



Evolution Management and Process for
Real-Time Embedded Software Systems

A Component Architecture for Evolution: Consolidation

Deliverable D.2.7

Edited by Stefan Van Baelen, K.U.Leuven

13. November 2003

Version 1.0

Status final

Public Version



I T E A

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT

This document is part of the work of the EUREKA Σ! 2023 – ITEA 00103 project EMPRESS.
Copyright © 2002-2003 EMPRESS consortium.

Authors/Partners:

Partner	Author	Email
Fraunhofer IESE	Hans-Gerhard Groß	grossh@iese.fhg.de
K.U. Leuven	Stefan Van Baelen	Stefan.VanBaelen@cs.kuleuven.ac.be

Document History:

Date	Version	Editor	Description
22 Sep 03	0.1	Stefan Van Baelen	Initial Document based on D2.1.2/2.2.2 v2.0 - Chapter 4
25 Sep 03	0.2	Stefan Van Baelen	Added material from the Eindhoven Meeting
13 Nov 03	1.0	Stefan Van Baelen	Additions from ESI

Filename: D2.7_v1.0.doc

Abstract

This document describes the positioning of WP2 within the EMPRESS project. It describes

- the interaction of and integration between the performed work and the developed methods and techniques of EMPRESS Work Package 2 "Evolution of the Component architecture"
- the positioning of 2 major tasks of WP2: the design-time evolution task and the run-time evolution task
- the connection between WP2 and the other EMPRESS work packages
- the positioning of EMPRESS WP2 on the Unified Process (UP) as a reference model

As such, this document is the link to integrate the relevant results obtained in WP2 with the results of the other EMPRESS work packages, and concentrates on a consolidation of the component architecture for evolution, based on the experiences from building system support and tools and the demonstrator development.

It is the goal of this document to serve as an entry point and reference framework to the performed work and the developed methods and techniques of EMPRESS Work Package 2 "Evolution of the Component architecture".

Keywords: Consolidation, Component Architecture, Architectural Support, Design-time Evolution, Run-time Evolution, static configuration, dynamic configuration, dynamic reconfiguration, Unified Process (UP).

ABSTRACT.....	3
1 INTRODUCTION.....	5
2 DETAILING THE EMPRESS PROCESS: WP2 EXTENSIONS TO THE UNIFIED PROCESS	5
3 MAPPING OF THE WP2 METHODS AND TECHNIQUES ONTO THE EMPRESS PROCESS	9
3.1 THE WP2 DESIGN-TIME METHODS AND TECHNIQUES	9
3.1.1 <i>Embedded Beans in Hardware/Software Component Integration.....</i>	<i>11</i>
3.1.2 <i>Flexible Component Approach for Design Time Evolution.....</i>	<i>12</i>
3.1.3 <i>Pattern-based Architecture Analysis and Design of Embedded Software Product Lines</i>	<i>13</i>
3.1.4 <i>The Interface Wrapper Architecture</i>	<i>14</i>
3.1.5 <i>Architectural Support for Validation of Component Assemblies.....</i>	<i>15</i>
3.2 THE WP2 RUN-TIME METHODS AND TECHNIQUES	16
3.2.1 <i>Feature Modelling using Aspect-Oriented Programming</i>	<i>18</i>
3.2.2 <i>The Bridge Pattern.....</i>	<i>19</i>
3.2.3 <i>State Transfer approaches</i>	<i>20</i>
3.2.4 <i>Resource-Aware Components and the CRuMB Run-Time Component System</i>	<i>21</i>
4 CONCLUSION.....	22

1 Introduction

This document is part of the work done in the EMPRESS Work Package 2 "Evolution of the Component architecture". It combines the results of the 2 major tasks of WP2 "Design-time Evolution" and "Run-time Evolution", which have been described in deliverables D2.1.2-D2.2.2 and D2.4.2-D2.5.2.

Using the Unified Process as a reference model, the results will be combined and integrated with each other, highlighting the links and interactions between them. In addition, this UP mapping can also be used to link the result of WP2 to the other EMPRESS Work Packages, given insight on the connections between them and the way they complement each other.

As such, this document is the link to integrate the relevant results obtained in WP2 with the results of the other EMPRESS work packages, and as such concentrates on a consolidation of the component architecture for evolution, based on the experiences from building system support and tools and the demonstrator development.

2 Detailing the EMPRESS Process: WP2 extensions to the Unified Process

Of the 9 defined UP disciplines (earlier called workflows), 4 are relevant to WP2: Analysis & Design, Implementation, Validation & Verification (called Test in UP) and Deployment. The other ones (Business Modelling, Requirements, Configuration & Change Management, Project Management and Environment) are treated in the other EMPRESS Work Packages or fall beyond the scope of the EMPRESS Project.

For these relevant disciplines, the UP workflow details, activities and activity steps has been further extended and detailed within the EMPRESS Project. On the other hand, the 'Run-Time' phase that has been added to the UP as a 5th phase is also crucial within WP2.

Below, you can find the description of the EMPRESS Process WP2 details. The color codes used are the following: red = standard UP, black = EMPRESS extension, Green = detailed mapping of partner contribution to the EMPRESS Process]

- EMPRESS Process [Missing: Barco/DC/Siemens]
 - Analysis & Design
 - SW Feature approach [Magdeburg]
 - Feature Analysis
 - Feature Modelling
 - Define anticipated change /variation points [FIRST] [IESE-Gerd]
 - Feature variation point [ESI-MSI]
 - Interface variation point
 - Component variation point [ESI-MSI]
 - Define a candidate architecture (early elaboration iteration) Architecture Development [ESI-MSI]

2 Detailing the EMPRESS Process: WP2 extensions to the Unified Process

- Architecture analysis
- Architectural guidelines [IESE-Gerd]
- Influencing SW architecture with NFRs (QoS)
- Refine the architecture (elaboration iteration) Refactoring (Nth run)
- (Design)
 - Analyze behavior [IESE-Gerd]
 - Design components [KULeuven] [ESI-MSI] [IESE-Gerd]
 - Define needed components (feature2component mapping) [IESE-Gerd]
 - Search for suitable components & adapt them [UNIS] [IESE-Gerd]
 - Model & develop new components [UNIS] [IESE-Gerd]
 - Design non real-time components [UNIS]
 - Design real-time components [UNIS] [IESE-Gerd]
 - Specification/analysis of Component QoS [KULeuven]
 - Design the database (optional)
- QoS Mgmt
 - Resource Budget mgmt (space, time)
 - NFR constraint analysis
- Implementation
 - Structure the implementation model
 - Plan the integration
 - Implement components
 - Generate code [FIRST] [KULeuven] [UNIS] [ESI-MSI]
 - Implement components from scratch [UNIS]
 - Re-implement components [UNIS]
 - Integrate each subsystem [IESE-Gerd]
 - First integration
 - Re-integration
 - Integrate the system
 - First integration
 - Re-integration
- Validation & Verification (Test) [IESE-Gerd]
 - Plan test [IESE-Gerd]

