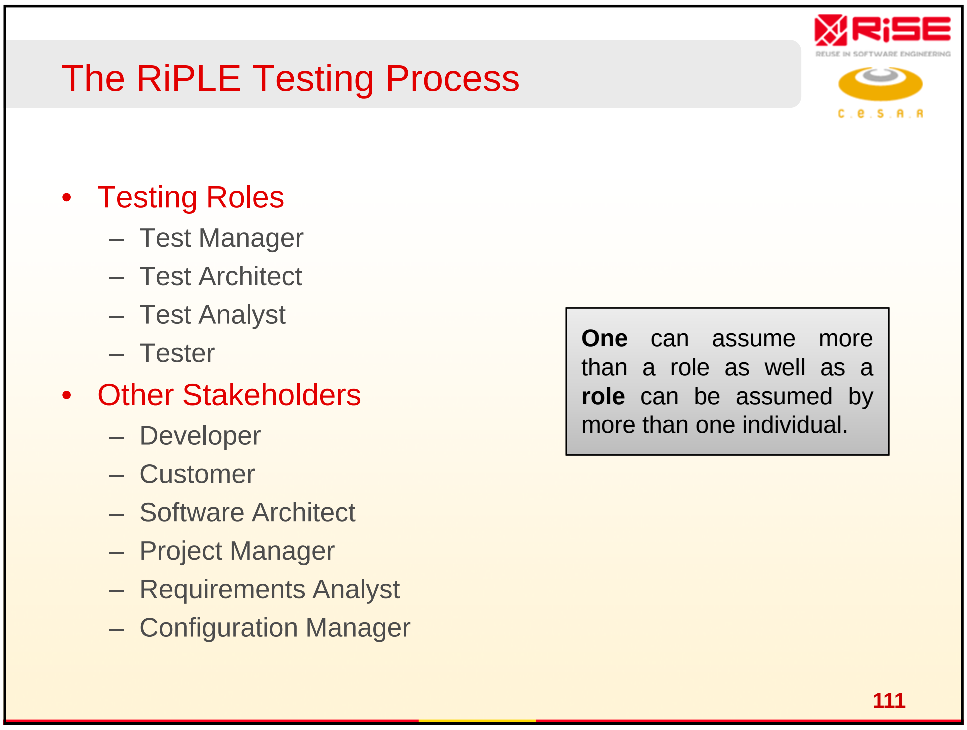
Source: [Tevanlinna, 2004]

The RiPLE Testing Process

- Testing considered along the Software Dev. Lifecycle
  - Testing Phases X Development Phases
  - Traceability is truly necessary
  - Variability concerns must be handled
- Interaction with other RIPLE disciplines
  - Scoping – RIPLE-SC
  - Requirements – RIPLE-RE
  - Analysis & Design – RIPLE-DE
  - Implementation - RIDE
  - Evolution – RIPLE-EM

The RiPLE Testing Process

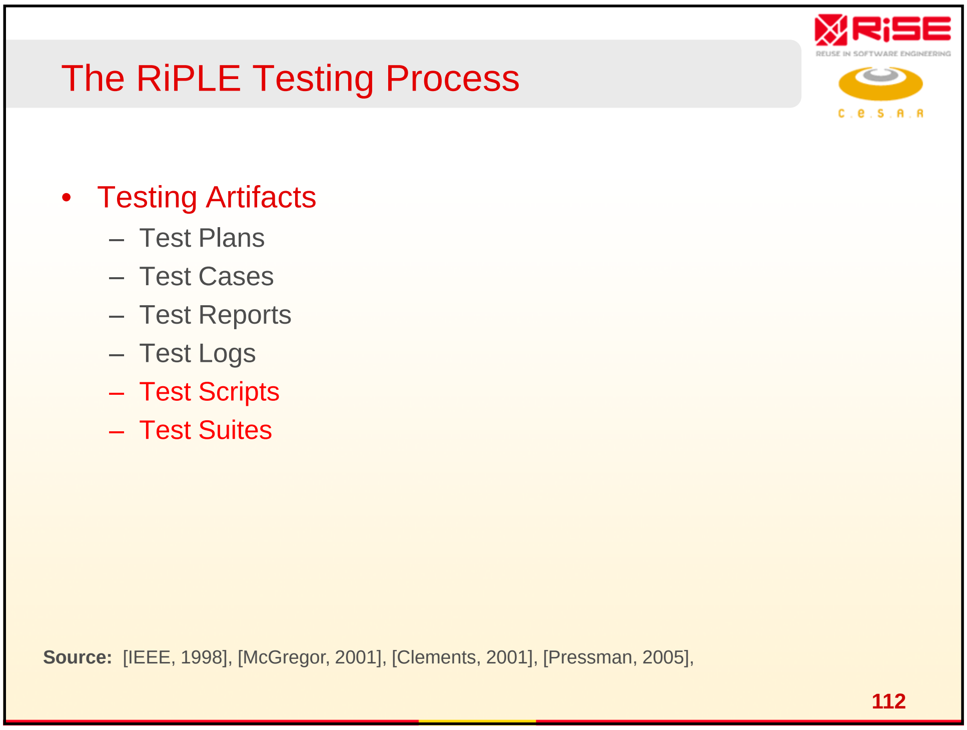




The RiPLE Testing Process

- Testing Roles
  - Test Manager
  - Test Architect
  - Test Analyst
  - Tester
- Other Stakeholders
  - Developer
  - Customer
  - Software Architect
  - Project Manager
  - Requirements Analyst
  - Configuration Manager

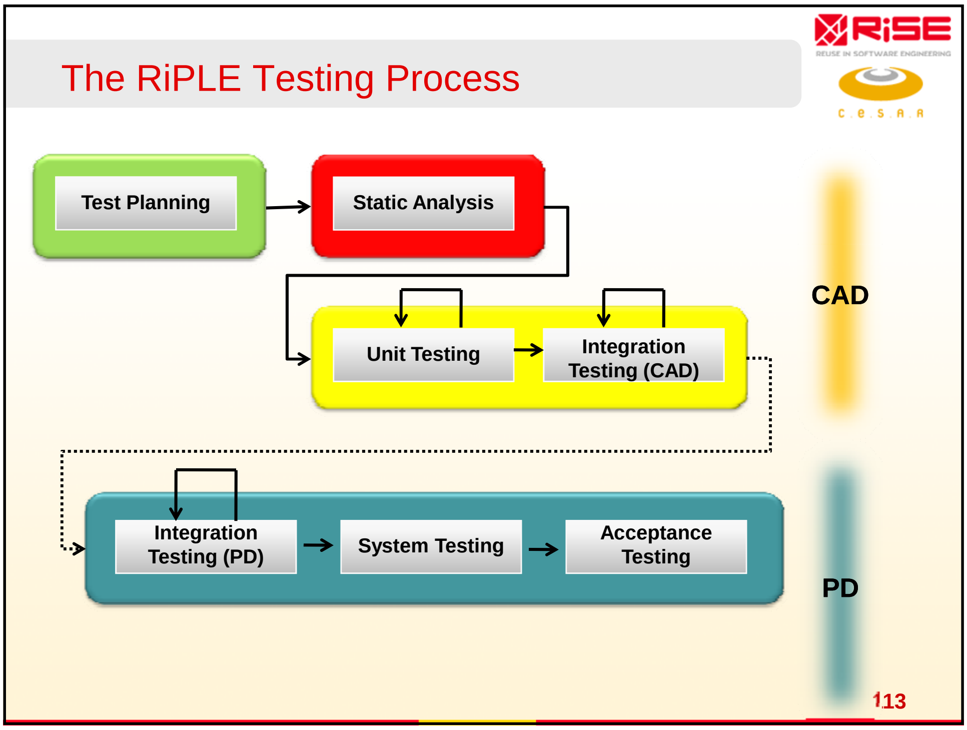
One can assume more than a role as well as a **role** can be assumed by more than one individual.



The RiPLE Testing Process

- Testing Artifacts
  - Test Plans
  - Test Cases
  - Test Reports
  - Test Logs
  - Test Scripts
  - Test Suites

Source: [IEEE, 1998], [McGregor, 2001], [Clements, 2001], [Pressman, 2005],



The RiPLE Testing Process

Test Planning

Static Analysis

Unit Testing

Integration Testing (CAD)

Integration Testing (PD)

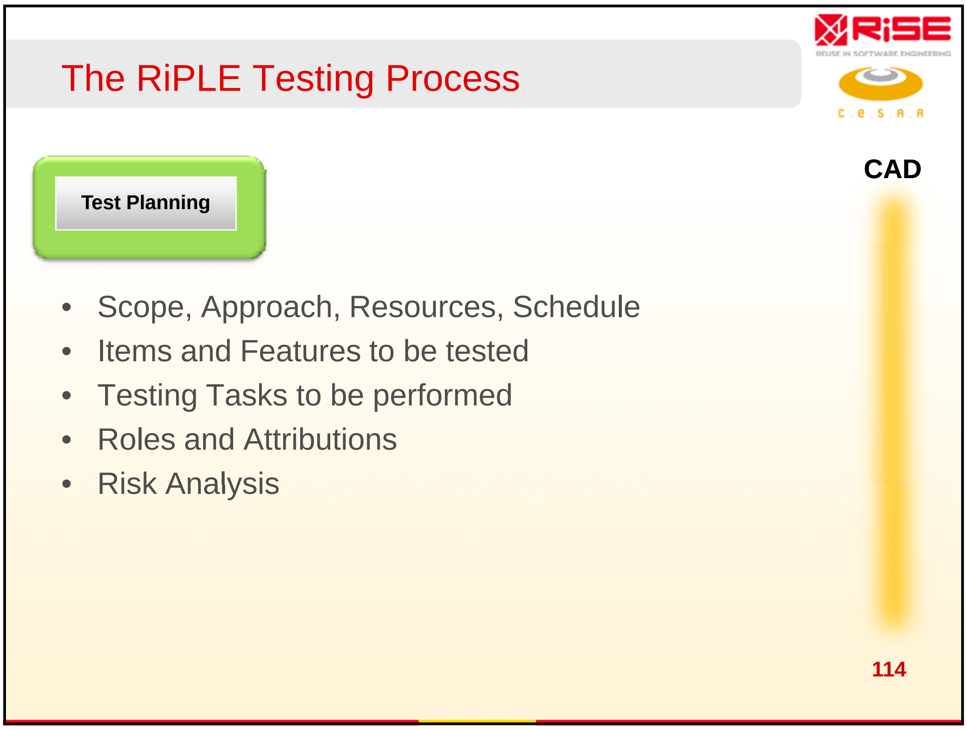
System Testing

Acceptance Testing

CAD

PD

113



The RiPLE Testing Process

Test Planning

- Scope, Approach, Resources, Schedule
- Items and Features to be tested
- Testing Tasks to be performed
- Roles and Attributions
- Risk Analysis

CAD

114

The RiPLE Testing Process

Static Analysis

CAD

- Validate:
  - Feature Model
  - Use Cases and Requirements
  - Design
  - Feature Dependency
- Checklists and Validation Meeting
- We are not covering "Inspection"

The RiPLE Testing Process

Unit Testing → **Integration Testing (CAD)**

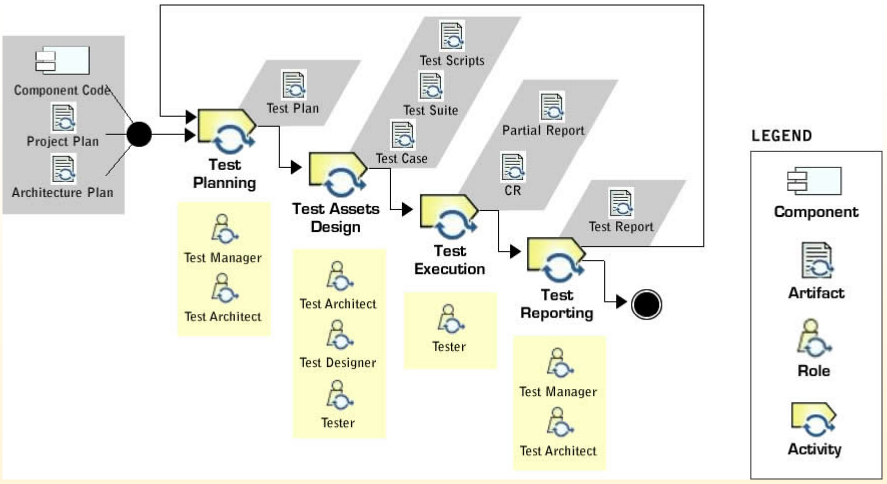
CAD

- Activities
  - Plan Tests
  - Create Test Assets
  - Execute Tests
  - Report Tests
- Regression Testing when necessary
- A Component is considered a Unit in this approach

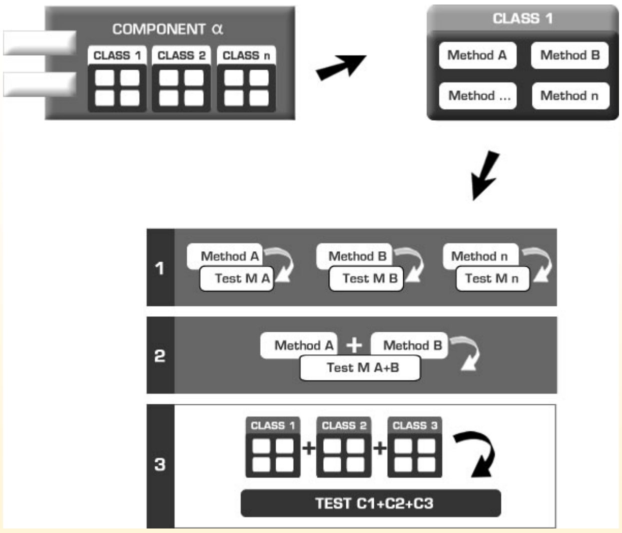
RiPLE-TE : Unit Testing

- Objective:
  - Exercises a unit (component) of code
    - Classes and methods integration
- Source information
  - Component Code
  - Component Specification

RiPLE-TE : Unit Testing



RiPLE-TE : Unit Testing



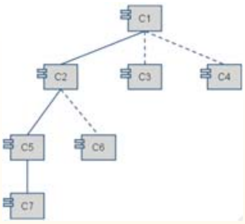
RiPLE-TE : Integration Testing (CAD)

- Objective:
  - Reference Architecture Conformance (Code x Specification)
  - Module Testing
  - Components integration testing

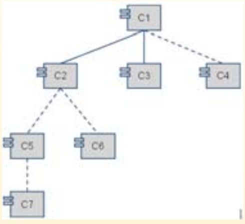
- Source information
  - Architecture Specification (e.g.: Behavioral and Structural views)
  - Feature Model
  - Feature Dependency
  - Code

RiPLE-TE : Integration Testing (CAD)

- Integration Testing Strategies
  - Non-Incremental (Big-Bang)
  - Incremental
    - Bottom-up
    - Top-Down
      - Depth-First
      - Breadth-First

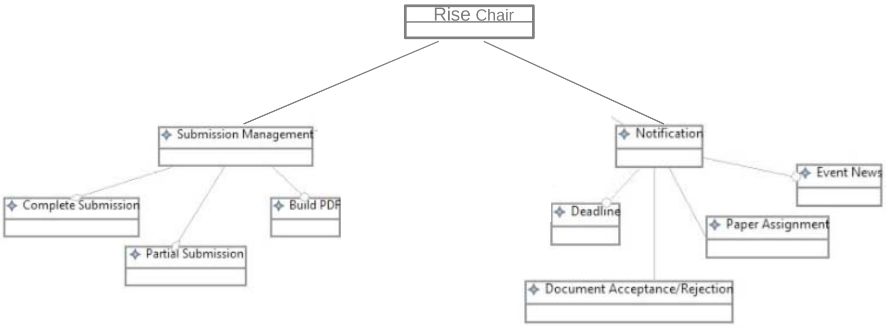


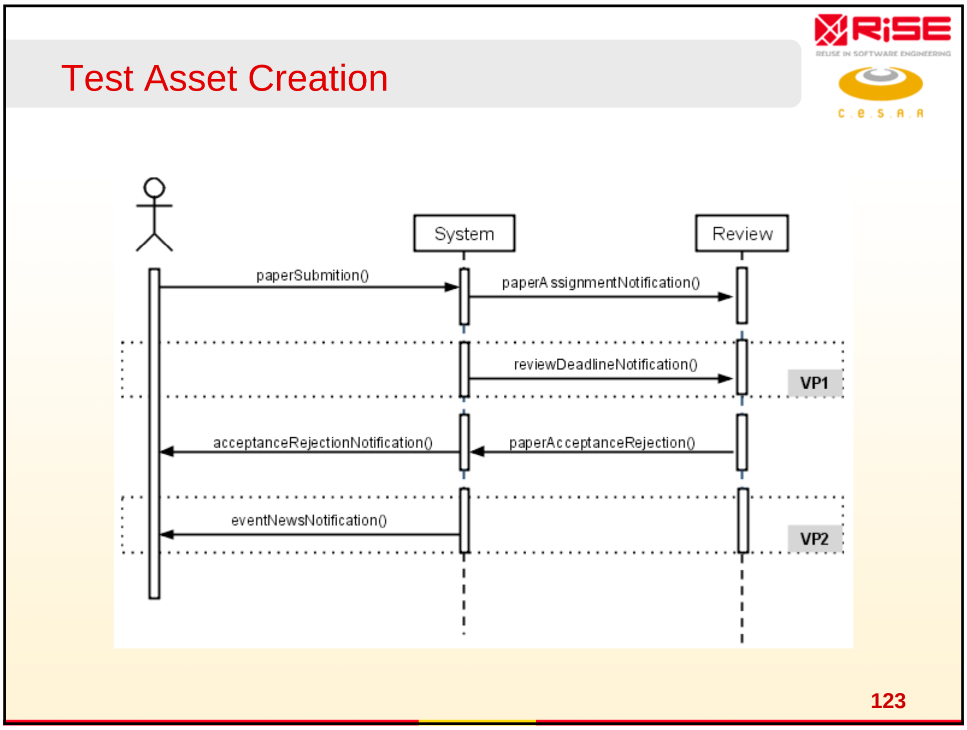
Depth-First



Breadth-First

Feature Model





Test Asset Creation

System

Review

paperSubmition()

paperA ssignmentNotification()

reviewDeadlineNotification()