2 Detailing the EMPRESS Process: WP2 extensions to the Unified Process

- Design test [IESE-Gerd]
- Implement test [IESE-Gerd] [ESI-MSI]
- Execute unit tests [IESE-Gerd]
- Execute tests in integration test stage [IESE-Gerd]
- Execute tests in system test stage [IESE-Gerd]
- Evaluate test [IESE-Gerd]
- Deployment
 - Deployment scenarios
 - Deployment scenario modelling
 - Start-up scenario modelling [Magdeburg]
 - Update scenario modelling (Nth run)
 - Plan deployment
 - SW Packaging
 - Develop support material
 - Test product at development site
 - Create release
 - Beta-test release
 - (non beta-release)
 - Test product at installation site (custom install)
 - Package product (Shrink-wrapped product)
 - Provide access to download site (downloadable software)
 - Installation
 - Initial installation [KULeuven]
 - Full Replacement (Nth run) = design-time update
 - Dynamic Replacement (Nth run) = run-time update [KULeuven]
 - Configuration
 - Static (Basic) configuration [Magdeburg] [UNIS]
 - System configuration
 - Component composition/configuration
 - Composition correctness validation
 - Functional
 - Non-functional (QoS)
 - Adapt components to SW environment
 - Dynamic Configuration [KULeuven] [Magdeburg]

2 Detailing the EMPRESS Process: WP2 extensions to the Unified Process

- Dynamic Reconfiguration (Nth run) [FIRST] [KULeuven]
- Training
 - Initial training
 - Transition training
 - Training update (Nth run)

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.1 The WP2 Design-Time Methods and Techniques

The report of the EMPRESS WP2 Design-time Task (Deliverable D2.1.2/2.2.2) has introduced notations, techniques and processes that support the change of a component-based embedded system at design time. It has briefly defined what we mean by design time evolution, that is an evolutionary development step (advancement, or update) of the system performed while the system is not operational. This stands in contrast with run-time evolution where updates of the system are performed when it is actually operational. The main parts have introduced notations, methods and a process for performing system change. This is basically a standard development process such as the Kobra method that is amended with techniques that the particular requirements of embedded system impose on development. These techniques were detailed in the second main section.

The goal of this section is mainly to integrate the introduced architectural techniques in the overall EMPRESS process. From the WP2 Design-time task document, we can identify a number of core methods for the development of component-based embedded systems that are related to architectural issues in design-time evolution. The different methods can be assigned to the two main areas of interest:

- Model, Notation and Processes for Design-Time Evolution
- Architectural Support for Design-Time Evolution

The first item is mainly concerned with very general observations regarding design-time evolution. Models, Notations and Processes support developers in achieving this task, e.g. methods for functional and QoS Validation. Other issues that belong to this first class are the Flexible Component Approach and the Embedded Beans.

The second item concentrates more how a concrete system should be organized ideally in order to support design-time evolution activities. Here we have some product-line approaches, although it is difficult to draw the border between the two areas of interest, the Interface Wrapper Architecture, and the Built-in Testing Technology.

1 gives a high-level overview of the interest points of the WP2 Design-Time Methods and Techniques within the Unified Process.

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

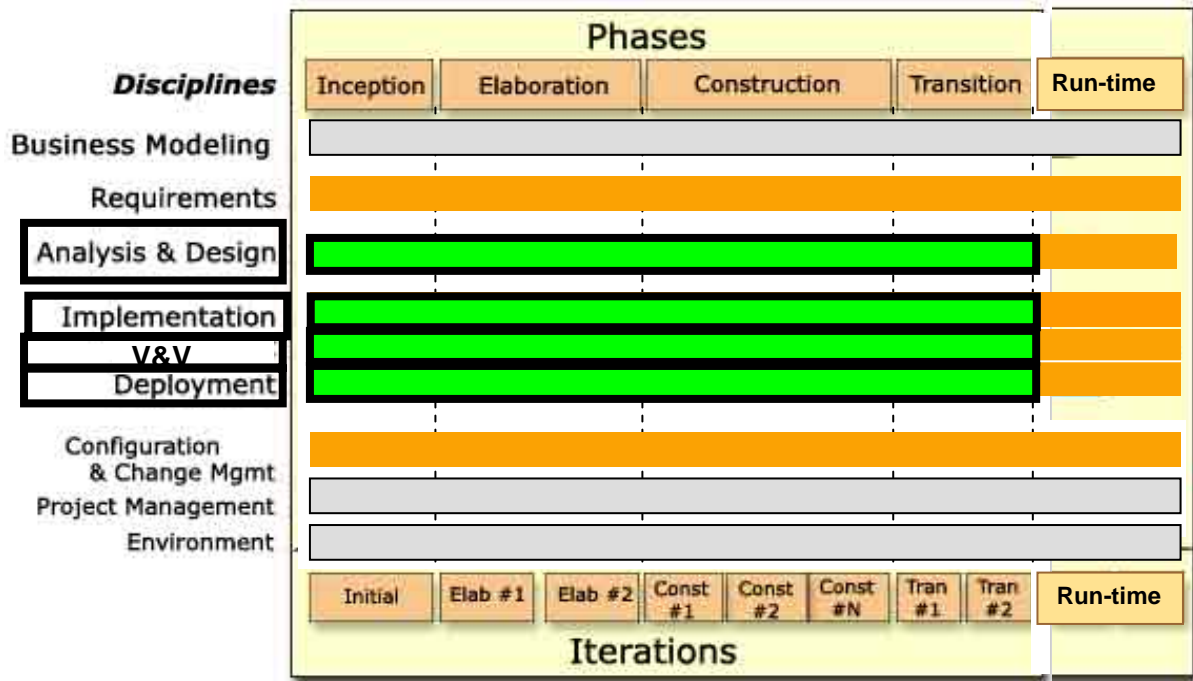


Figure 1: Interest points of Design-Time Methods and Techniques within the UP

Figure 2 provides a more detailed view on the locations of the methods and techniques on the UP/EMPRESS Process chart. Each area of interest is assigned a particular color. The area regarding "Model, Notation and Processes for Design-Time Evolution" is presented in turquoise, whereas the area regarding "Architectural Support for Design-Time Evolution" is presented in dark green.

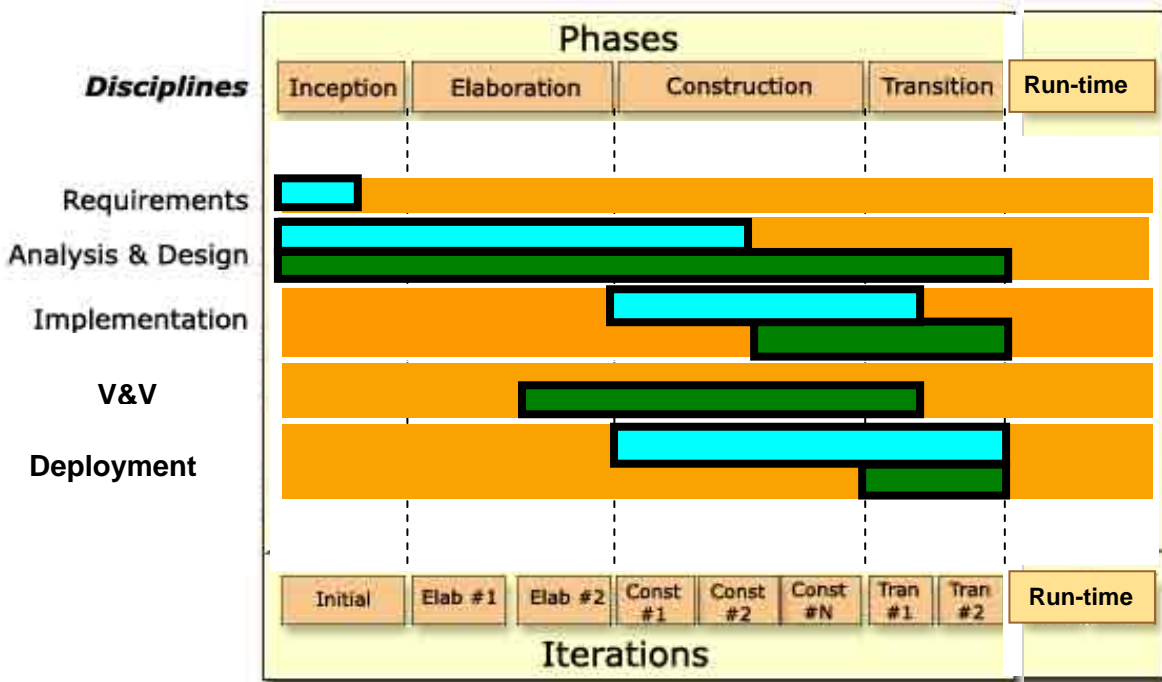


Figure 2: Mapping of the design-time methods and techniques onto the EMPRESS Process.

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

In the remaining of this section, a positioning for each of the approaches, methods and techniques will be given, using the UP/EMPRESS Process chart as a reference. All the techniques of the design-time task are related to five EMPRESS Process disciplines (Requirements, Analysis & Design, Implementation, Validation & Verification and Deployment), whereby the effects on the Requirements and Validation & Verification disciplines are only of minor importance.

3.1.1 Embedded Beans in Hardware/Software Component Integration

The Embedded Beans is almost fully located in the construction and transition phase within the EMPRESS Process, because we are mainly looking at a tool for component assembly with respect to hard/software integration. This is typical embodiment activity. However, since there are some design issues to be resolved before we can apply the embedded beans technique it has also some minor impact on the elaboration phase.

A detailed description of the Embedded Beans approach can be found in Section 2.3 of Deliverable D2.1.2/D2.2.2

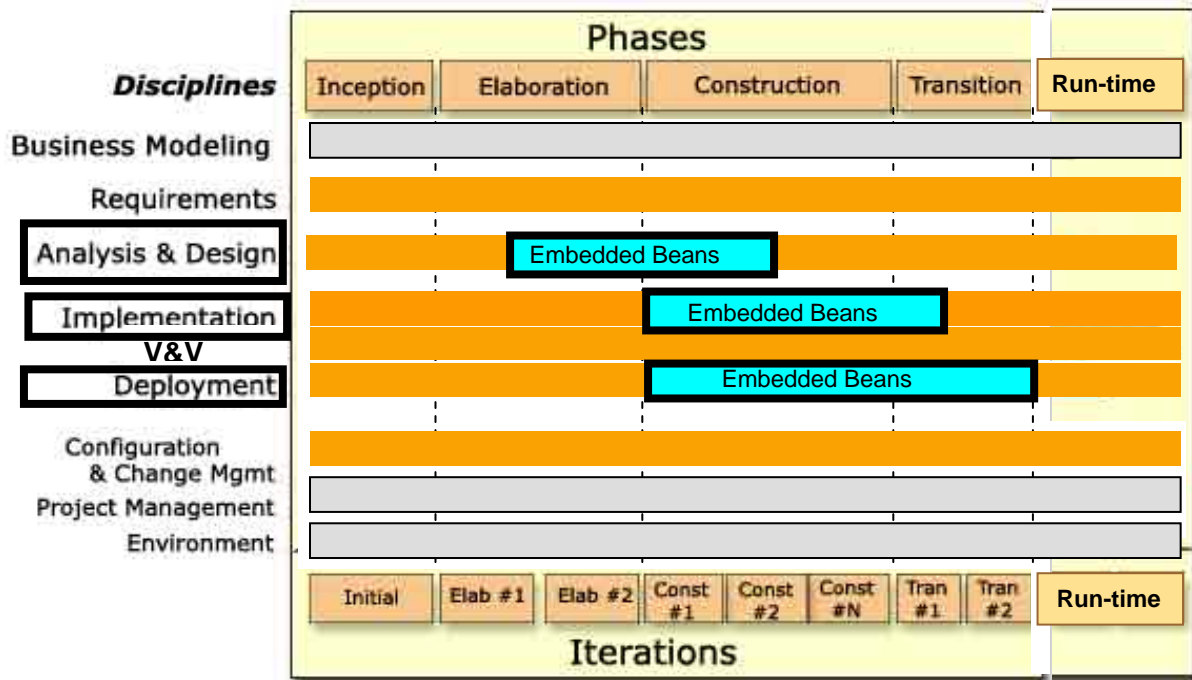


Figure 3: The Embedded Beans approach

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.1.2 Flexible Component Approach for Design Time Evolution

The product line engineering technique "Flexible Components" is primarily located on the Analysis & Design discipline. It is already starting at the very earliest development phase of Inception (with a small part located in the requirements discipline), and span through Elaboration into Construction. It means these techniques are readily used throughout these phases. The code generation and usage validation within the approach has an impact on Validation & Verification and Deployment, although the effect is only of minor importance.

A detailed description of the Flexible Component approach can be found in Section 2.5 and Appendix A of Deliverable D2.1.2/D2.2.2.