VP1

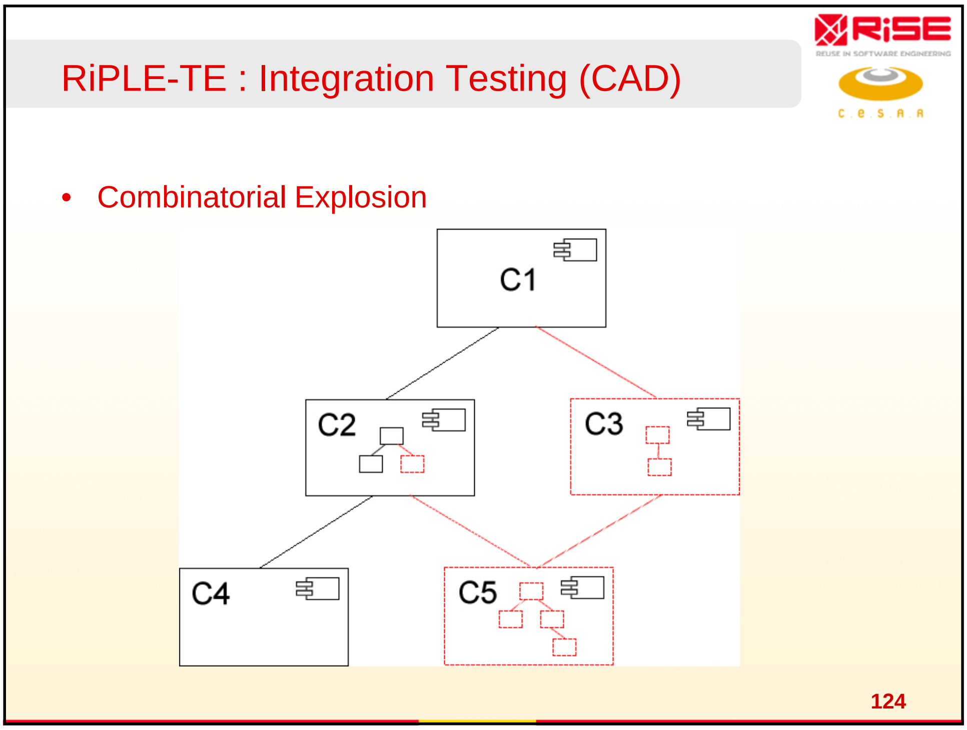
acceptanceRejectionNotification()

paperAcceptanceRejection()

eventNewsNotification()

VP2

123



RiPLE-TE : Integration Testing (CAD)

- Combinatorial Explosion

C1

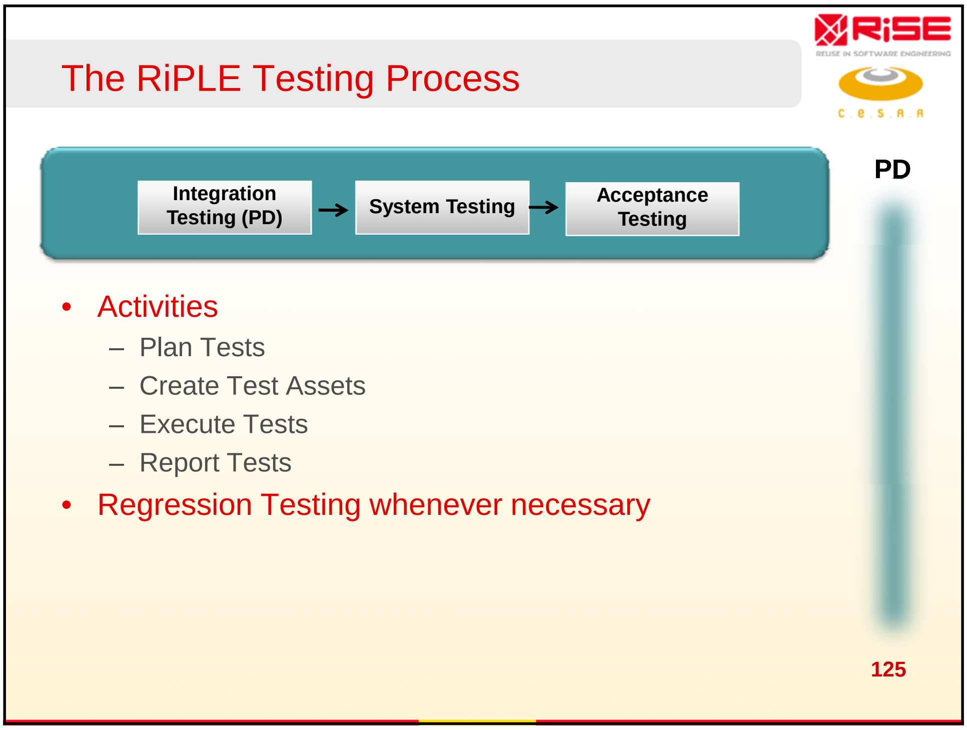
C2

C3

C4

C5

124

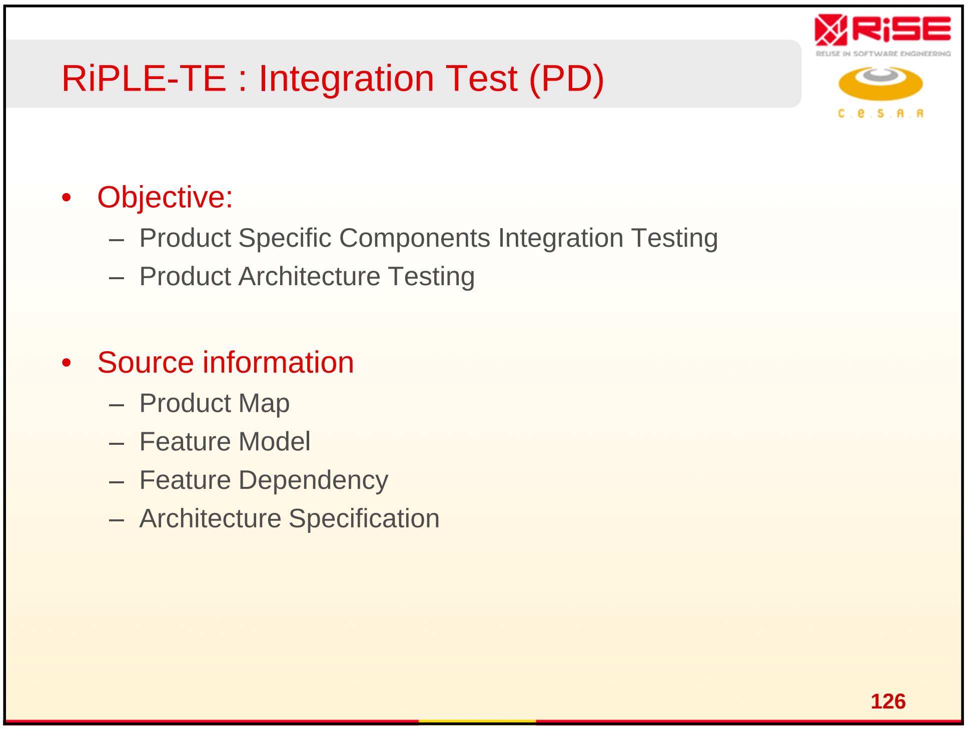


The RiPLE Testing Process

Integration Testing (PD) → System Testing → Acceptance Testing

PD

- Activities
  - Plan Tests
  - Create Test Assets
  - Execute Tests
  - Report Tests
- Regression Testing whenever necessary

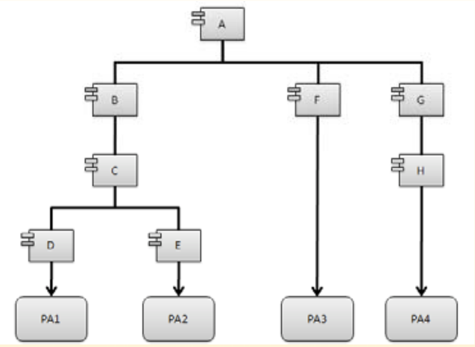


RiPLE-TE : Integration Test (PD)

- Objective:
  - Product Specific Components Integration Testing
  - Product Architecture Testing
- Source information
  - Product Map
  - Feature Model
  - Feature Dependency
  - Architecture Specification

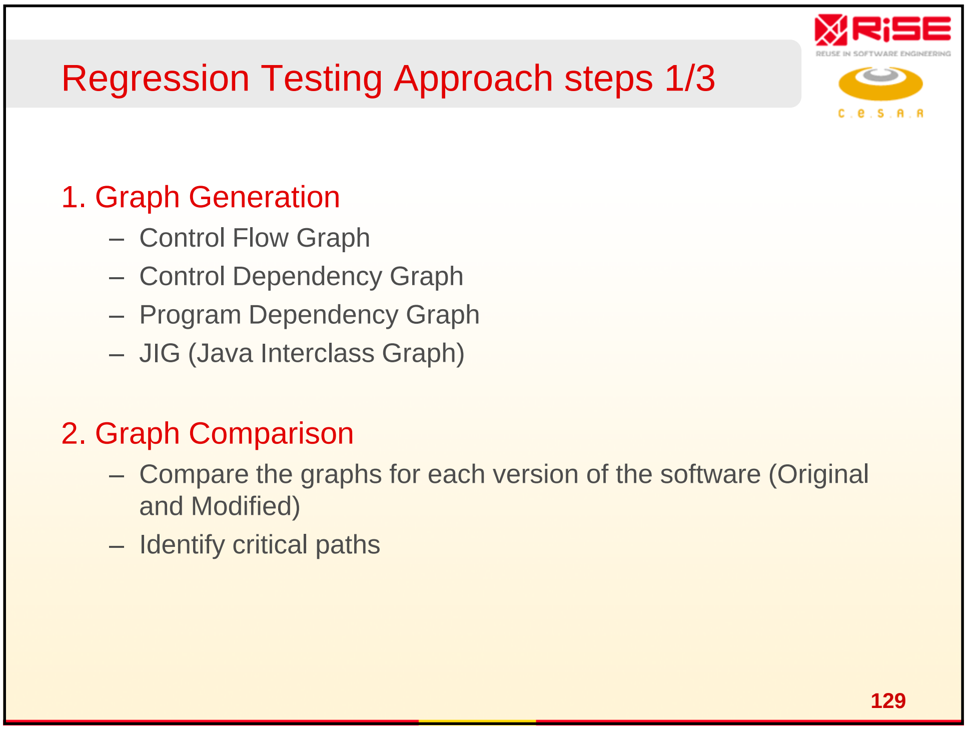
Architecture Regression Testing

- Objective
  - Test the reference architecture after modification or evolution
  - Conformance between product and reference architectures
  - Test product specific architecture



Types Of Regression

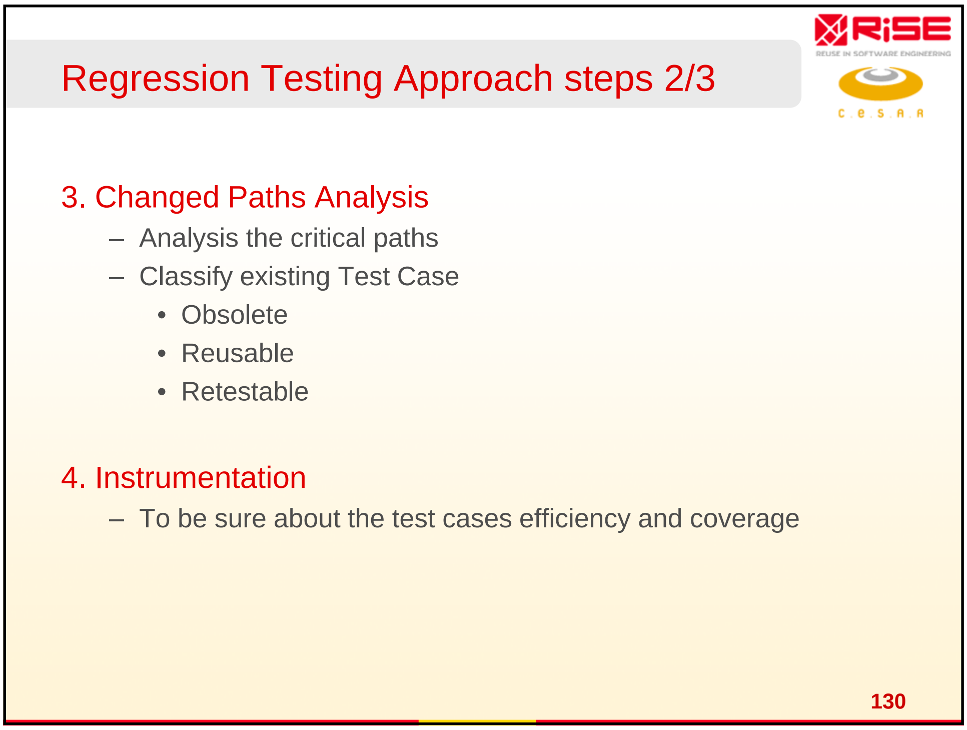
Corrective Regression	Progressive Regression
• Specification is not changed	• Specification is changed
• Involves minor modification to code	• Involves major modification
• Many test cases can be reused	• Fewer test cases can be reused
• Invoked at irregular intervals	• Invoked at regular intervals



Regression Testing Approach steps 1/3

1. Graph Generation
   - Control Flow Graph
   - Control Dependency Graph
   - Program Dependency Graph
   - JIG (Java Interclass Graph)

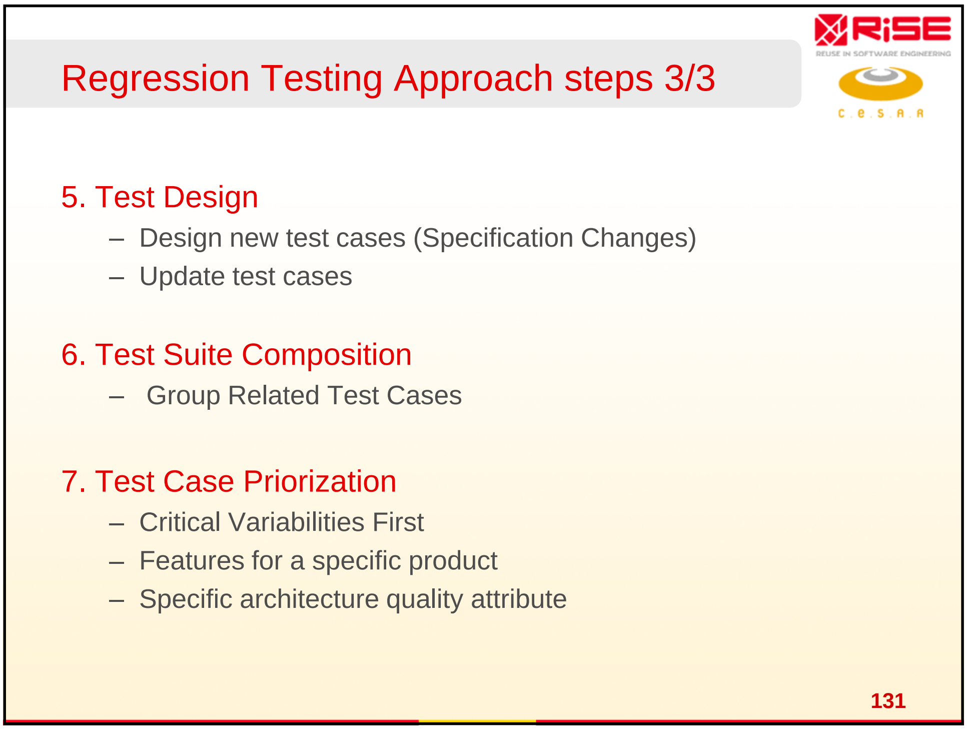
2. Graph Comparison
   - Compare the graphs for each version of the software (Original and Modified)
   - Identify critical paths



Regression Testing Approach steps 2/3

3. Changed Paths Analysis
   - Analysis the critical paths
   - Classify existing Test Case
     - Obsolete
     - Reusable
     - Retestable

4. Instrumentation
   - To be sure about the test cases efficiency and coverage

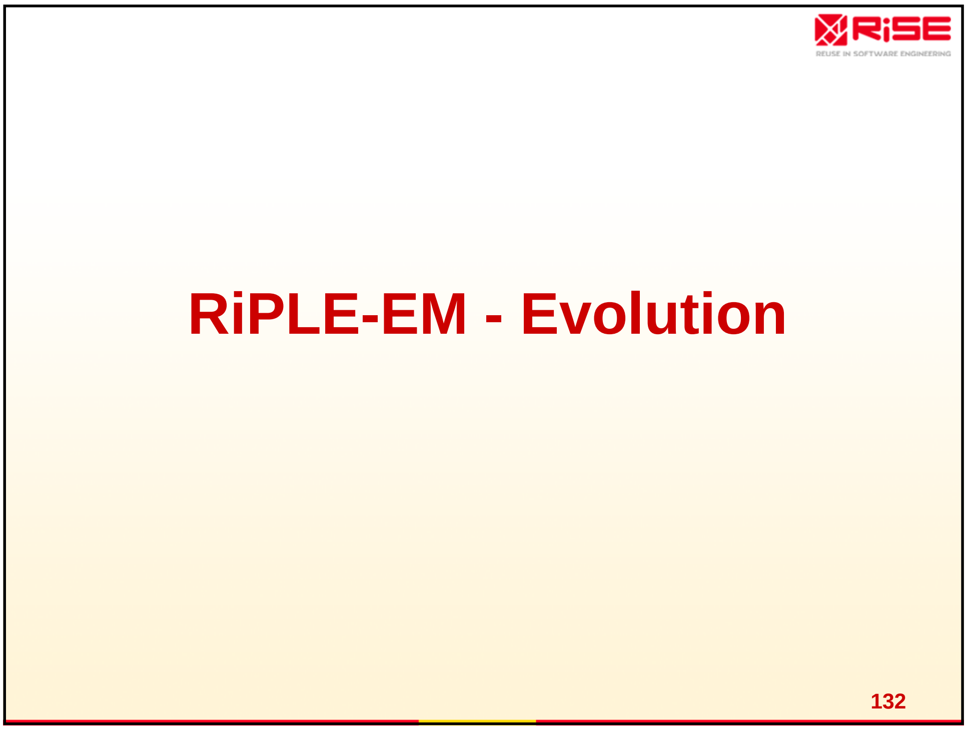


Regression Testing Approach steps 3/3

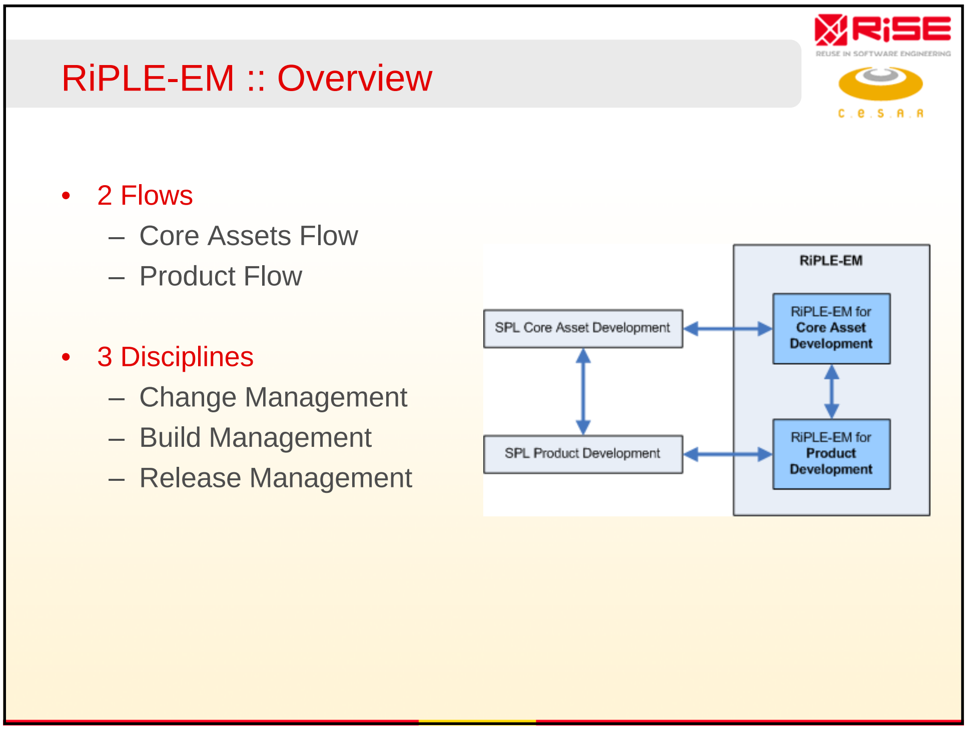
5. Test Design
   - Design new test cases (Specification Changes)
   - Update test cases