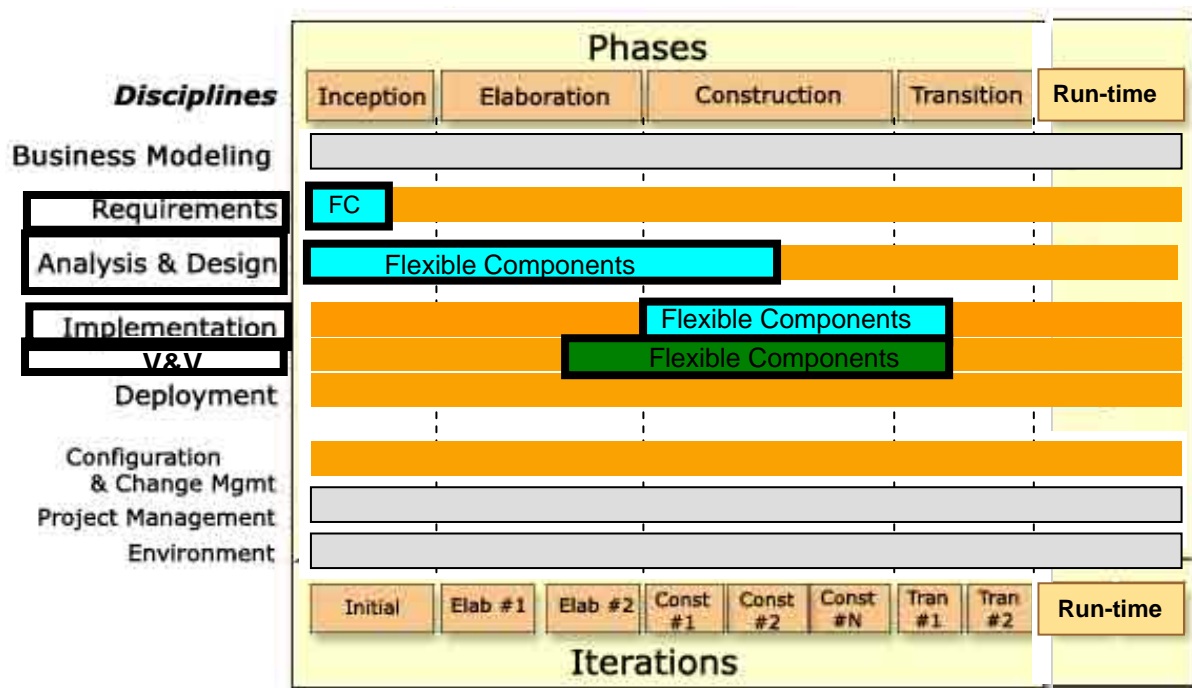


Figure 4: The Flexible Components approach

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.1.3 Pattern-based Architecture Analysis and Design of Embedded Software Product Lines

The product line engineering technique "Pattern-based Architecture & Design" is located on the Analysis & Design discipline. It starts at the very earliest development phase of Inception, and span through Elaboration into Construction.

A detailed description of the Pattern-based Architecture can be found in Section 3.1 and Appendix B of Deliverable D2.1.2/D2.2.2.

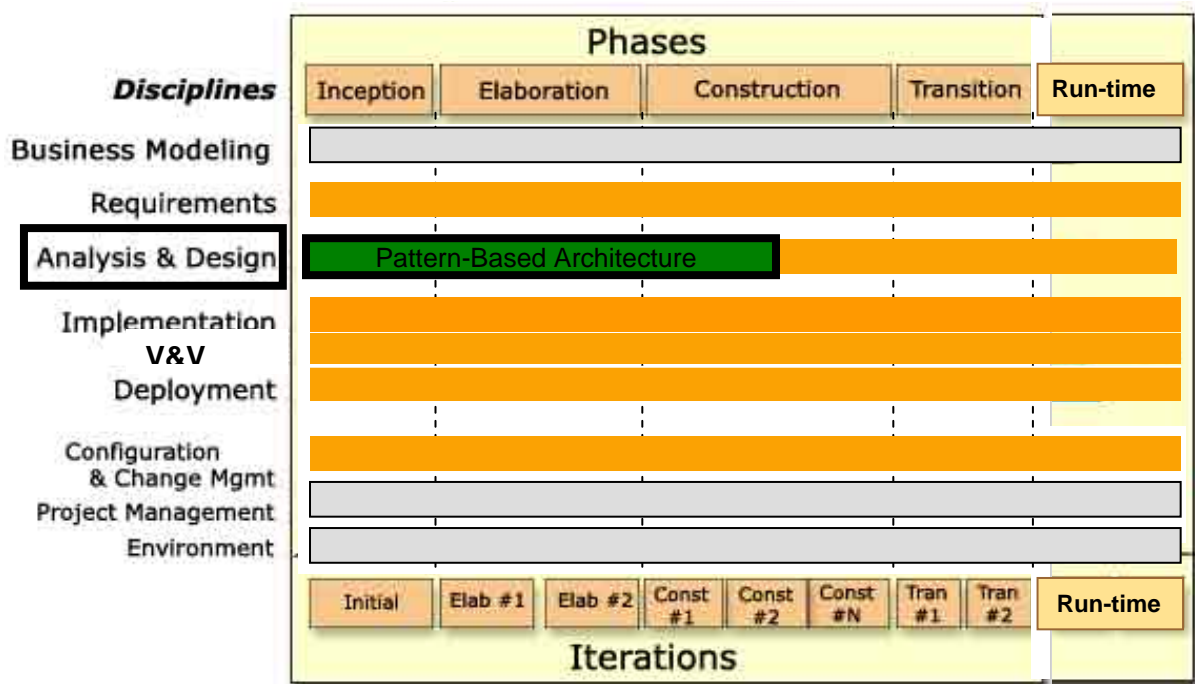


Figure 5: Pattern-Based Architecture Analysis and Design of Embedded Software Product Lines

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.1.4 The Interface Wrapper Architecture

The Interface Wrapper Architecture starts towards the end of the Elaboration phase when the overall component architecture and component selection and assembly plan is more or less established. Only if we have defined our components can we start to define the necessary wrappers that map component contracts. The main emphasis and effort for this technique is spread throughout the Construction and Implementation Phase. Since the component wrappers also have to be implemented and deployed, there is an impact on the Implementation and Deployment disciplines, although the effect is only of minor importance.

A detailed description of the Interface Wrapper Architecture can be found in Section 3.2 and Appendix C of Deliverable D2.1.2/D2.2.2.

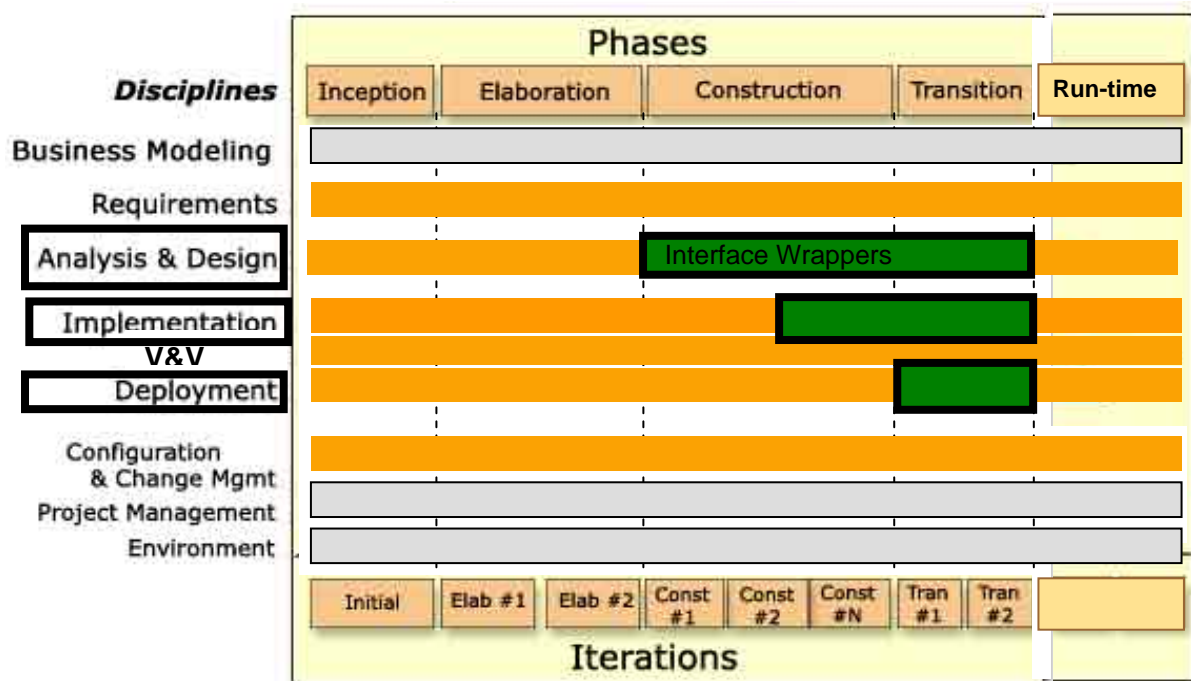


Figure 6: The Interface Wrapper Architecture

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.1.5 Architectural Support for Validation of Component Assemblies

The main emphasis and effort for this technique is spread throughout the Construction and Implementation Phase. Since the component wrappers also have to be implemented and deployed, there is an impact on the Implementation and Deployment disciplines, although the effect is only of minor importance.

The built-in contract testing technique starts towards the end of the Elaboration phase when the component architecture is more or less established. Only if we have defined our components can we define the testing architecture. The main emphasis and effort for this technique is spread throughout the Construction Phase. We also have some activities during the Validation & Verification discipline but this is concentrating on Construction effort rather than Transition or Run-Time.

A detailed description of the Architectural support for Validation of Component Assemblies can be found in Section 3.3 of Deliverable D2.1.2/D2.2.2.

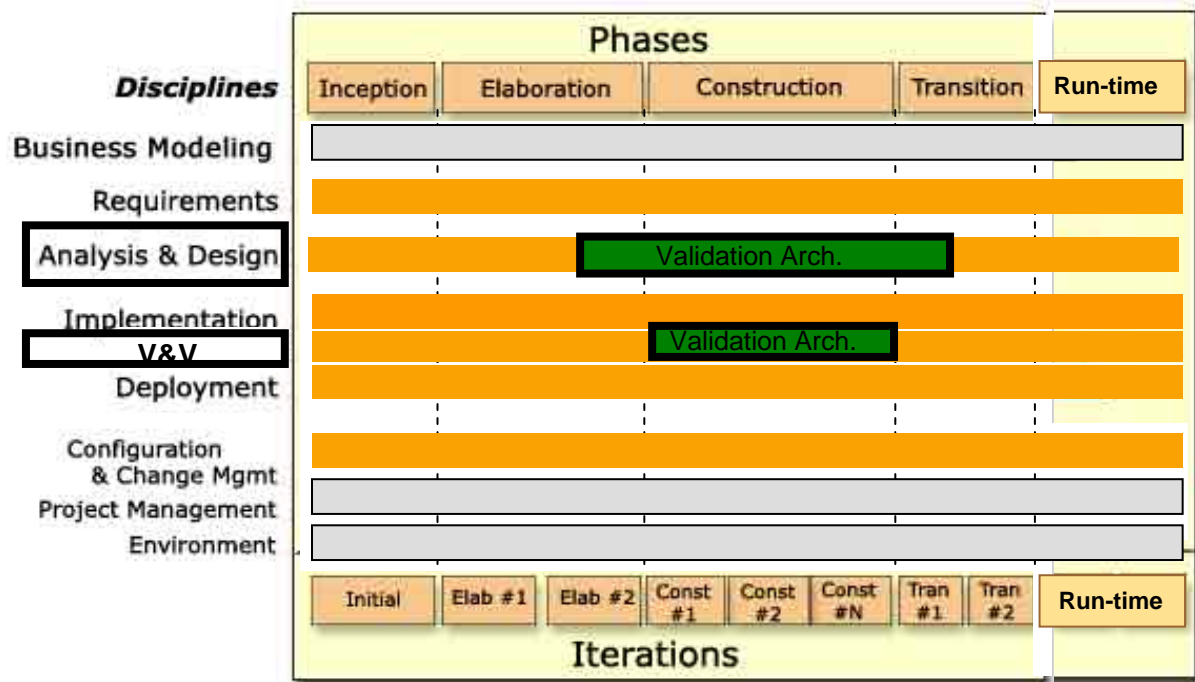


Figure 7: Architectural Support for Validation of Component Assemblies

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.2 The WP2 Run-Time Methods and Techniques

The report of the EMPRESS WP2 Run-time Task (Deliverable D2.4.2/2.5.2) has introduced notations, techniques and processes that support the change of a component-based embedded system at run-time. It has explained the difference between static configuration, dynamic configuration and dynamic reconfiguration, and concluded that although dynamic reconfiguration can be necessary for a number of embedded systems, dynamic configuration can be the best compromise for deeply embedded systems with limited room for additional run-time overhead.

The first part of the document describes the main issues for run-time evolution have been presented, that is synchronization of exchanges, component connections, state transfer and interface changes. In the second part of the document, architectures, techniques, design patterns and approaches are presented to provide a solution for these issues.