6. Test Suite Composition
   - Group Related Test Cases

7. Test Case Priorization
   - Critical Variabilities First
   - Features for a specific product
   - Specific architecture quality attribute



RiPLE-EM - Evolution



RiPLE-EM :: Overview

- 2 Flows
  - Core Assets Flow
  - Product Flow
- 3 Disciplines
  - Change Management
  - Build Management
  - Release Management

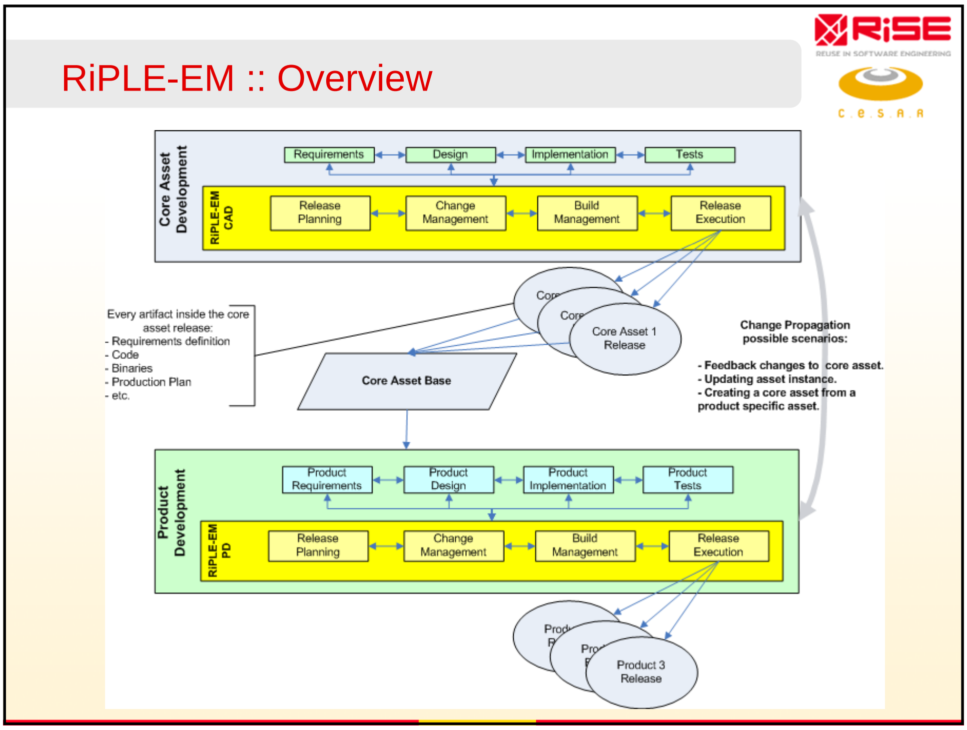
RiPLE-EM

SPL Core Asset Development

RiPLE-EM for **Core Asset Development**

SPL Product Development

RiPLE-EM for **Product Development**



RiPLE-EM :: Overview

Core Asset Development

Requirements

Design

Implementation

Tests

RiPLE-EM CAD

Release Planning

Change Management

Build Management

Release Execution

Core

Core

Core Asset 1 Release

Every artifact inside the core asset release:
- Requirements definition
- Code
- Binaries
- Production Plan
- etc.

Core Asset Base

Change Propagation possible scenarios:

- **Feedback changes to core asset.**
- **Updating asset instance.**
- **Creating a core asset from a product specific asset.**

Product Development

Product Requirements

Product Design

Product Implementation

Product Tests

RiPLE-EM PD

Release Planning

Change Management

Build Management

Release Execution

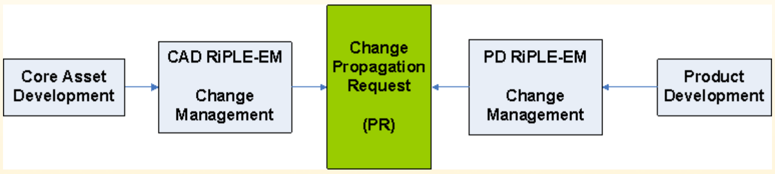
Prod R

Pro F

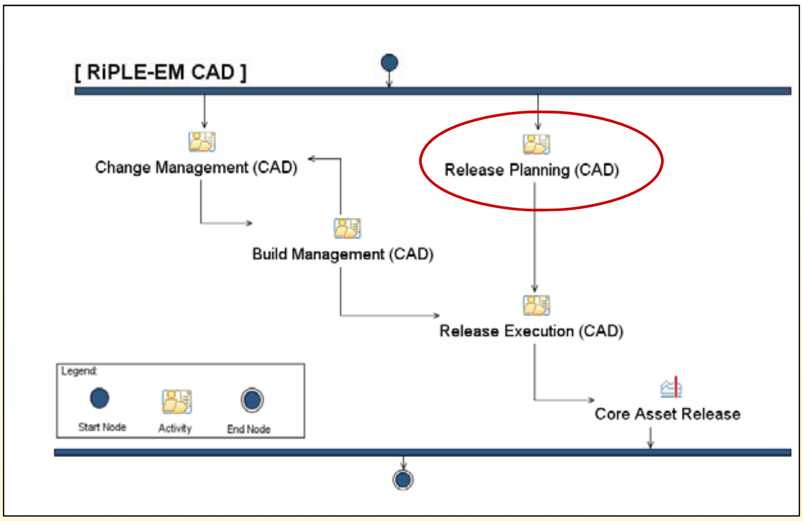
Product 3 Release

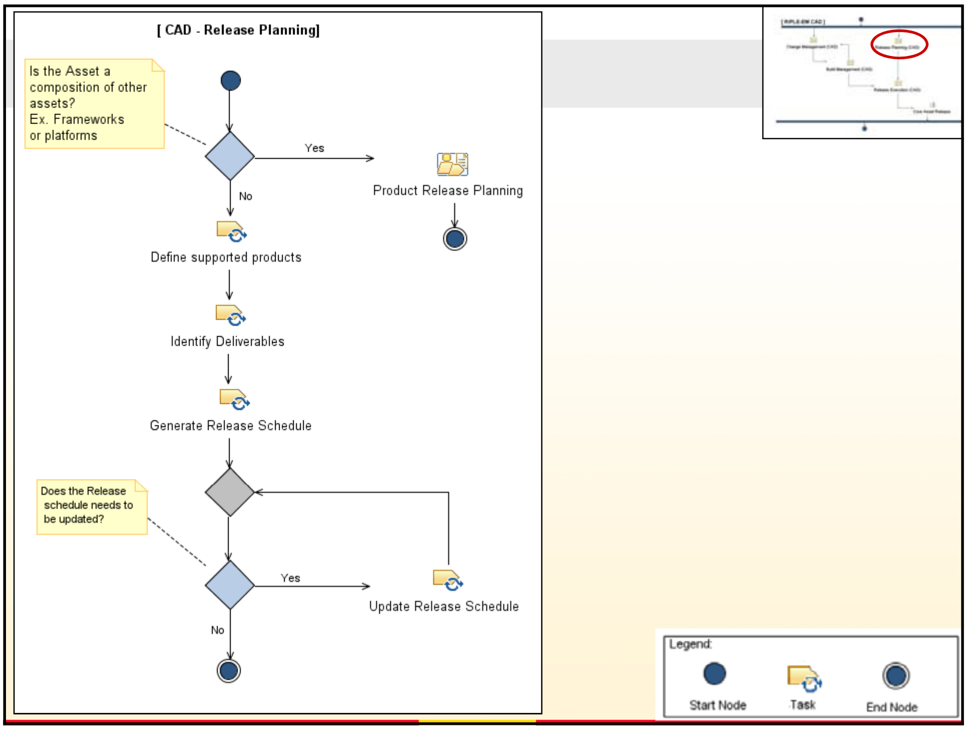
RiPLE-EM :: CADxPD Communication

- The Propagation Request (PR) is a way to propagate the evolution (changes made to an asset or product) of a certain asset or product to another asset or product.
- It is managed by the change control tool used.



RiPLE-EM CAD :: Macro Flow





[CAD - Release Planning]

Is the Asset a composition of other assets? Ex. Frameworks or platforms

Yes

Product Release Planning

No

Define supported products

Identify Deliverables

Generate Release Schedule

Does the Release schedule needs to be updated?

Yes

Update Release Schedule

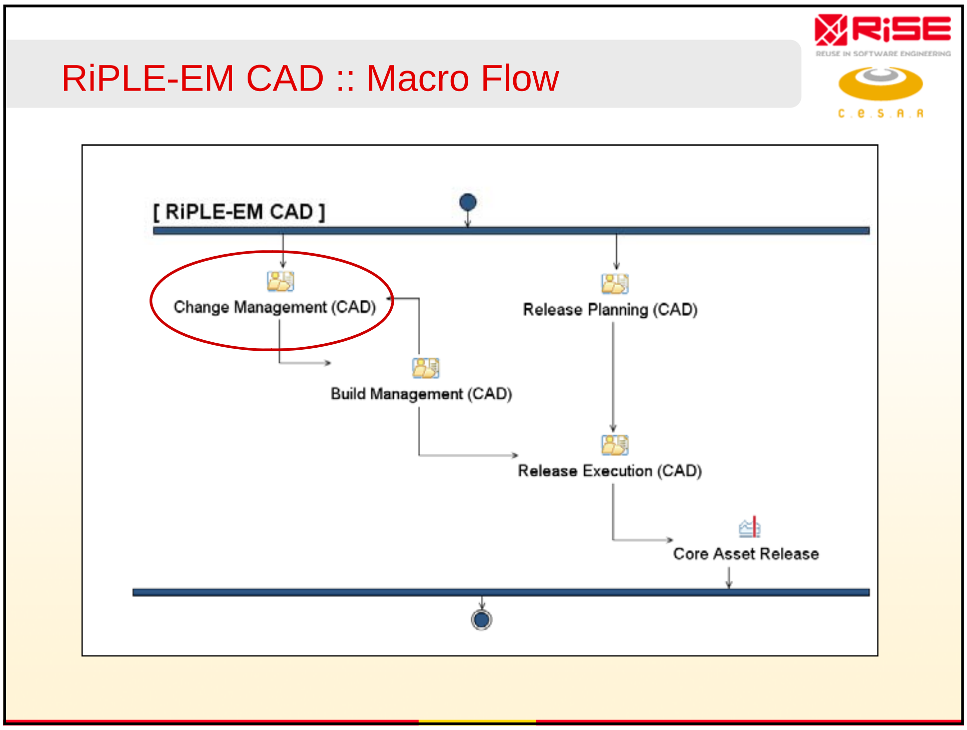
No

Legend:

● Start Node

Task

● End Node



RiPLE-EM CAD :: Macro Flow

[RiPLE-EM CAD]

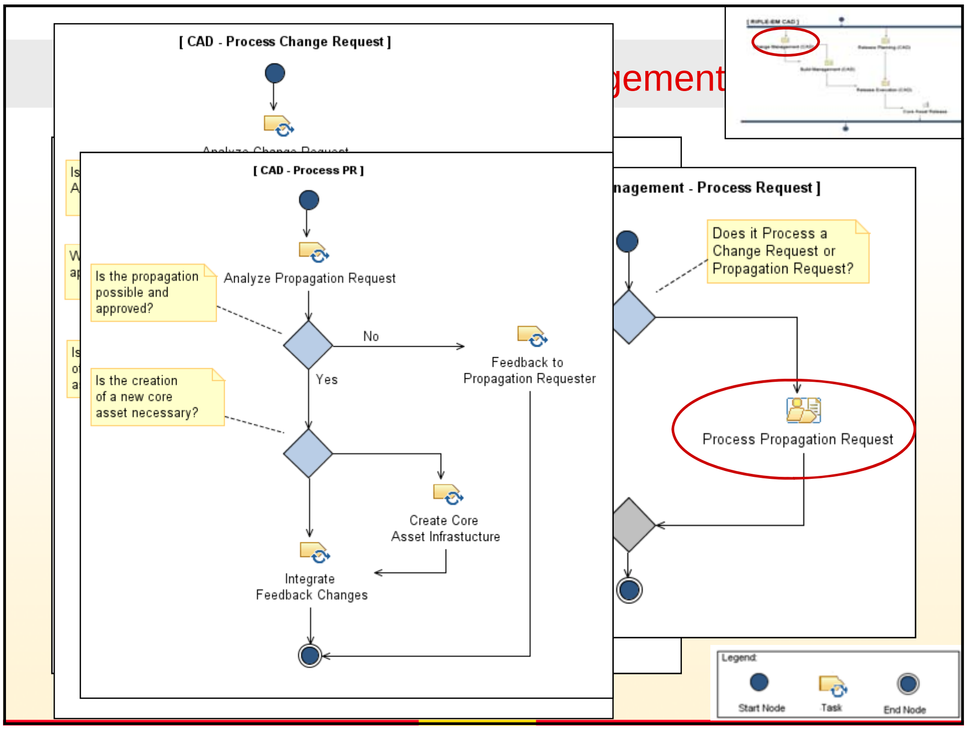
Change Management (CAD)

Release Planning (CAD)

Build Management (CAD)

Release Execution (CAD)

Core Asset Release



gement

[CAD - Process Change Request]

Analyze Change Request

[CAD - Process PR]

Analyze Propagation Request

Is the propagation possible and approved?

No

Yes

Feedback to Propagation Requester

Is the creation of a new core asset necessary?

Create Core Asset Infrastucture

Integrate Feedback Changes

nagement - Process Request]

Does it Process a Change Request or Propagation Request?

Process Propagation Request

Legend

Start Node

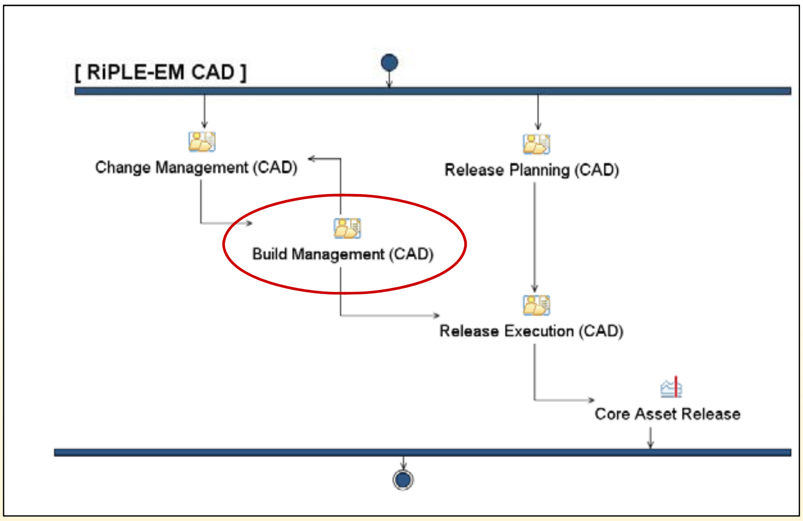
Task

End Node

RiPLE-EM CAD :: Macro Flow

RiSE REUSE IN SOFTWARE ENGINEERING

C.E.S.A.R



[RiPLE-EM CAD]

Change Management (CAD)

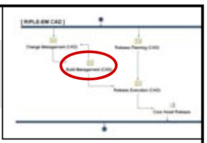
Release Planning (CAD)

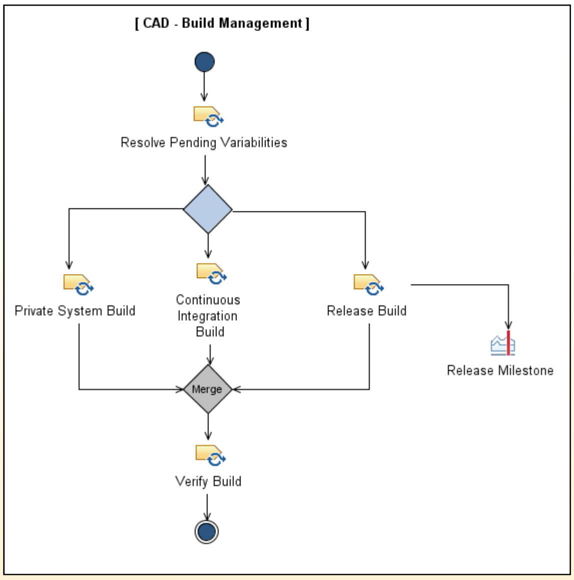
Build Management (CAD)

Release Execution (CAD)

Core Asset Release

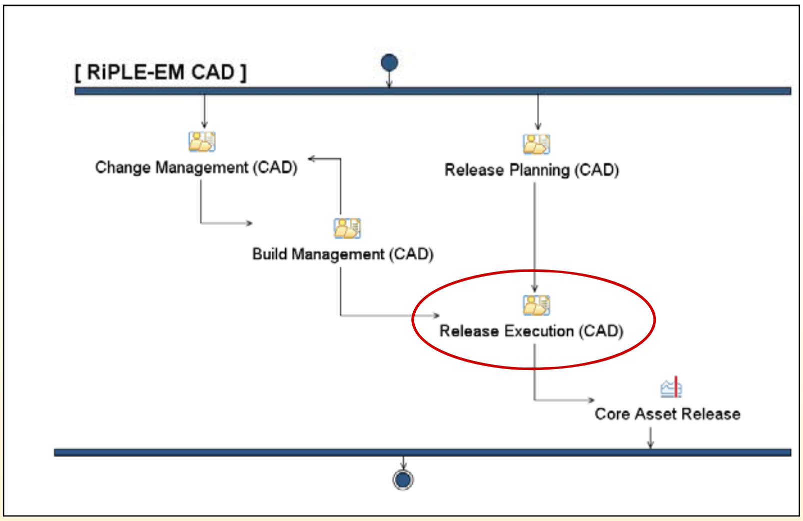
RiPLE-EM CAD :: Build Management



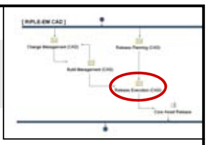

[CAD - Build Management]
Resolve Pending Variabilities
Private System Build
Continuous Integration Build
Release Build
Release Milestone
Merge
Verify Build

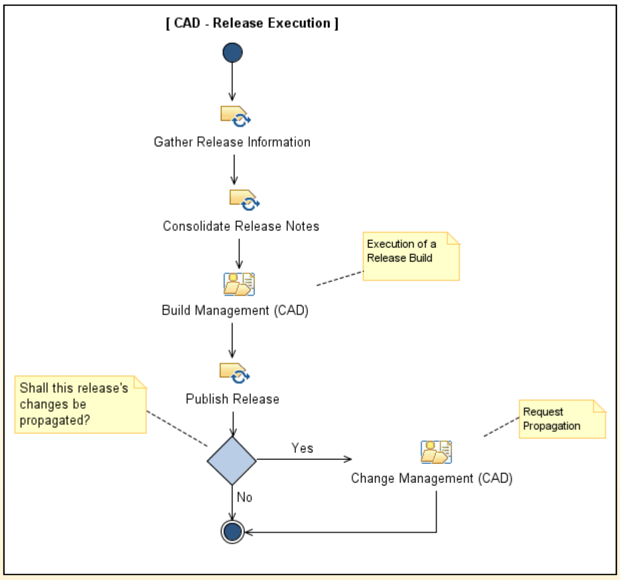
Legend:
● Start Node
Task
● End Node

RiPLE-EM CAD :: Macro Flow

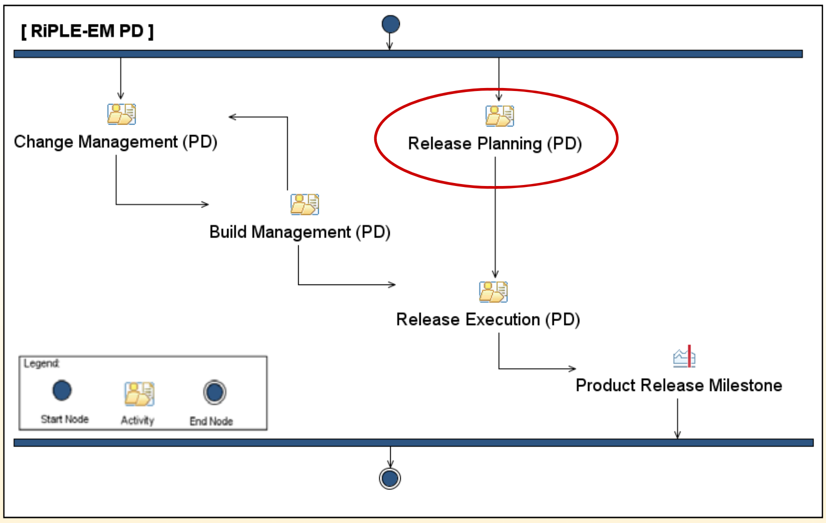


RiPLE-EM CAD :: Release Execution



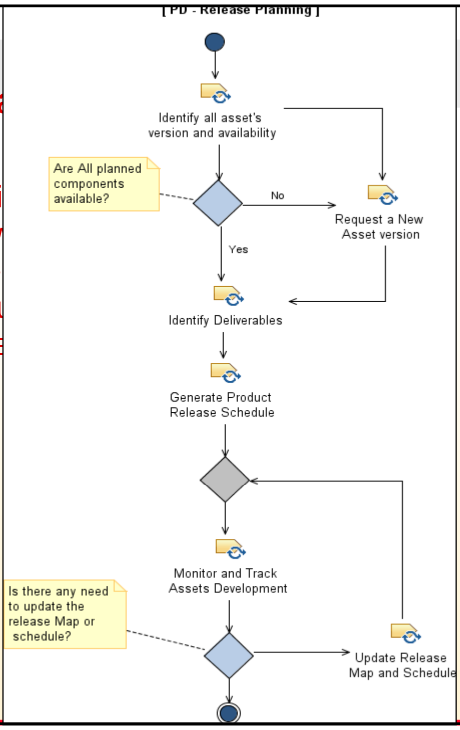

[CAD - Release Execution]
Gather Release Information
Consolidate Release Notes
Execution of a Release Build
Build Management (CAD)
Publish Release
Shall this release's changes be propagated?
Yes
No
Change Management (CAD)
Request Propagation

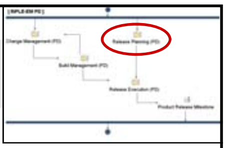
RiPLE-EM PD :: Macro Flow



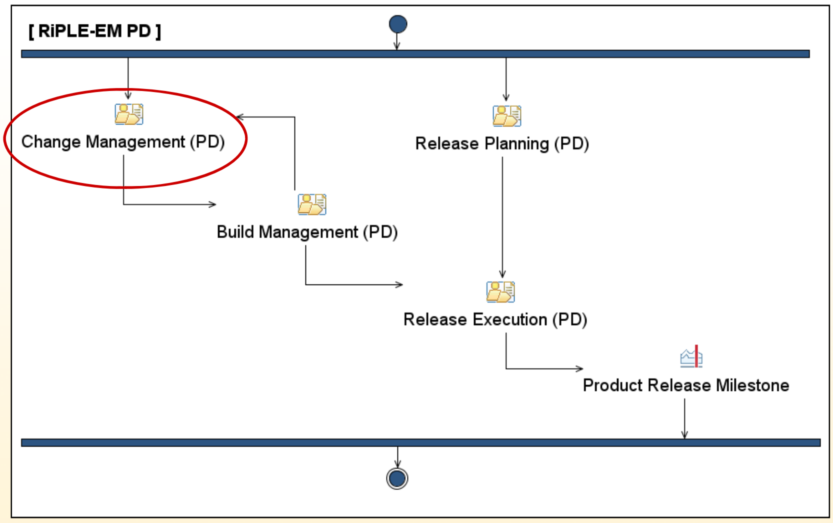
RiPLE-EM Release Pl

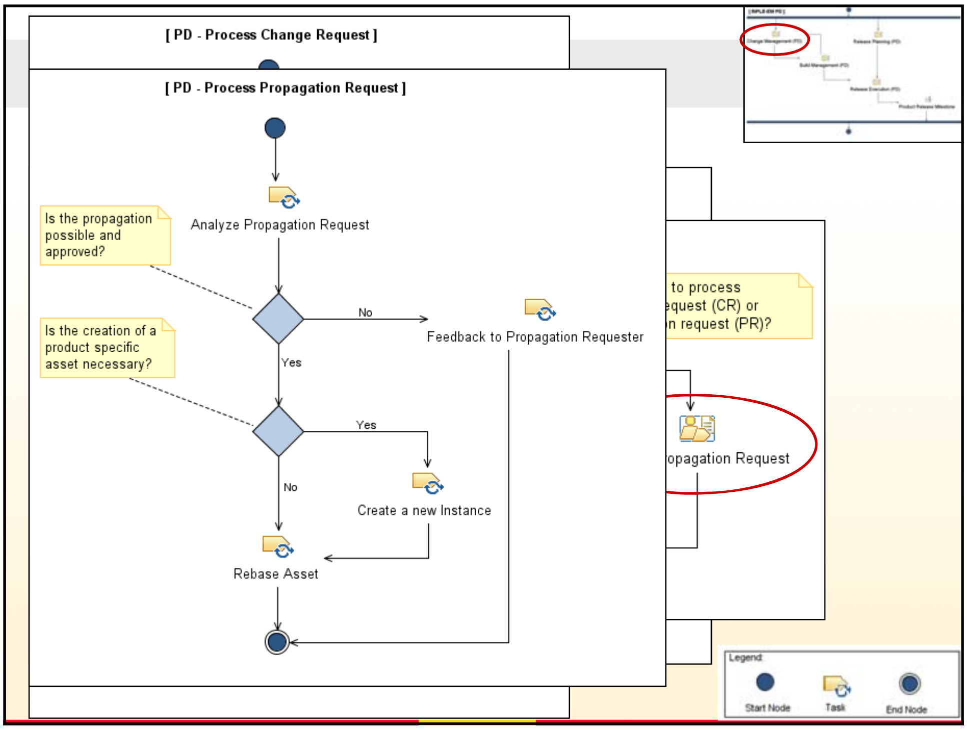
- This activiti in parallel w developme be refined u release exe


[PD - Release Planning]
Identify all asset's version and availability
Are All planned components available?
No
Yes
Request a New Asset version
Identify Deliverables
Generate Product Release Schedule
Monitor and Track Assets Development
Is there any need to update the release Map or schedule?
Update Release Map and Schedule

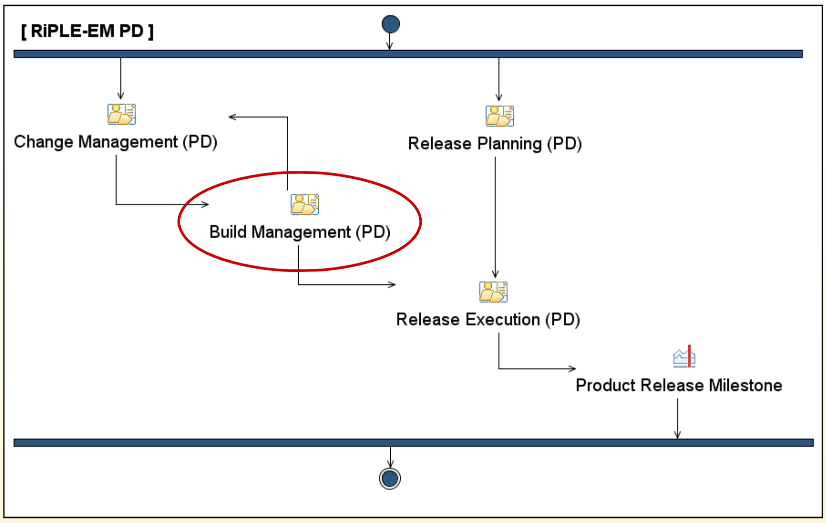


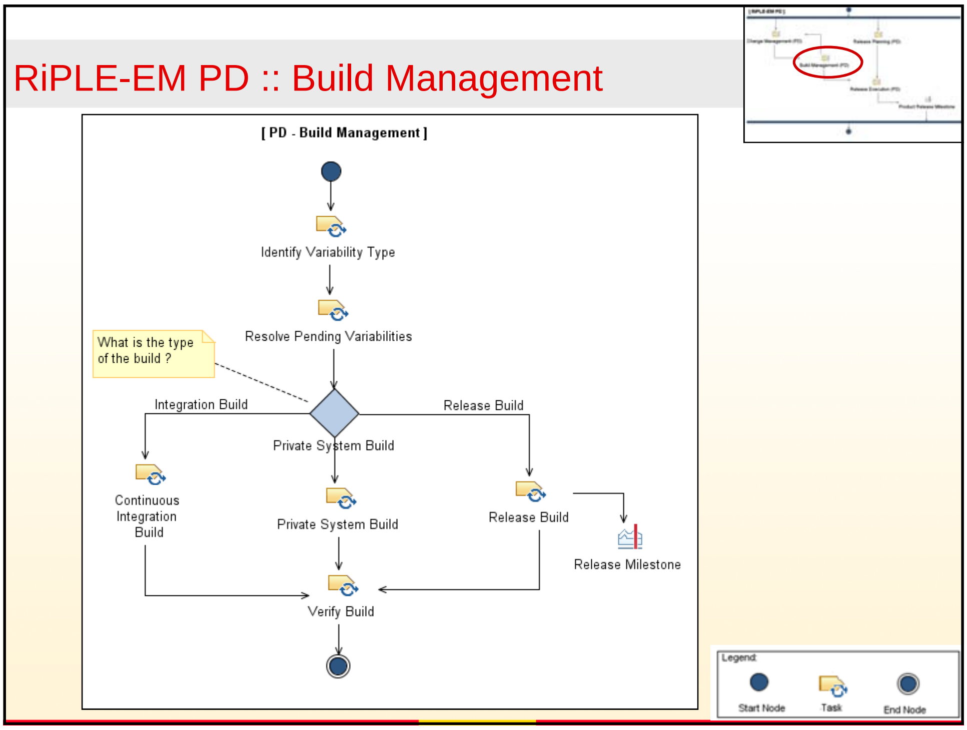
RiPLE-EM PD :: Macro Flow




[PD - Process Change Request]
[PD - Process Propagation Request]
Analyze Propagation Request
Is the propagation possible and approved?
No
Feedback to Propagation Requester
Yes
Is the creation of a product specific asset necessary?
Yes
Create a new Instance
No
Rebase Asset
to process
equest (CR) or
n request (PR)?
opagation Request
Legend:
Start Node
Task
End Node

RiPLE-EM PD :: Macro Flow





RiPLE-EM PD :: Build Management

[PD - Build Management]

Identify Variability Type

Resolve Pending Variabilities

What is the type of the build ?

Integration Build

Release Build

Private System Build

Continuous Integration Build

Private System Build

Release Build

Release Milestone

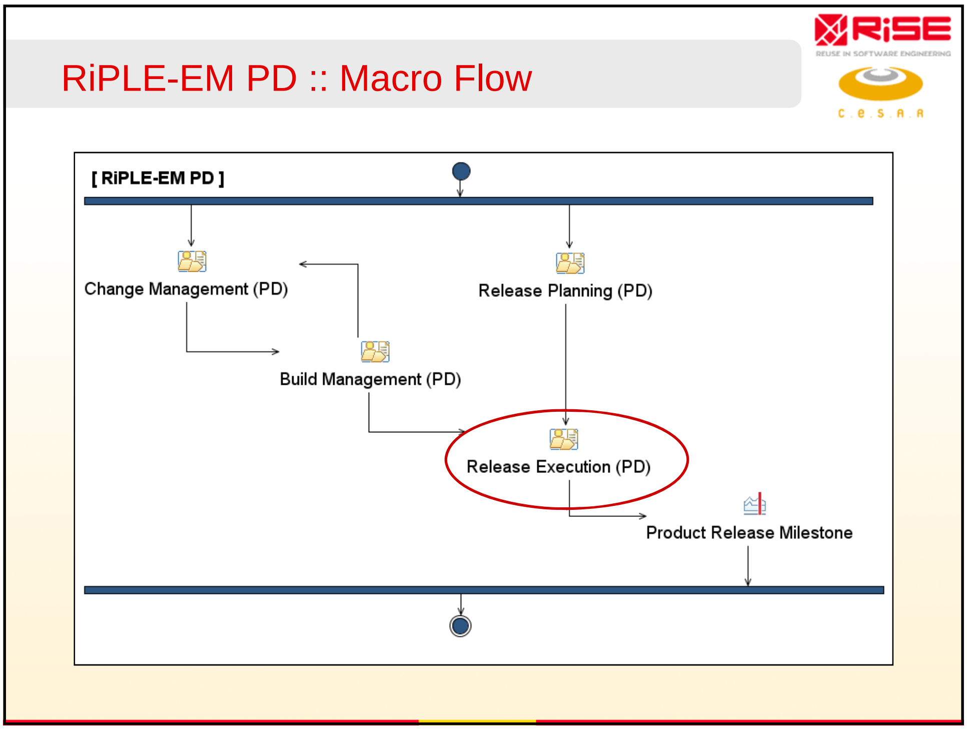
Verify Build

Legend

Start Node

Task

End Node



RiPLE-EM PD :: Macro Flow

[RiPLE-EM PD]

Change Management (PD)

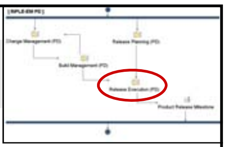
Release Planning (PD)

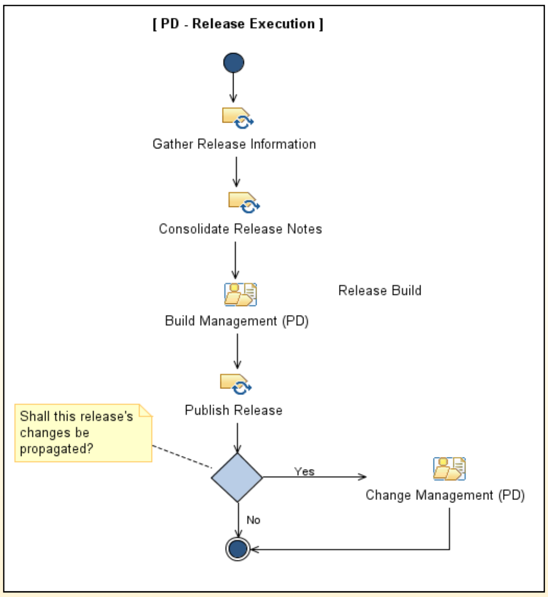
Build Management (PD)

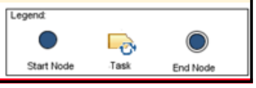
Release Execution (PD)

Product Release Milestone

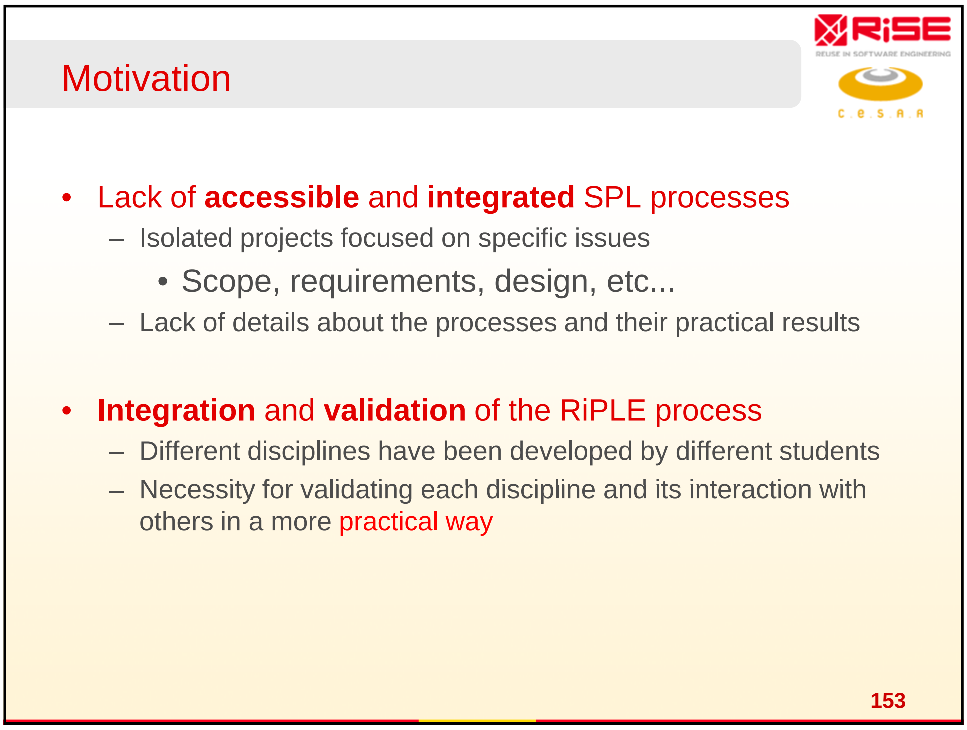
RiPLE-EM PD :: Release Execution




[PD - Release Execution]
Gather Release Information
Consolidate Release Notes
Build Management (PD)
Release Build
Publish Release
Shall this release's changes be propagated?
Yes
Change Management (PD)
No