The goal of this section is mainly to integrate the introduced architectural techniques in the overall EMPRESS process. From the WP2 Run-time task document, we can identify a number of core methods for the development of component-based embedded systems that supports evolution and reconfiguration at run-time. The different methods can be assigned to the two main areas of interest:

- Design Techniques and Patterns for Run-time Evolution
- Resource-Aware Components in a supporting Run-time Component System

The first item is mainly concerned with very general observations regarding run-time evolution. Models, approaches and techniques support developers in achieving this task, e.g. Feature Modelling using Aspect-Oriented Programming and the Bridge Pattern. Other issues that belong to this first class are State Transfer approaches, Introspection & Meta-information and update control.

The second item concentrates more on a full-blown supporting environment for Resource-Aware Components, enabling components and the embedded system to fine-tune their behaviour according to the available resources at each moment. CruMB, the developed resource-aware run-time component system provides a robust environment for resource-aware component updates and QoS evolution.

Figure 8 gives a high-level overview of the interest points of the WP2 Run-Time Methods and Techniques within the Unified Process.

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

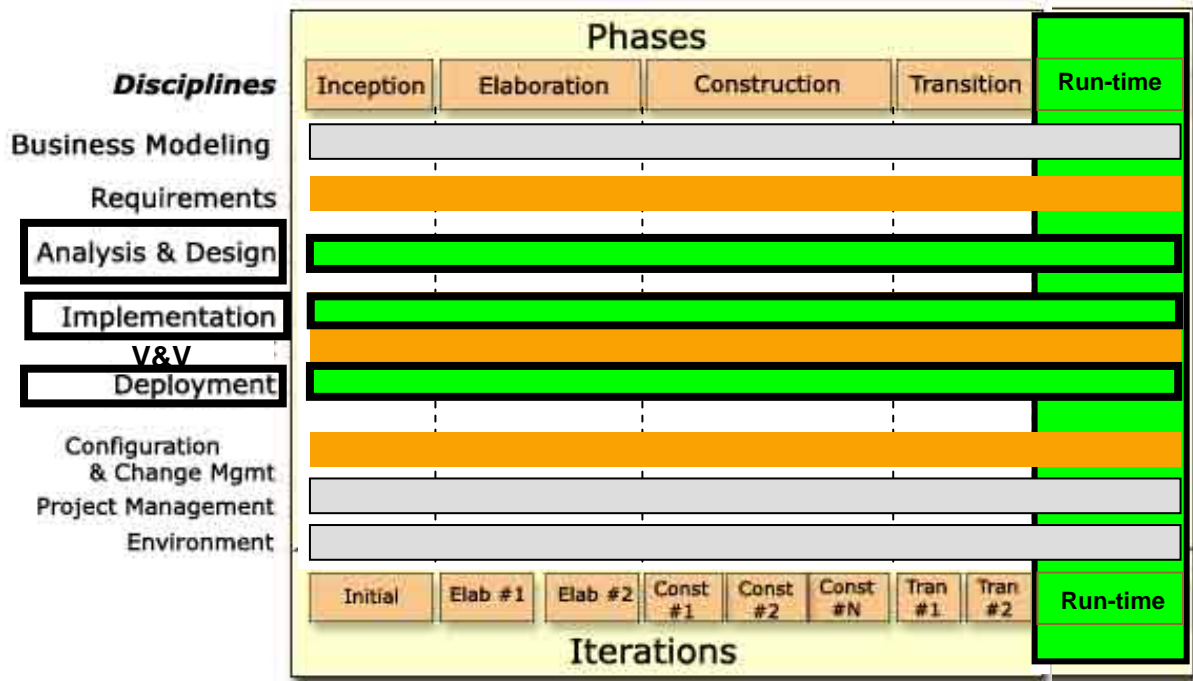


Figure 8: Interest points of Run-Time Methods and Techniques within the UP

Figure 8 provides a more detailed view on the locations of the methods and techniques on the UP/EMPRESS Process chart. Each area of interest is assigned a particular color. The area regarding "Design Techniques and Patterns for Run-time Evolution" is presented in turquoise, whereas the area regarding "Resource-aware components" is presented in dark green.

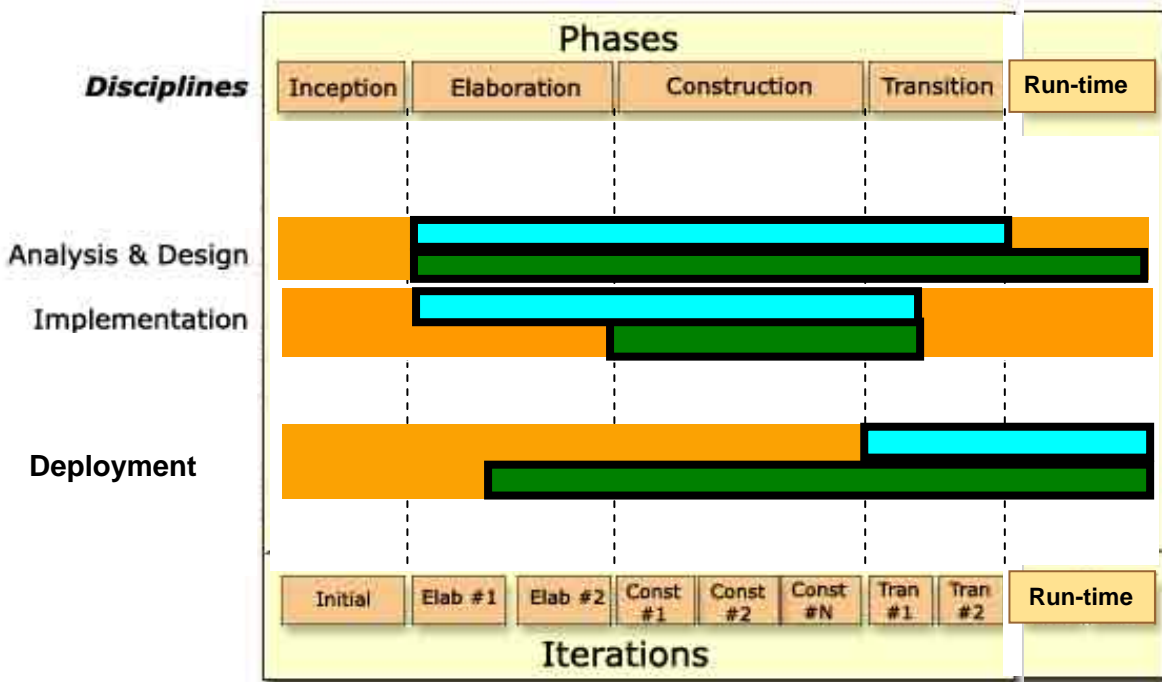


Figure 9: Mapping of the run-time methods and techniques onto the EMPRESS Process.

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

In the remaining of this section, a positioning for each of the approaches, methods and techniques will be given, using the UP/EMPRESS Process chart as a reference. All the techniques of the run-time task are related to three EMPRESS Process disciplines (Analysis & Design, Implementation and Deployment).