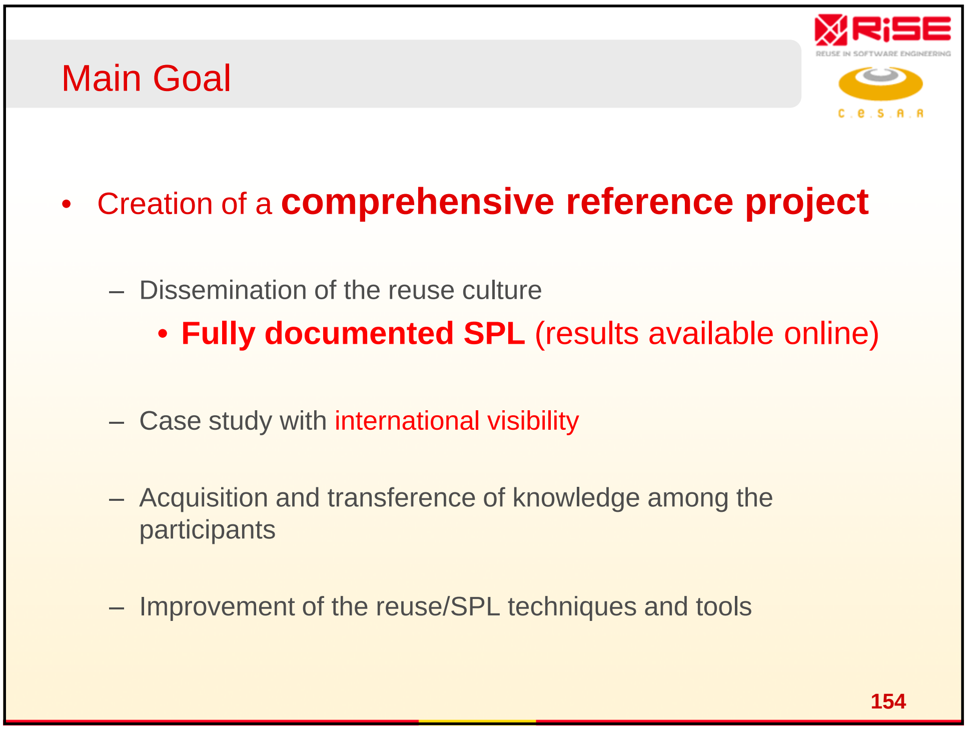
Case Study: Papers Management Software Product Line



Motivation

- Lack of **accessible** and **integrated** SPL processes
  - Isolated projects focused on specific issues
    - Scope, requirements, design, etc...
  - Lack of details about the processes and their practical results

- **Integration** and **validation** of the RiPLE process
  - Different disciplines have been developed by different students
  - Necessity for validating each discipline and its interaction with others in a more practical way



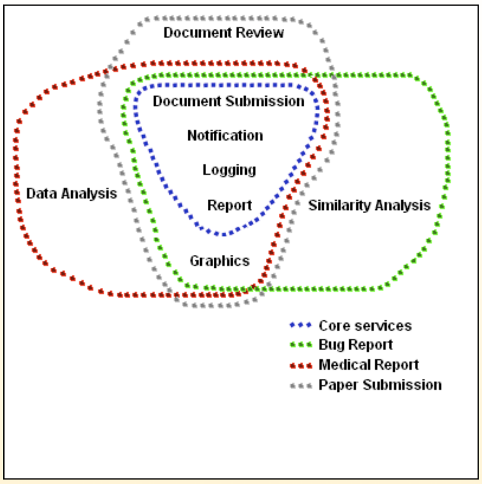
Main Goal

- Creation of a **comprehensive reference project**
  - Dissemination of the reuse culture
    - **Fully documented SPL** (results available online)
  - Case study with international visibility
  - Acquisition and transference of knowledge among the participants
  - Improvement of the reuse/SPL techniques and tools

First step: Domain Selection...

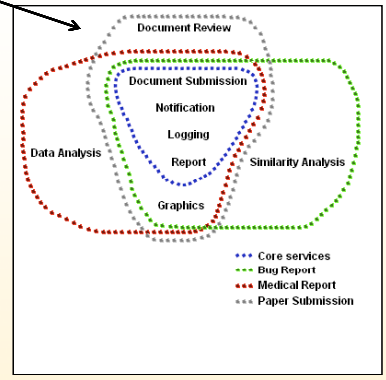
Domain

- Document submission systems
  - Submission
  - Review
  - Notification
  - Logging
  - Report
  - ....



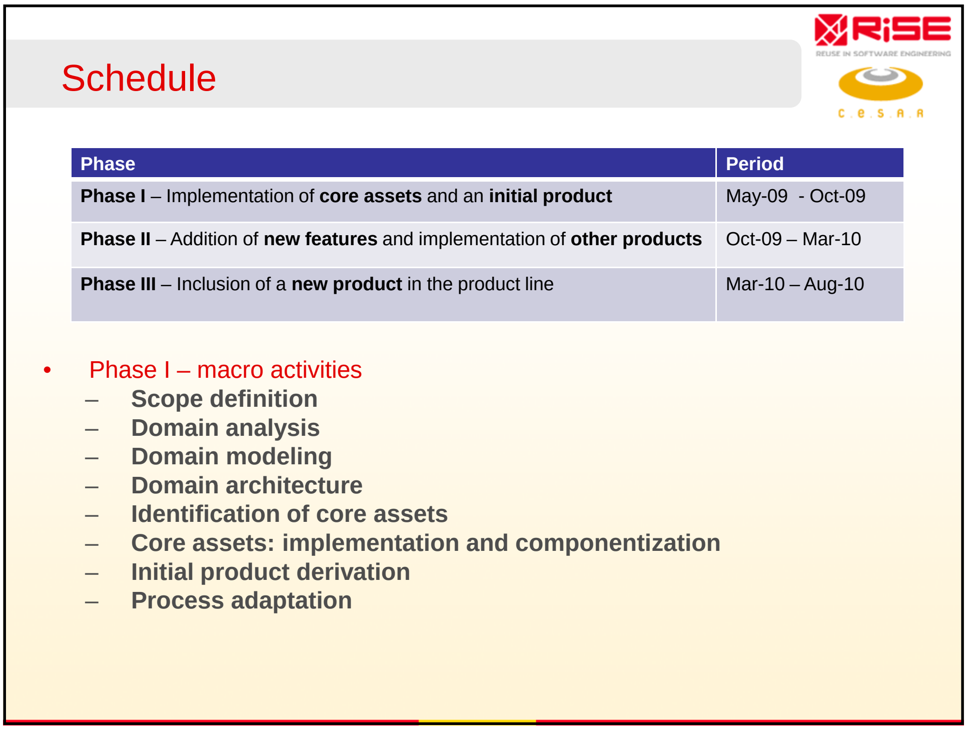
Domain

- **Initial domain: paper submission**
  - **Easier to understand**
  - **Experts accessible**
    - Some rise members developed applications in such domain
  - **Variabilities enough to create a SPL**
    - **Conferences, journals, workshops, etc...**



Involved people

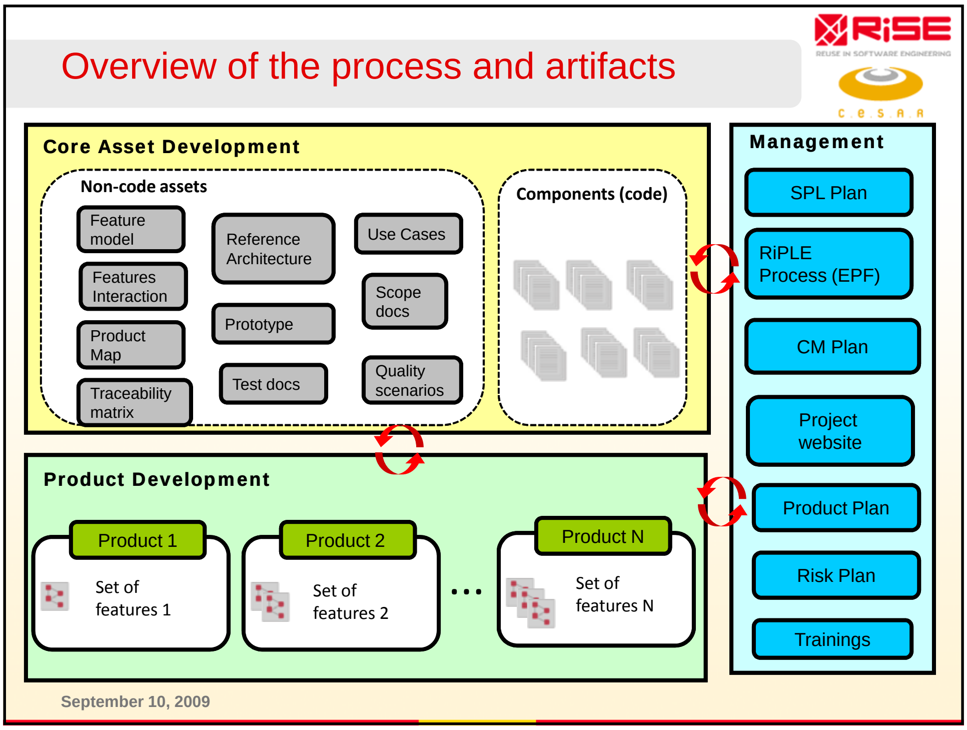
- Team
  - 13 people (reuse specialists)
    - 4 Ph.D students
    - 9 M.Sc. students
- 1 customer and 3 domain experts
  - Professors
- Roles
  - Internal
    - SPL manager; SPL Engineers (SPL Architect, Developer,Testers); Requitements Analyst; Domain Analyst; Scoping Expert; SQA; CCB
  - External
    - Domain Expert; Customer; End Users



Schedule

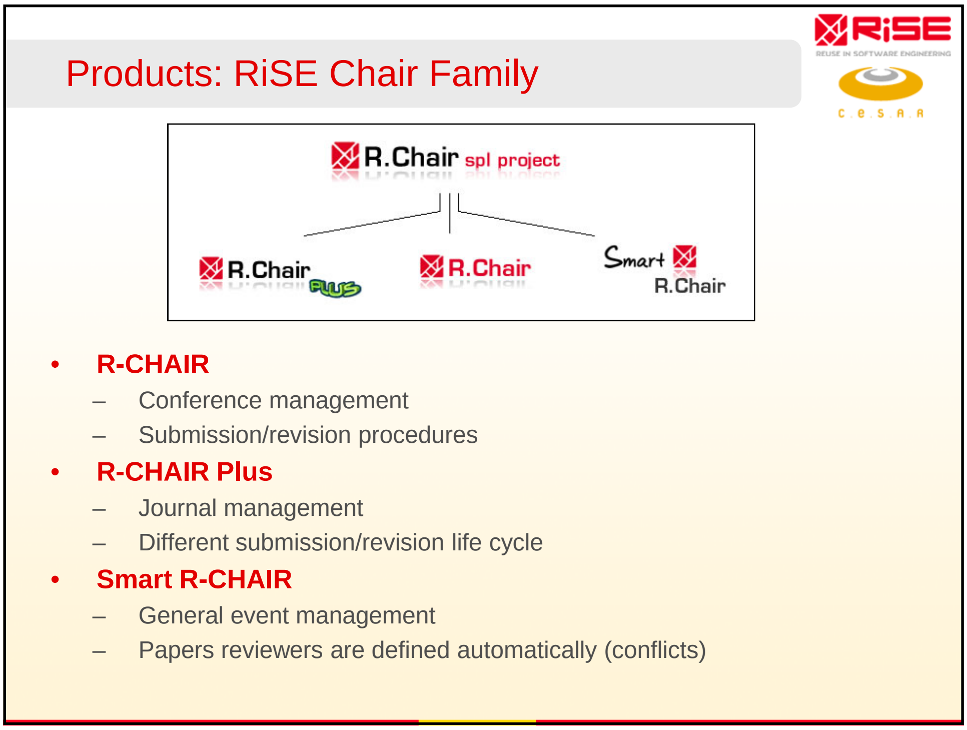
Phase	Period
Phase I – Implementation of **core assets** and an **initial product**	May-09 - Oct-09
Phase II – Addition of **new features** and implementation of **other products**	Oct-09 – Mar-10
Phase III – Inclusion of a **new product** in the product line	Mar-10 – Aug-10

- Phase I – macro activities
  - Scope definition
  - Domain analysis
  - Domain modeling
  - Domain architecture
  - Identification of core assets
  - Core assets: implementation and componentization
  - Initial product derivation
  - Process adaptation



Overview of the process and artifacts

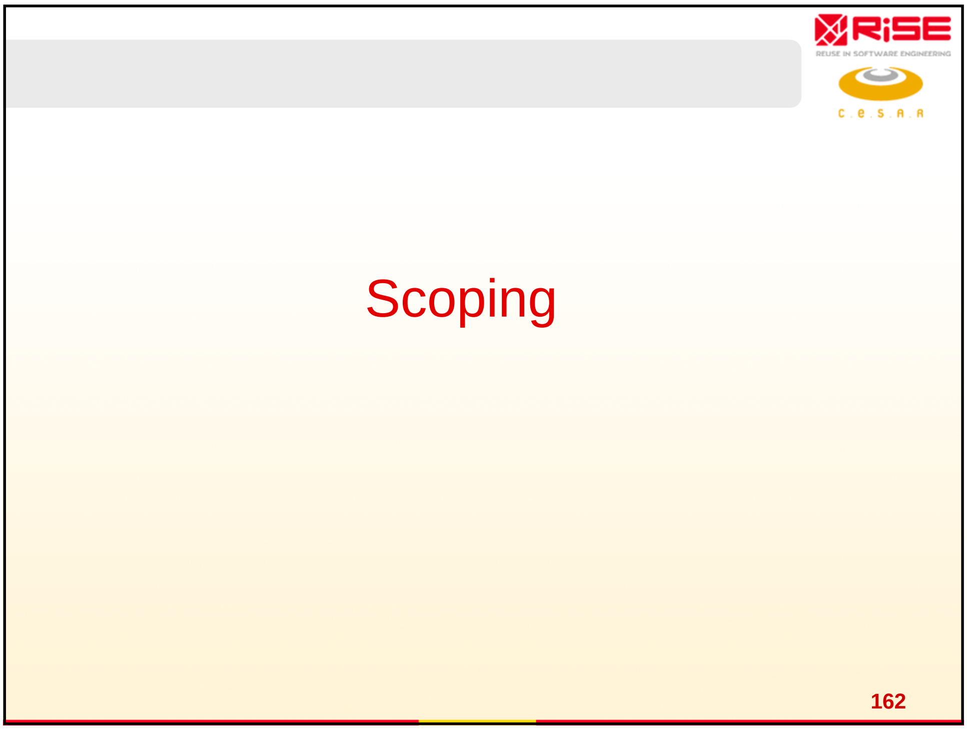
September 10, 2009



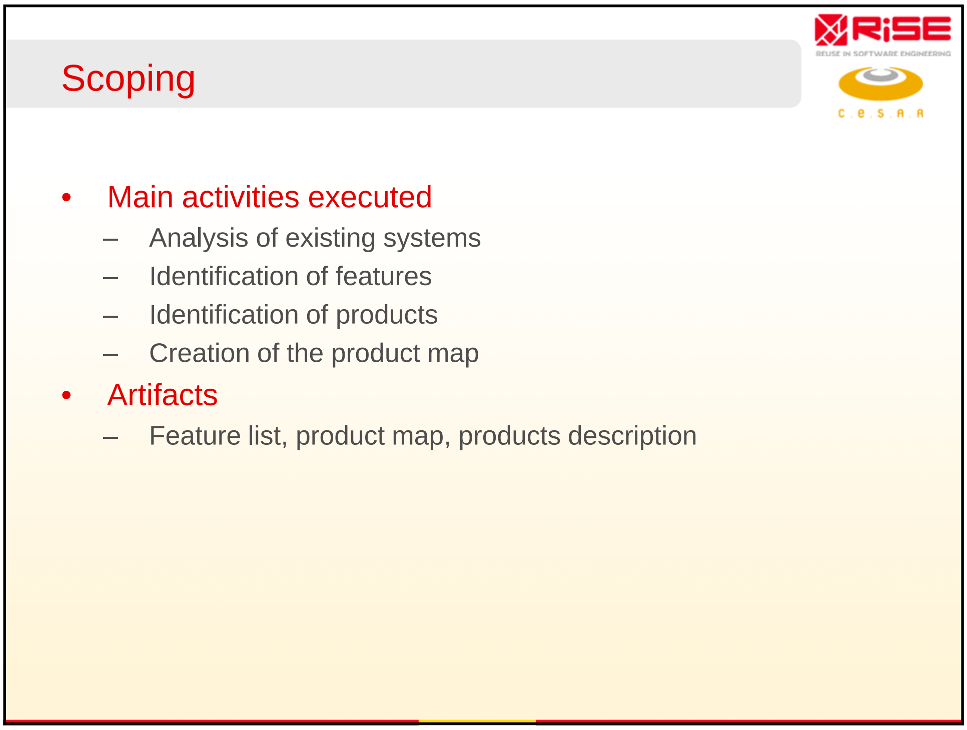
Products: RiSE Chair Family

R.Chair spl project
R.Chair PLUS
R.Chair
Smart R.Chair

- **R-CHAIR**
  - Conference management
  - Submission/revision procedures
- **R-CHAIR Plus**
  - Journal management
  - Different submission/revision life cycle
- **Smart R-CHAIR**
  - General event management
  - Papers reviewers are defined automatically (conflicts)

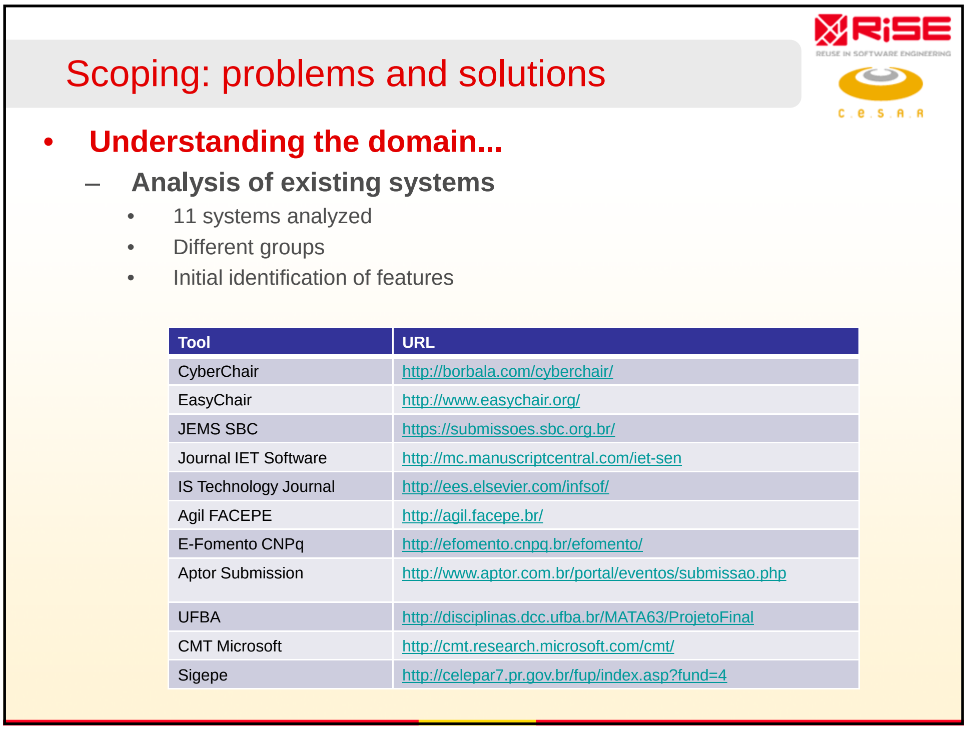


Scoping



Scoping

- Main activities executed
  - Analysis of existing systems
  - Identification of features
  - Identification of products
  - Creation of the product map
- Artifacts
  - Feature list, product map, products description