3.2.1 Feature Modelling using Aspect-Oriented Programming

Feature modelling using aspect-oriented programming is primarily located on the Analysis & Design discipline. It starts at the Elaboration phase, through Construction into the transition phase. The realization of feature modeling using AOP aspects are located on the implementation discipline.

A detailed description of Feature Modelling using aspect-oriented programming can be found in Section 2.1 of Deliverable D2.4.2/D2.5.2

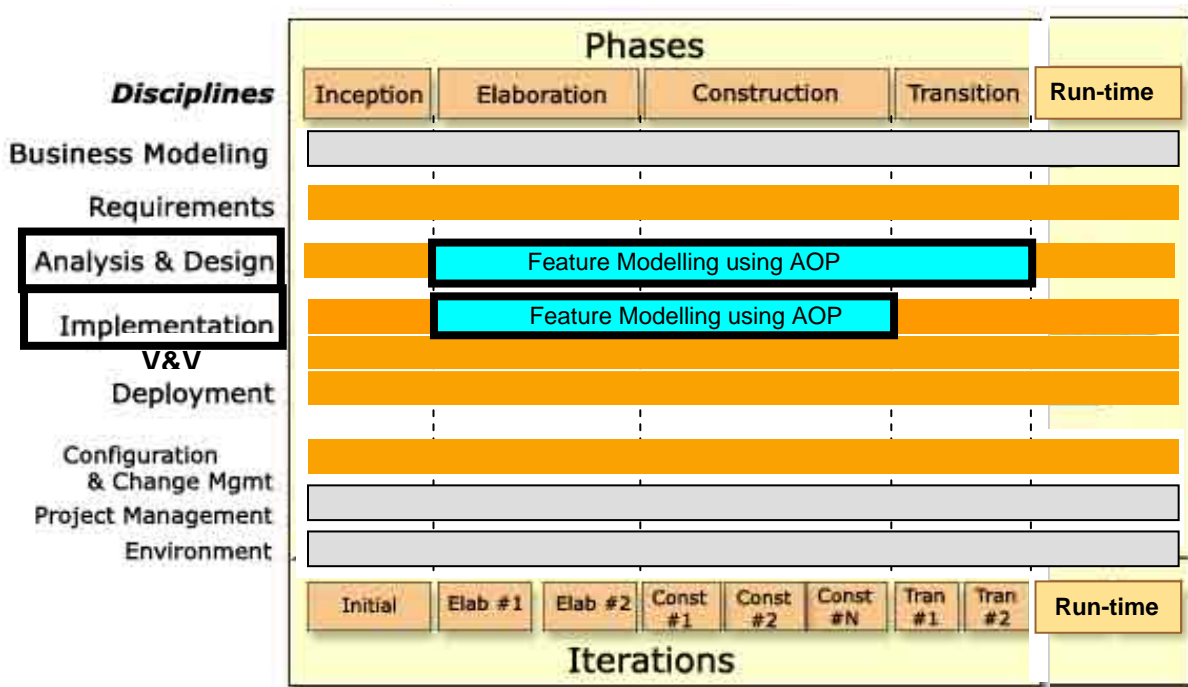


Figure 10: Feature Modelling using AOP

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.2.2 The Bridge Pattern

The Bridge Pattern approach is located on the Analysis & Design, Implementation and Deployment disciplines, whereby the Elaboration phase is targeted to the Analysis & Design discipline, the Construction phase to the Implementation discipline, and the Transition and Run-time phase to the Deployment discipline.

A detailed description of the Bridge Pattern approach can be found in Section 2.4 of Deliverable D2.4.2/D2.5.2.

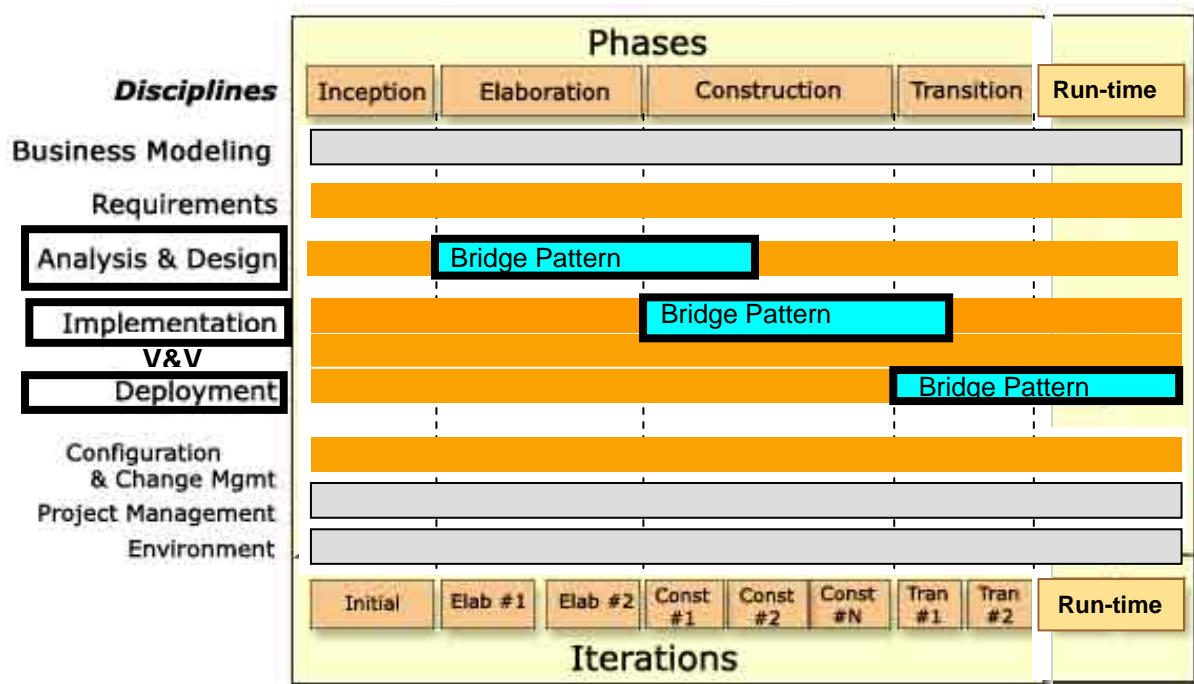


Figure 11: The Bridge Pattern approach

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.2.3 State Transfer approaches

The state transfer approaches are located on the Analysis & Design, Implementation and Deployment disciplines, whereby the Elaboration phase is targeted to the Analysis & Design discipline, the Construction phase to the Analysis & Design and Implementation disciplines, and the Run-time phase to the Deployment discipline.

A detailed description of the State Transfer approaches can be found in Section 2.5 of Deliverable D2.4.2/D2.5.2.

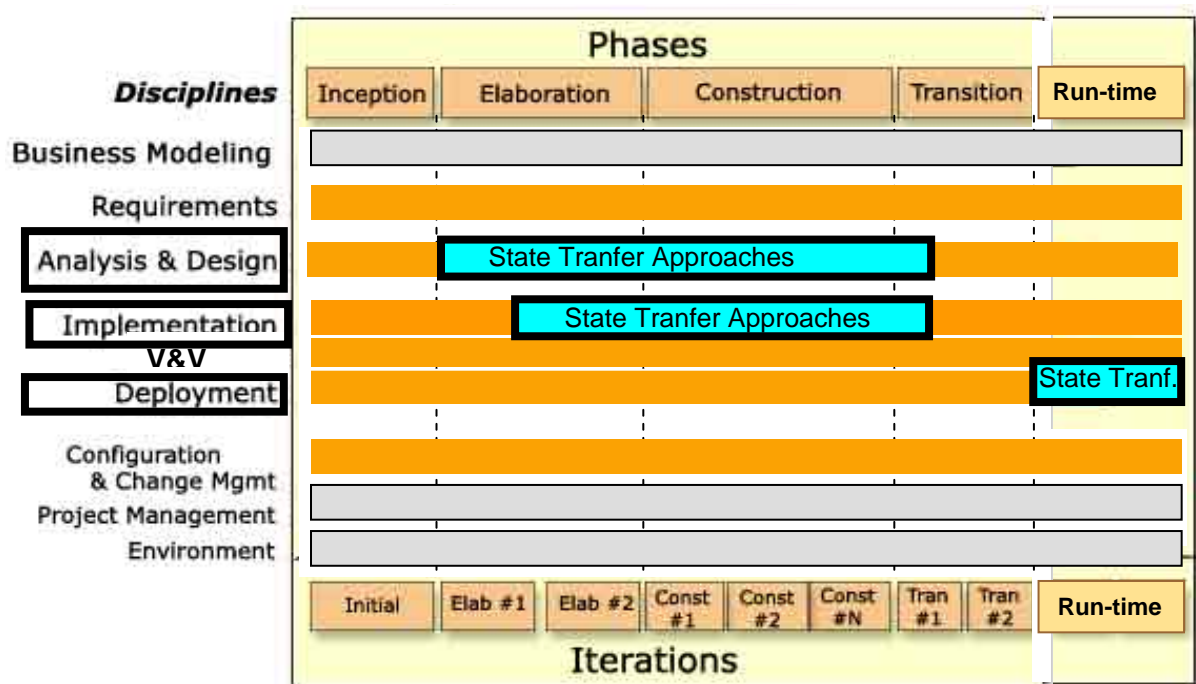


Figure 12: State Transfer approaches

3 Mapping of the WP2 Methods and Techniques onto the EMPRESS Process

3.2.4 Resource-Aware Components and the CRuMB Run-Time Component System

The Resource-Aware Components within the CruMB Run-Time Component System are located on the Analysis & Design, Implementation and Deployment disciplines, whereby the Analysis & Design and Deployment disciplines focuss on the Elaboration, Construction, Transition and Run-time phases while the Implementation discipline focusses on the Construction phase.

A detailed description of the Resource-Aware Component System can be found in Chapter 3 and Appendix A of Deliverable D2.4.2/D2.5.2.

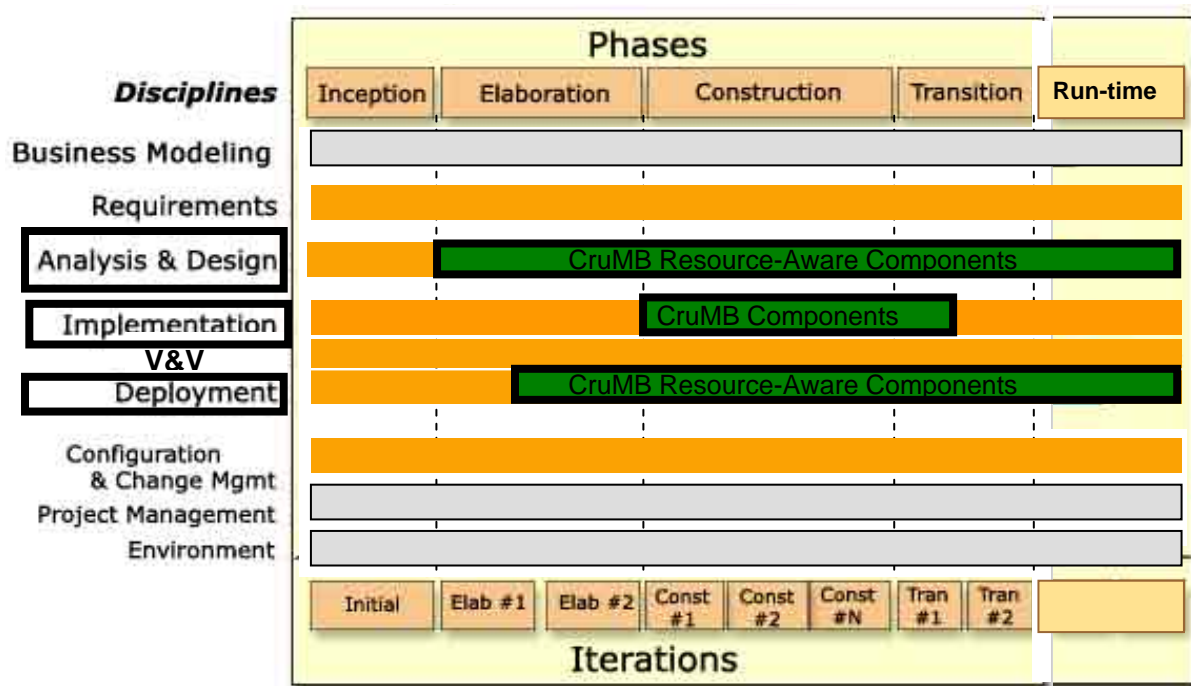


Figure 13: Resource-Aware Components and theCRuMB Run-Time Component System

4 Conclusion

This document presented the combination and integration of the results of the EMPRESS WP2 tasks "Design-time Evolution" and "Run-time Evolution", highlighting the links and interactions between them. This is done using the Unified Process as a reference model.

As such, this document is the link to integrate the relevant results obtained in WP2 with the results of the other EMPRESS work packages, and concentrates on a consolidation of the component architecture for evolution, based on the experiences from building system support and tools and the demonstrator development.

This document serves as an entry point and reference framework to the performed work and the developed methods and techniques of EMPRESS Work Package 2 "Evolution of the Component architecture".