Scoping: problems and solutions

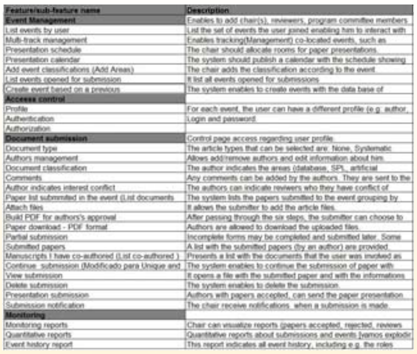
- **Understanding the domain...**
  - **Analysis of existing systems**
    - 11 systems analyzed
    - Different groups
    - Initial identification of features

Tool	URL
CyberChair	http://borbala.com/cyberchair/
EasyChair	http://www.easychair.org/
JEMS SBC	https://submissoes.sbc.org.br/
Journal IET Software	http://mc.manuscriptcentral.com/iet-sen
IS Technology Journal	http://ees.elsevier.com/infsof/
Agil FACEPE	http://agil.facepe.br/
E-Fomento CNPq	http://efomento.cnpq.br/efomento/
Aptor Submission	http://www.aptor.com.br/portal/eventos/submissao.php
UFBA	http://disciplinas.dcc.ufba.br/MATA63/ProjetoFinal
CMT Microsoft	http://cmt.research.microsoft.com/cmt/
Sigepe	http://celepar7.pr.gov.br/fup/index.asp?fund=4

Scoping: problems and solutions

- Consolidation of the features
  - Problems with Features Definition
    - **Granularity level**
  - **Lack of a “deep” knowledge of the domain**
    - the most experienced professionals were included in the group of **domain experts**

Initial list of features (consolidated)

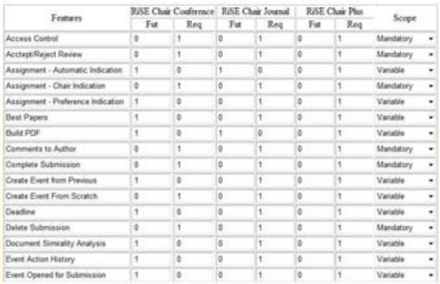


Feature/sub-feature name	Description
Event Management	Enables to add chair(s), reviewers, program committee members
List events by user	List the set of events the user joined enabling him to interact with
Multi-track management	Enables tracking(Management) co-located events, such as
Presentation schedule	The chair should allocate rooms for paper presentations.
Presentation calendar	The system should publish a calendar with the schedule showing
Add event classifications (Add Areas)	The chair adds the classification according to the event
List events opened for submission	It list all events opened for submissions
Create event based on a previous	The system enables to create events with the data base of
Access control	
Profile	For each event, the user can have a different profile (e.g. author,
Authentication	Login and password
Authorization	
Document submission	Control page access regarding user profile
Document type	The article types that can be selected are: None, Systematic
Authors management	Allows add/remove authors and edit information about him.
Document classification	The author indicates the areas (database, SPL, artificial
Comments	Any comments can be added by the authors. They are sent to the
Author indicates interest conflict	The authors can indicate reviewers who they have conflict of
Paper list submitted in the event (List documents	The system lists the papers submitted to the event grouping by
Attach files	It allows the submitter to add the article files.
Build PDF for authors's approval	After passing through the six steps, the submitter can choose to
Paper download - PDF format	Authors are allowed to download the uploaded files.
Partial submission	Incomplete forms may be completed and submitted later. Some
Submitted papers	A list with the submitted papers (by an author) are provided.
Manuscripts I have co-authored (List co-authored)	Presents a list with the documents that the user was involved as
Continue submission (Modificado para Unique and	The system enables to continue the submission of paper with
View submission	It opens a file with the submitted paper and with the informations
Delete submission	The system enables to delete the submission.
Presentation submission	Authors with papers accepted, can send the paper presentation
Submission notification	The chair receive notifications when a submission is made.
Monitoring	
Monitoring reports	Chair can visualize reports (papers accepted, rejected, reviews
Quantitative reports	Quantitative reports about submissions and events [vamos explodir
Event history report	This report indicates all event history, including e.g. the roles

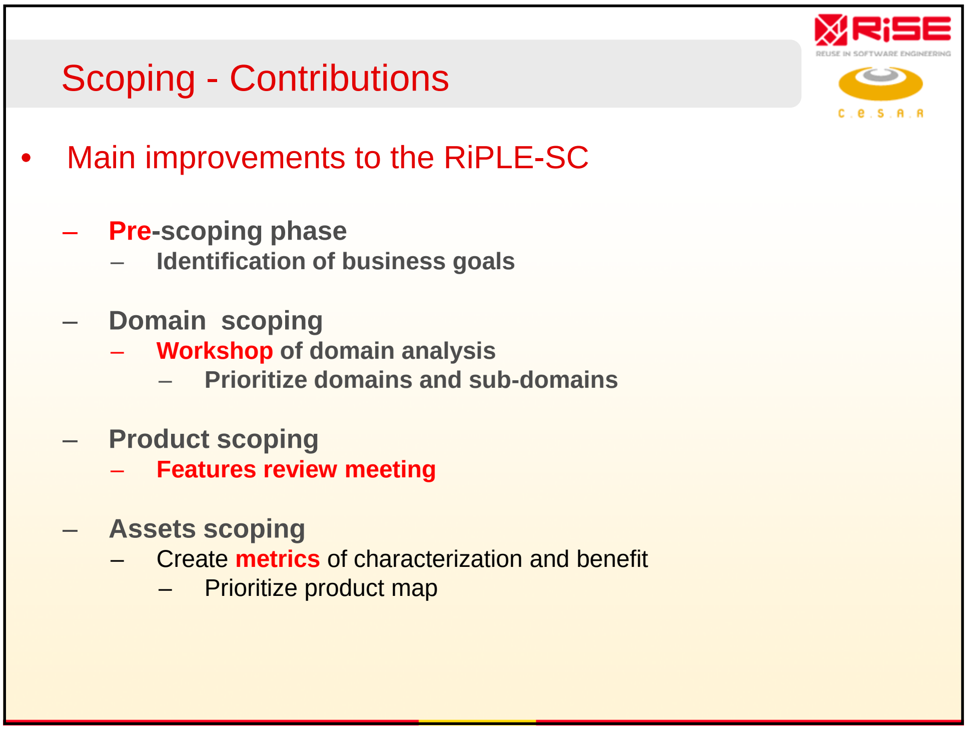
Scoping: problems and solutions

- Identification of products and creation of the product map
  - **Lack of metrics to support the decisions**
    - **What features should be included in the products?**

Product map

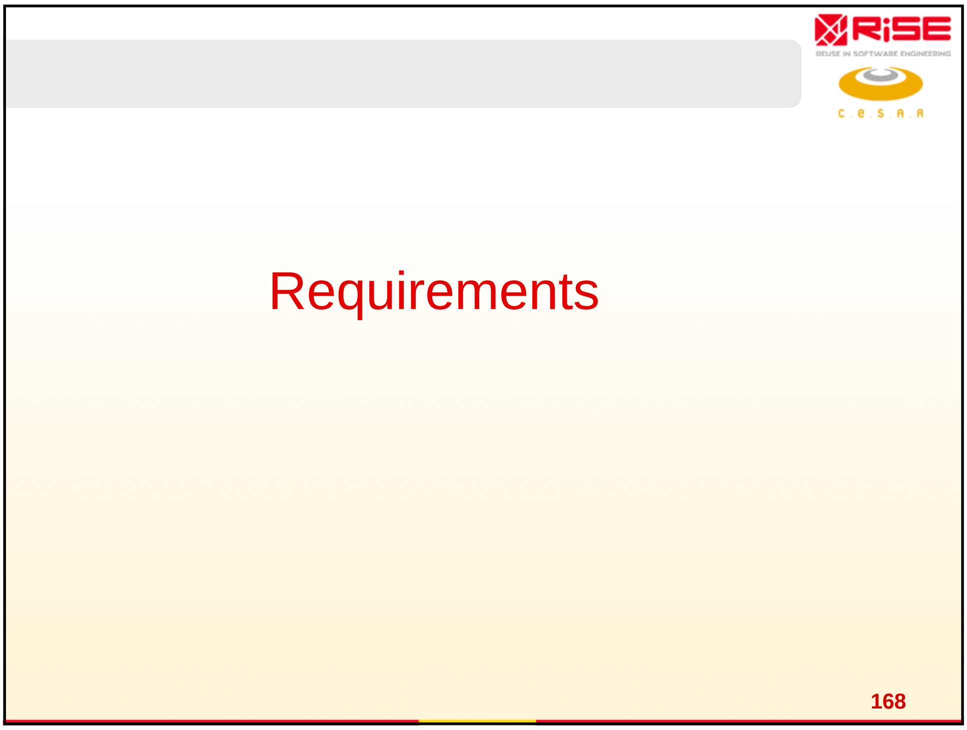


Features	RiSE Chair Conference		RiSE Chair Journal		RiSE Chair Plus		Scope
	Fut	Req	Fut	Req	Fut	Req	
Access Control	0	1	0	1	0	1	Mandatory ▾
Acctept/Reject Review	0	1	0	1	0	1	Mandatory ▾
Assignment - Automatic Indication	1	0	1	0	0	1	Variable ▾
Assignment - Chair Indication	0	1	0	1	0	1	Mandatory ▾
Assignment - Preference Indication	1	0	0	1	0	1	Variable ▾
Best Papers	1	0	0	1	0	1	Variable ▾
Build PDF	1	0	1	0	0	1	Variable ▾
Comments to Author	0	1	0	1	0	1	Mandatory ▾
Complete Submission	0	1	0	1	0	1	Mandatory ▾
Create Event from Previous	1	0	0	1	0	1	Variable ▾
Create Event From Scratch	0	1	0	1	0	1	Variable ▾
Deadline	1	0	0	1	0	1	Variable ▾
Delete Submission	0	1	0	1	0	1	Mandatory ▾
Document Simiralty Analysis	1	0	0	1	0	1	Variable ▾
Event Action History	1	0	0	1	0	1	Variable ▾
Event Opened for Submission	1	0	0	1	0	1	Variable ▾



Scoping - Contributions

- Main improvements to the RiPLE-SC
  - Pre-scoping phase
    - Identification of business goals
  - Domain scoping
    - Workshop of domain analysis
      - Prioritize domains and sub-domains
  - Product scoping
    - Features review meeting
  - Assets scoping
    - Create metrics of characterization and benefit
      - Prioritize product map



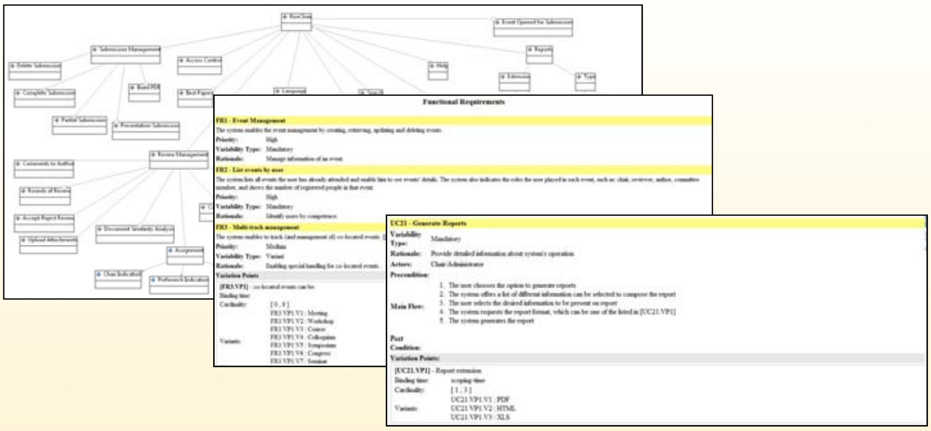
Requirements

Requirements

- Main activities
  - Construction of the **Feature Model**
  - Identification, specification and validation of **requirements** and **Use Cases**
  - Contruction of the traceability matrix
    - Split

Requirements

- Artifacts
  - Feature model, requirements , use cases, traceability matrix



Functional Requirements

FR1 - Event Management

The system enables the event management by creating, retrieving, updating and deleting events.

Priority: High
Variability Type: Mandatory
Rationale: Manage information of an event.

FR2 - List events by user

The system lists all events the user has already attended and enable him to see events' details. The system also indicates the roles the user played in each event, such as: chair, reviewer, author, committee member, and shows the number of registered people in that event.

Priority: High
Variability Type: Mandatory
Rationale: Identify users by competence.

FR3 - Multi-track management

The system enables to track (and management of) co-located events.

Priority: Medium
Variability Type: Variant
Rationale: Enabling special handling for co-located events.

Variation Points:

[FR3.VP1] - co-located events can be:
Binding time:
Cardinality: [0 .. 8]
Variants:
FR3.VP1.V1 : Meeting
FR3.VP1.V2 : Workshop
FR3.VP1.V3 : Course
FR3.VP1.V4 : Colloquium
FR3.VP1.V5 : Symposium
FR3.VP1.V6 : Congress
FR3.VP1.V7 : Seminar

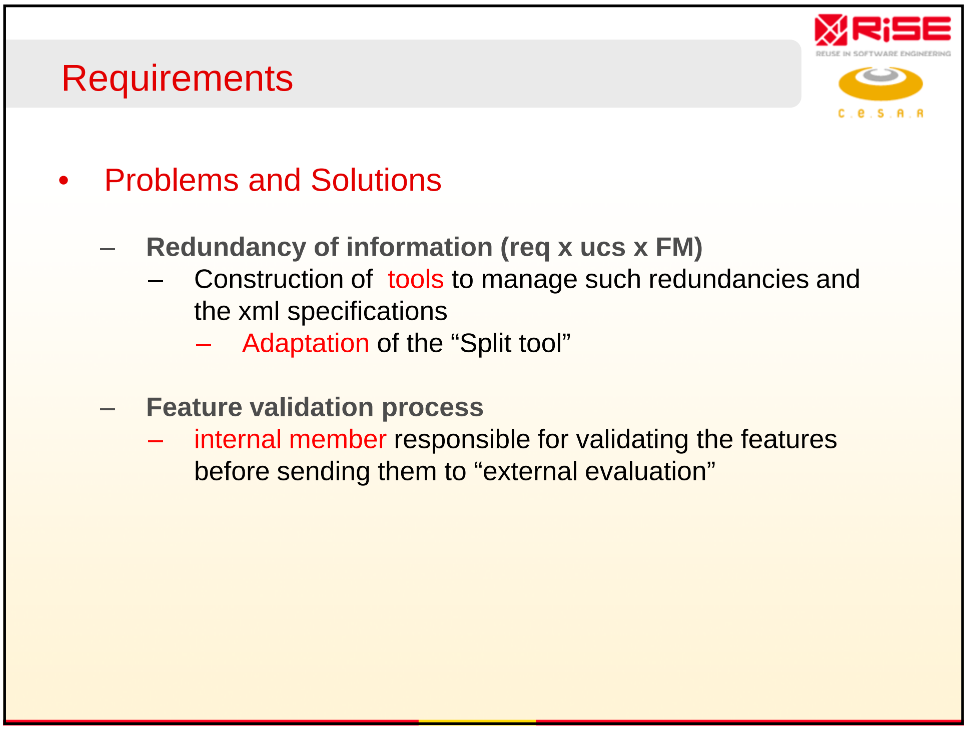
UC21 - Generate Reports

Variability Type: Mandatory
Rationale: Provide detailed information about system's operation
Actors: Chair/Administrator
Precondition:
Main Flow:
1. The user chooses the option to generate reports
2. The system offers a list of different information can be selected to compose the report
3. The user selects the desired information to be present on report
4. The system requests the report format, which can be one of the listed in [UC21.VP1]
5. The system generates the report

Post Condition:

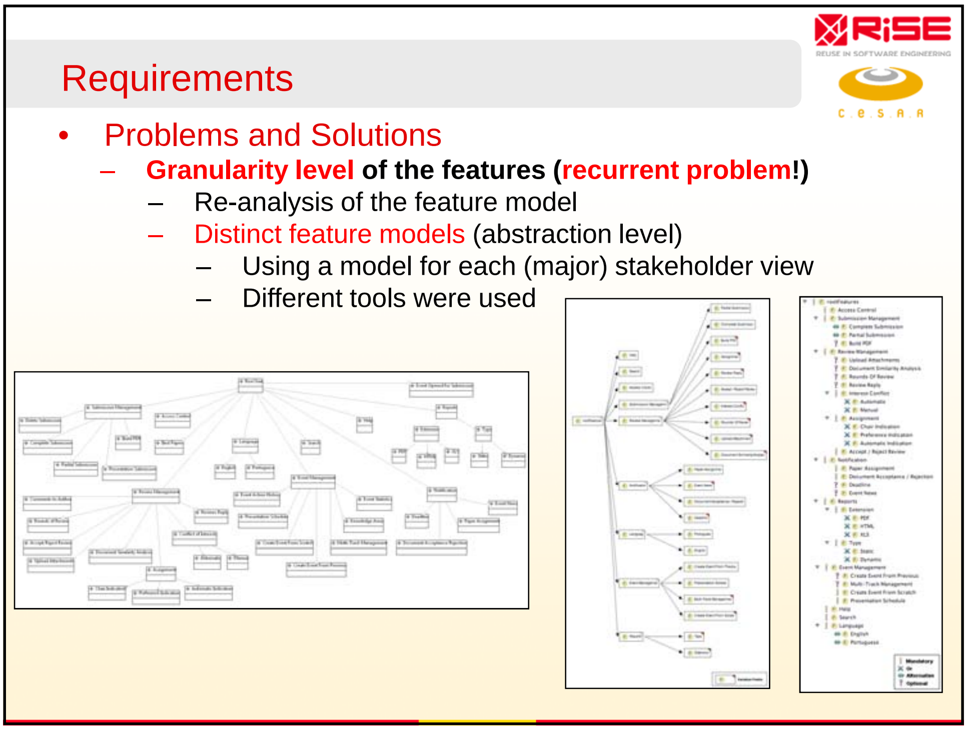
Variation Points:

[UC21.VP1] - Report extension
Binding time: scoping time
Cardinality: [1 .. 3]
Variants:
UC21.VP1.V1 : PDF
UC21.VP1.V2 : HTML
UC21.VP1.V3 : XLS



Requirements

- Problems and Solutions
  - **Redundancy of information (req x ucs x FM)**
    - Construction of tools to manage such redundancies and the xml specifications
      - Adaptation of the “Split tool”
  - **Feature validation process**
    - internal member responsible for validating the features before sending them to “external evaluation”

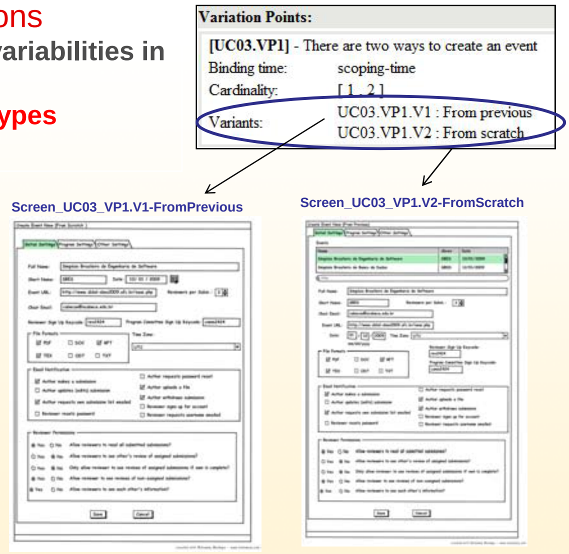


Requirements

- Problems and Solutions
  - **Granularity level of the features (recurrent problem!)**
    - Re-analysis of the feature model
    - Distinct feature models (abstraction level)
      - Using a model for each (major) stakeholder view
      - Different tools were used

Requirements

- Problems and Solutions
  - Difficult to identify variabilities in screens
    - Solution: **Prototypes**



Variation Points:

[UC03.VP1] - There are two ways to create an event

Binding time: scoping-time

Cardinality: [1 , 2]

Variants: UC03.VP1.V1 : From previous
UC03.VP1.V2 : From scratch

Screen_UC03_VP1.V1-FromPrevious
Screen_UC03_VP1.V2-FromScratch

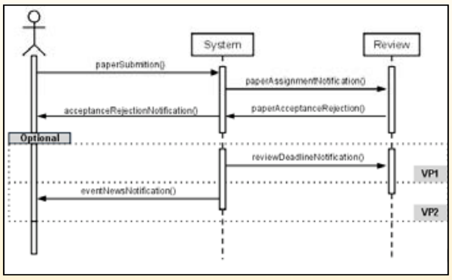
Design

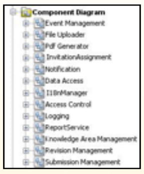
Design

- Main activities
  - Identification and priorization of quality attributes
  - Definition of architectural views
  - Specification of modules, components and classes

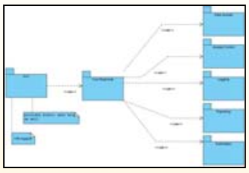
Design

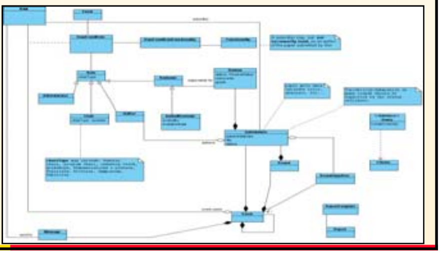
- Artifacts
  - Domain specific software architecture document; diagrams (components, classes, modules)
    - core-business: submission, event management, revision
    - Shared services: logging, notification, reporting, accesscontrol
  - Specification of variabilities


System
Review
paperSubmition()
paperAssignmentNotification()
acceptanceRejectionNotification()
paperAcceptanceRejection()
Optional
reviewDeadlineNotification()
VP1
eventNewsNotification()
VP2



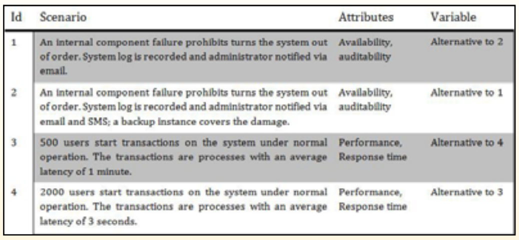
Component Diagram
- Event Management
- File Uploader
- Pdf Generator
- InvitationAssignment
- Notification
- Data Access
- I18nManager
- Access Control
- Logging
- ReportService
- Knowledge Area Management
- Revision Management
- Submission Management



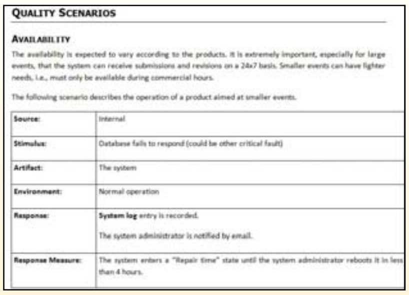


Design

- Quality scenarios documentation
  - Scenarios -> quality attibutes
  - Specification of variabilities in the scenarios



Id	Scenario	Attributes	Variable
1	An internal component failure prohibits turns the system out of order. System log is recorded and administrator notified via email.	Availability, auditability	Alternative to 2
2	An internal component failure prohibits turns the system out of order. System log is recorded and administrator notified via email and SMS; a backup instance covers the damage.	Availability, auditability	Alternative to 1
3	500 users start transactions on the system under normal operation. The transactions are processes with an average latency of 1 minute.	Performance, Response time	Alternative to 4
4	2000 users start transactions on the system under normal operation. The transactions are processes with an average latency of 3 seconds.	Performance, Response time	Alternative to 3



Quality Scenarios

Availability

The availability is expected to vary according to the products. It is extremely important, especially for large events, that the system can receive submissions and revisions on a 24x7 basis. Smaller events can have lighter needs, i.e., must only be available during commercial hours.

The following scenario describes the operation of a product aimed at smaller events.

Source:	Internal
Stimulus:	Database fails to respond (could be other critical fault)
Artifact:	The system
Environment:	Normal operation
Response:	**System log** entry is recorded. The system administrator is notified by email.
Response Measure:	The system enters a "Repair time" state until the system administrator reboots it in less than 4 hours.

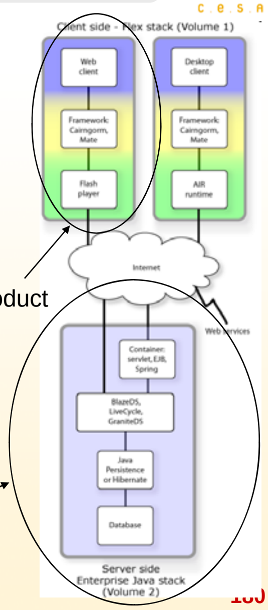
Design

- Problems and solutions
  - Lack of a deploy view
  - Non-fuctional variants should be considered
    - It is important to stablish a limit
      - Otherwise the creation of an architecture could be inviable
  - Specification of features interactions
    - Modification dependency
    - Concurrent dependency
    - Sequential dependency
    - Decomposition and Generalization Dependency
    - Usage dependency
    - Exclude dependency

Implementation

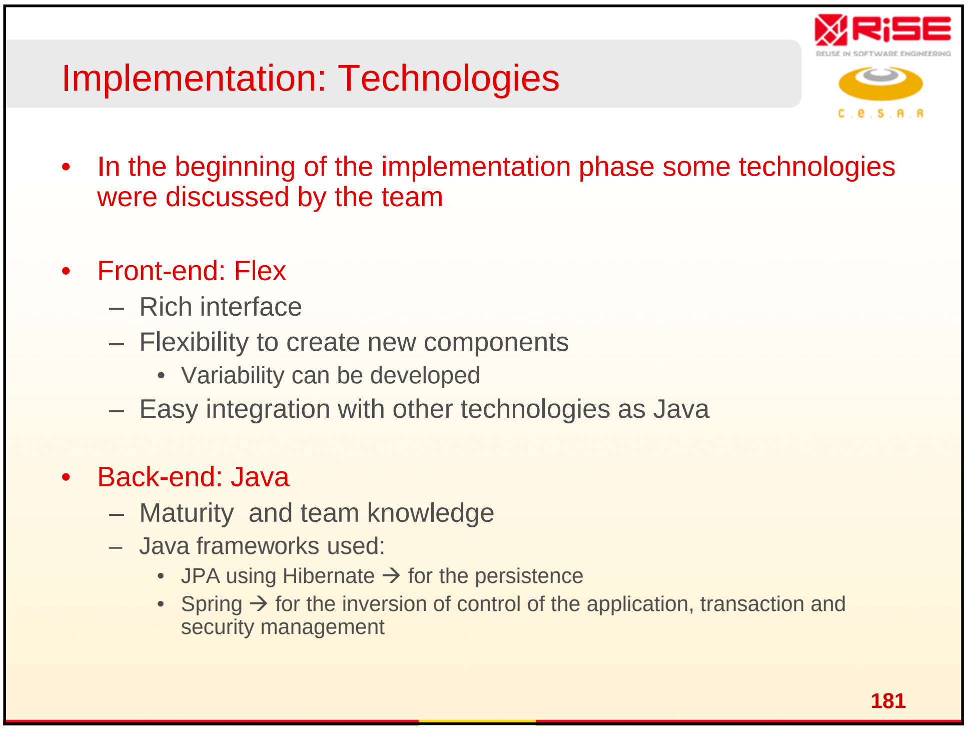
Implementation

- The architecture was divided in two layers
  - The front-end (gui): related to the products
    - Responsible for the product customization and "glue"
  - The back-end: related to the core assets
    - Common services
    - Business rules


Client side - Flex stack (Volume 1)
Web client
Framework: Cairngorm, Mate
Flash player
Desktop client
Framework: Cairngorm, Mate
AIR runtime
Internet
Web services
Container: servlet, EJB, Spring
BlazeDS, LiveCycle, GraniteDS
Java Persistence or Hibernate
Database
Server side
Enterprise Java stack
(Volume 2)

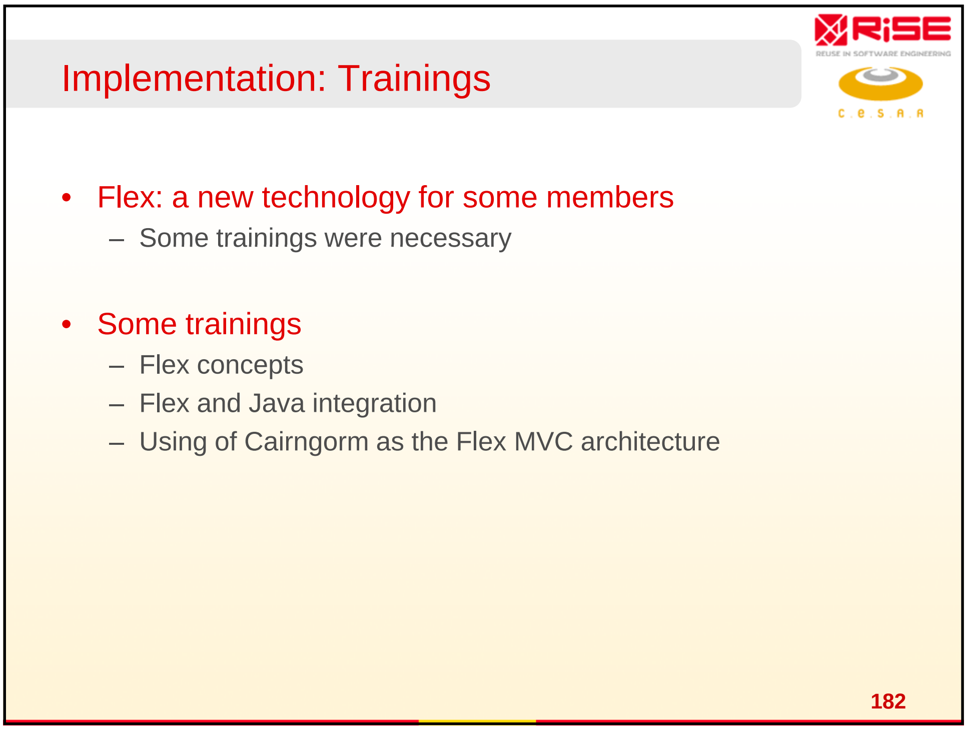
Product

Core Assets



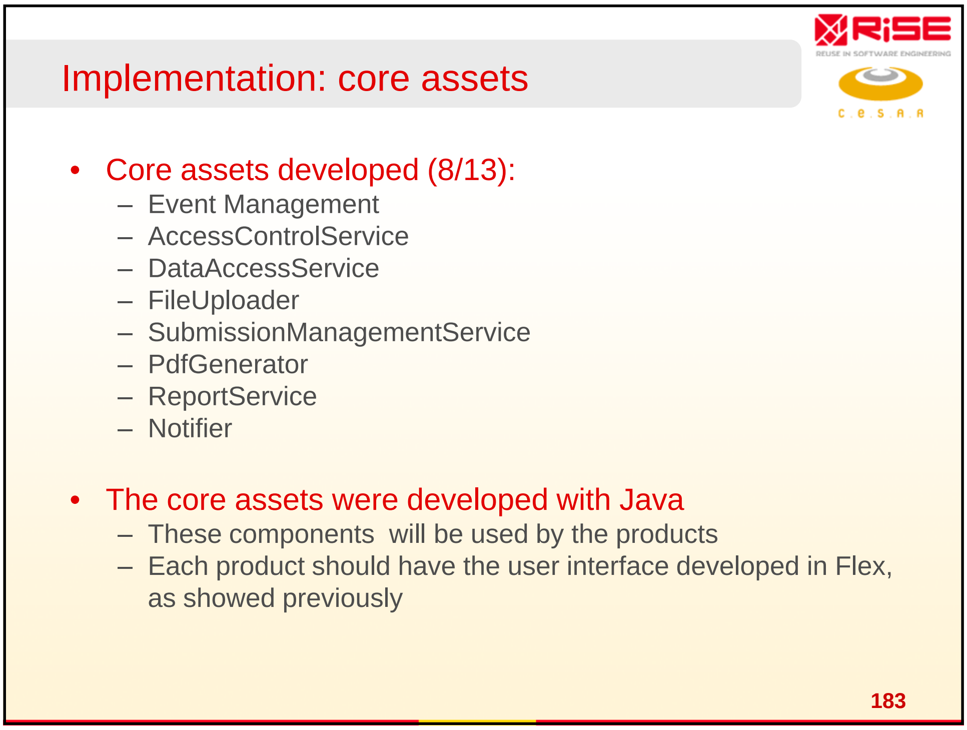
Implementation: Technologies

- In the beginning of the implementation phase some technologies were discussed by the team
- Front-end: Flex
  - Rich interface
  - Flexibility to create new components
    - Variability can be developed
  - Easy integration with other technologies as Java
- Back-end: Java
  - Maturity and team knowledge
  - Java frameworks used:
    - JPA using Hibernate → for the persistence
    - Spring → for the inversion of control of the application, transaction and security management



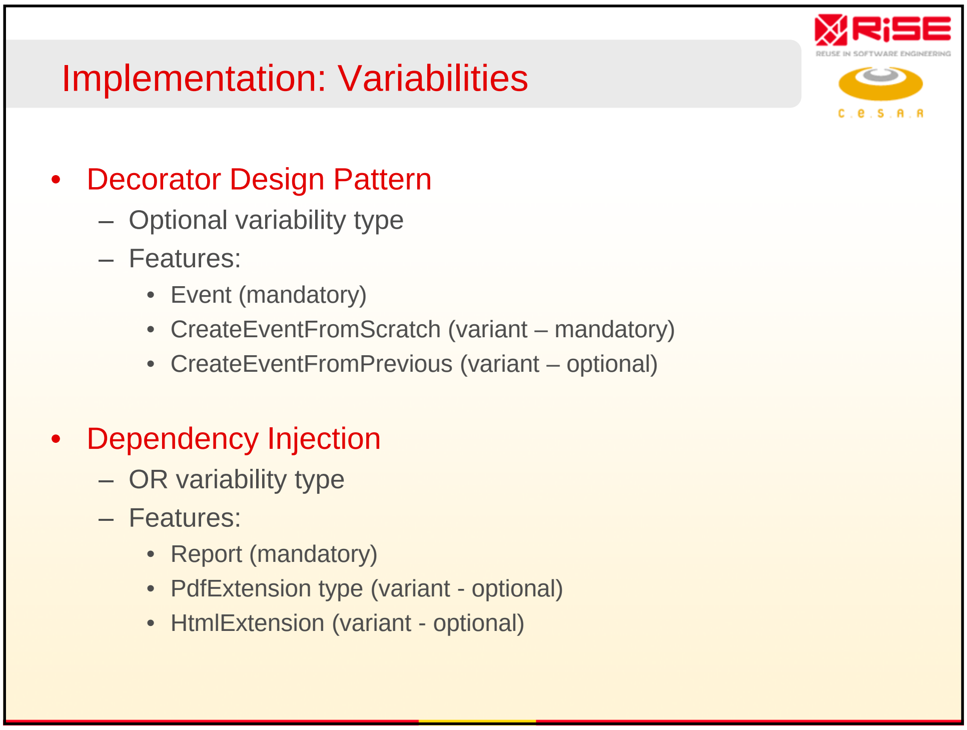
Implementation: Trainings

- Flex: a new technology for some members
  - Some trainings were necessary
- Some trainings
  - Flex concepts
  - Flex and Java integration
  - Using of Cairngorm as the Flex MVC architecture



Implementation: core assets

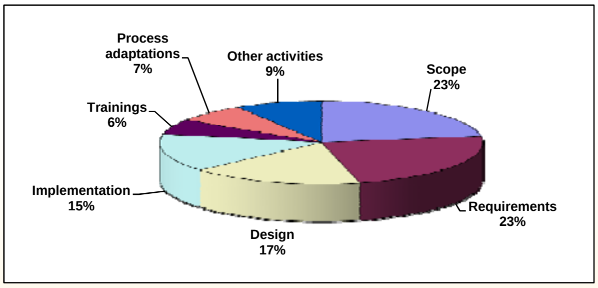
- Core assets developed (8/13):
  - Event Management
  - AccessControlService
  - DataAccessService
  - FileUploader
  - SubmissionManagementService
  - PdfGenerator
  - ReportService
  - Notifier
- The core assets were developed with Java
  - These components will be used by the products
  - Each product should have the user interface developed in Flex, as showed previously

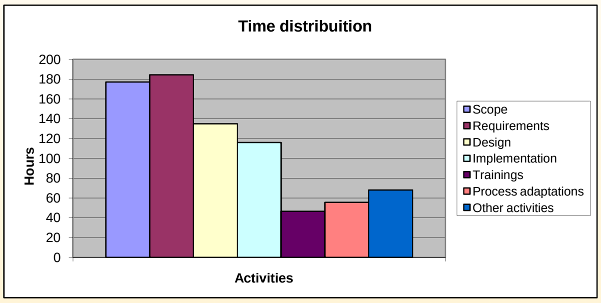


Implementation: Variabilities

- Decorator Design Pattern
  - Optional variability type
  - Features:
    - Event (mandatory)
    - CreateEventFromScratch (variant – mandatory)
    - CreateEventFromPrevious (variant – optional)
- Dependency Injection
  - OR variability type
  - Features:
    - Report (mandatory)
    - PdfExtension type (variant - optional)
    - HtmlExtension (variant - optional)

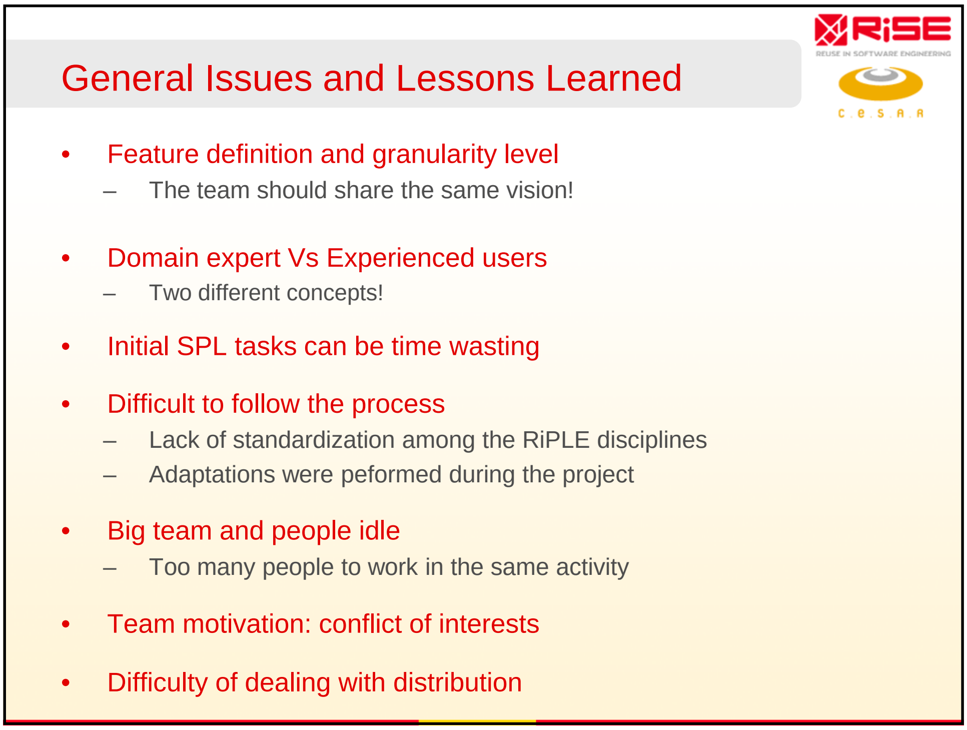
Some data: time distribution





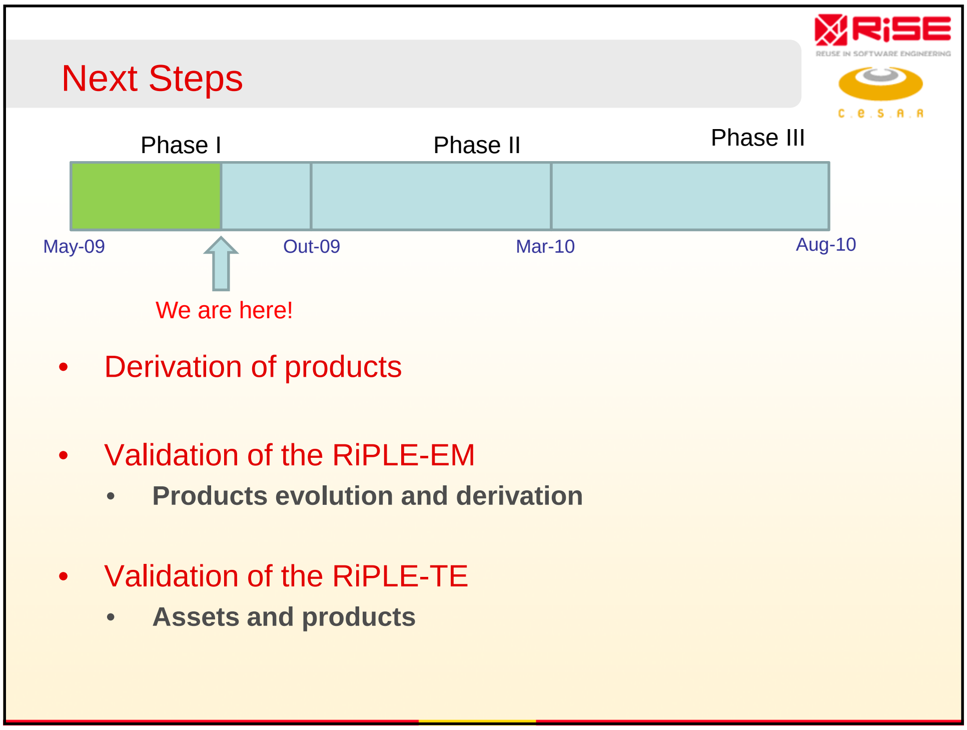
Some data

- Scope
  - 3 products
  - 41 features identified
- Requirements
  - 49 functional
  - 18 non-functional
  - 28 use cases specified
  - 23 screens (prototype)
- Design
  - 15 quality scenarios
  - 13 components
  - 7 modules
- Implementation
  - 8 core assets implemented
  - 20 classes



General Issues and Lessons Learned

- Feature definition and granularity level
  - The team should share the same vision!
- Domain expert Vs Experienced users
  - Two different concepts!
- Initial SPL tasks can be time wasting
- Difficult to follow the process
  - Lack of standardization among the RiPLE disciplines
  - Adaptations were peformed during the project
- Big team and people idle
  - Too many people to work in the same activity
- Team motivation: conflict of interests
- Difficulty of dealing with distribution



Next Steps

Phase I

Phase II

Phase III

May-09

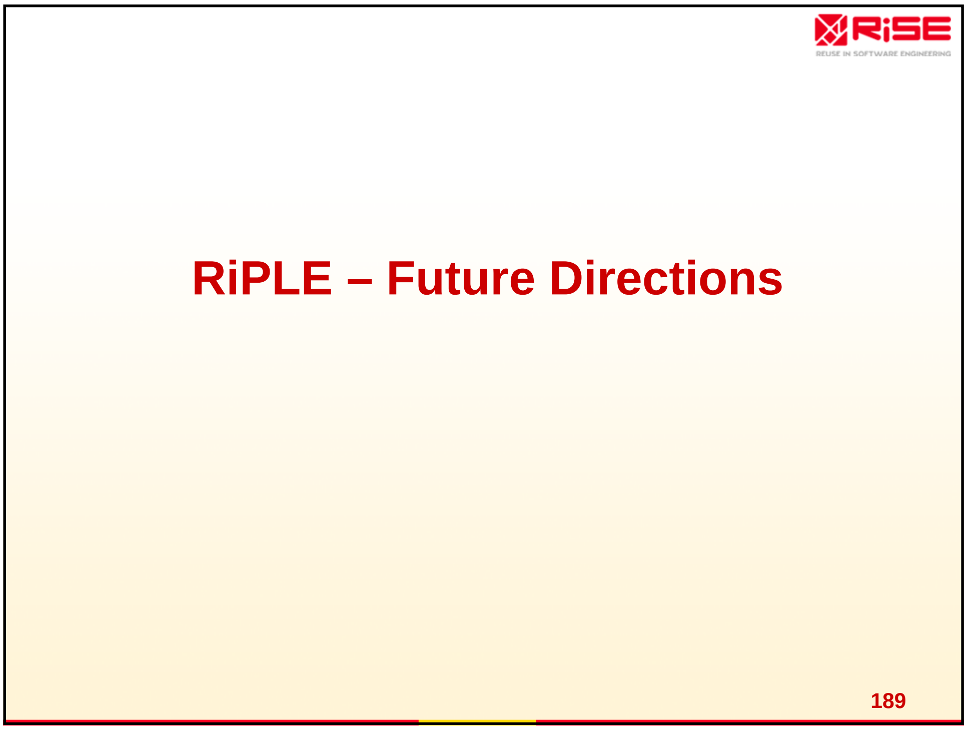
Out-09

Mar-10

Aug-10

We are here!

- Derivation of products
- Validation of the RiPLE-EM
  - Products evolution and derivation
- Validation of the RiPLE-TE
  - Assets and products

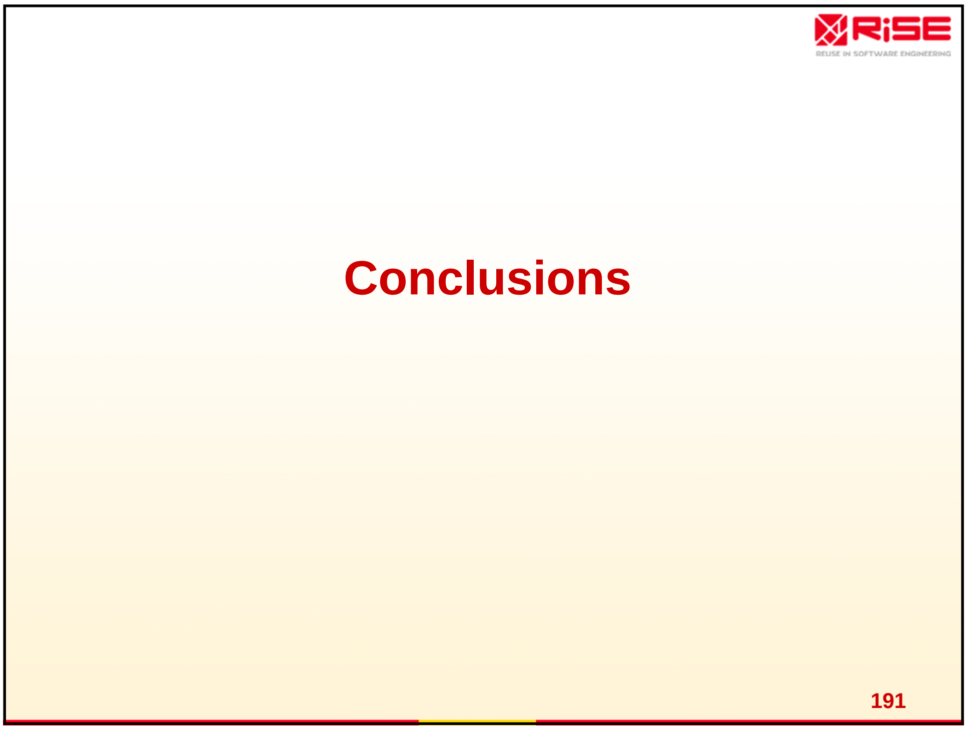


RiPLE – Future Directions

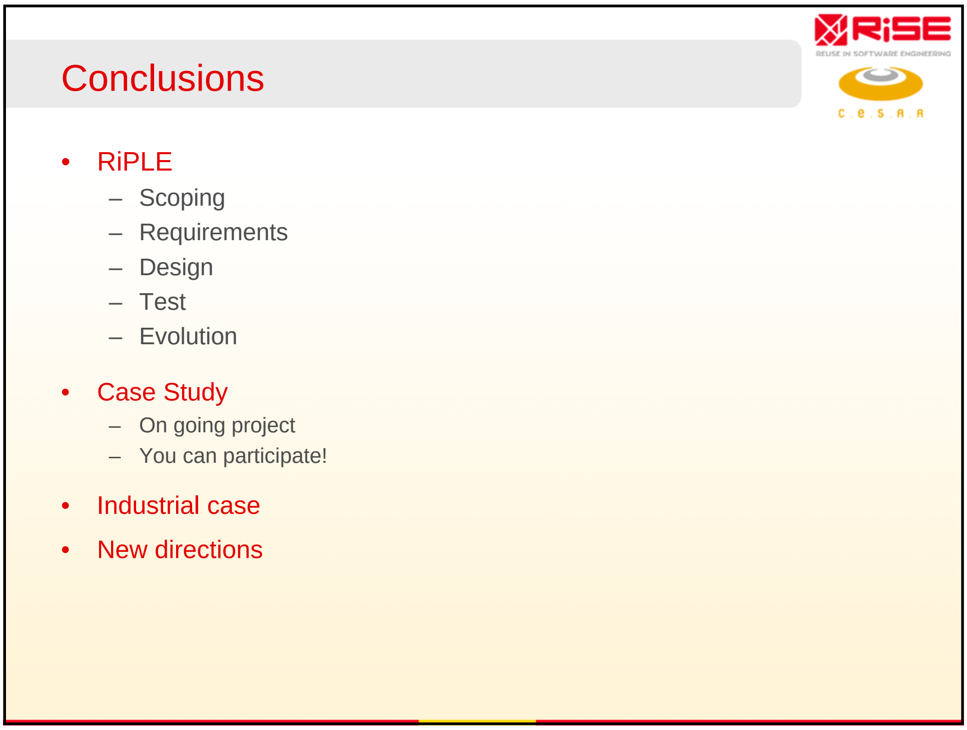
189

New Directions

- Research
  - Risk Management
  - Measurement
  - Feature Interaction
  - Architecture Recovery
  - Quality Attributes
  - Quality
    - Inspection
    - Test case selection | priorization
  - Product Derivation
  - Introduction in Companies
  - Tools
- Agile

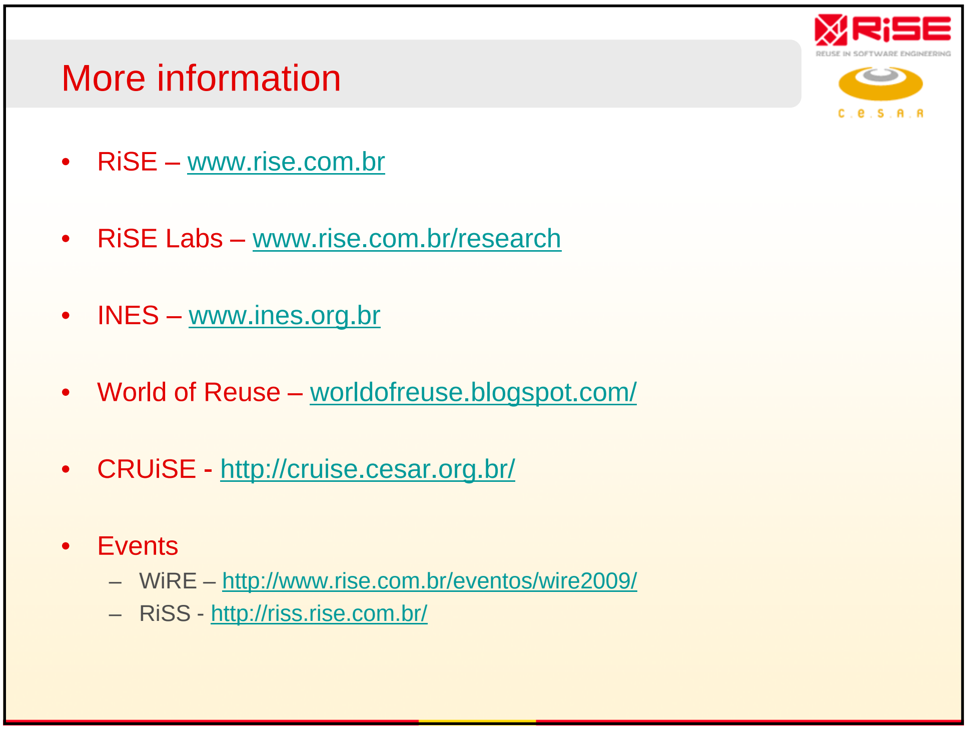


Conclusions



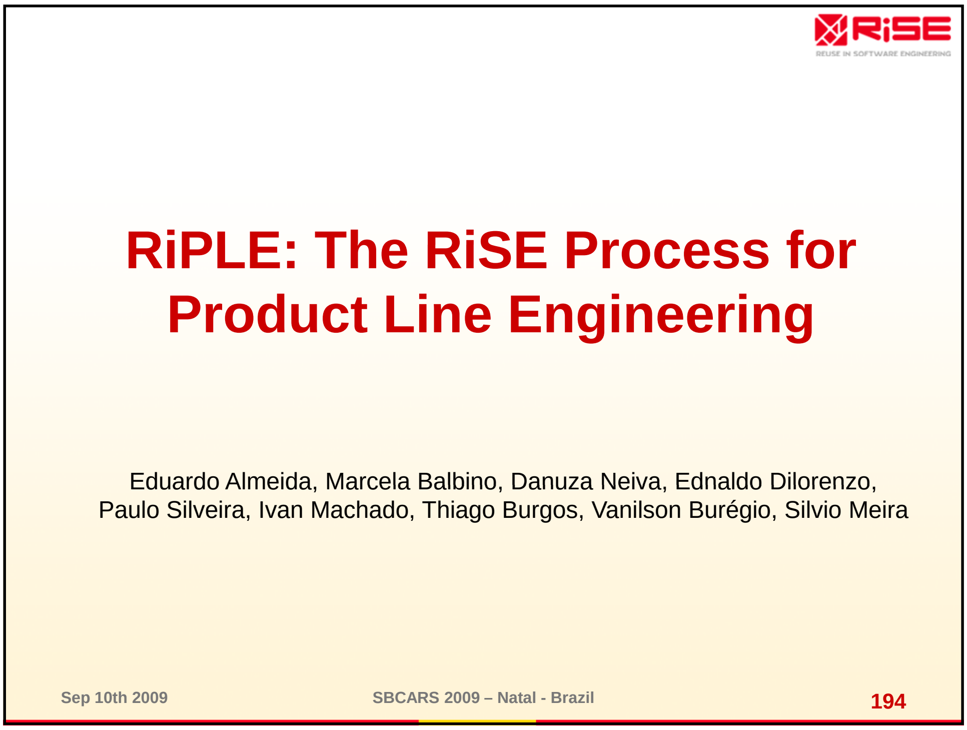
Conclusions

- RiPLE
  - Scoping
  - Requirements
  - Design
  - Test
  - Evolution
- Case Study
  - On going project
  - You can participate!
- Industrial case
- New directions



More information

- RiSE – www.rise.com.br
- RiSE Labs – www.rise.com.br/research
- INES – www.ines.org.br
- World of Reuse – worldofreuse.blogspot.com/
- CRUiSE - http://cruise.cesar.org.br/
- Events
  - WiRE – http://www.rise.com.br/eventos/wire2009/
  - RiSS - http://riss.rise.com.br/



RiPLE: The RiSE Process for Product Line Engineering

Eduardo Almeida, Marcela Balbino, Danuza Neiva, Ednaldo Dilorenzo,
Paulo Silveira, Ivan Machado, Thiago Burgos, Vanilson Burégio, Silvio Meira