Solution Architecture Template (SAT) Design Guidelines v2.0.0

1 INTRODUCTION

1.1 Purpose of this document

This document explains the purpose of a Solution Architecture Template (SAT) and how to design one. It starts by explaining what the use-cases of the EIRA© are, how an SAT can be used in certain of those use-cases and what the benefits of an SAT are. The different steps on how to create and use an SAT are explained afterwards.

This document gives a small introduction to the EIRA©, the document does not try to explain the EIRA© in depth, nor how the EIRA© can be used, however. Detailed information on the EIRA© is available on the Joinup page[1].

1.2 Target audience

This document targets enterprise/solution architects involved in the design of SAT's based on the EIRA©.

1.3 Further information

This document is part of the EIRA© support series, more information on the EIRA© itself can be found on the Joinup page[2]. This page provides download links to the EIRA© modelled using ArchiMate®, the overview document that explains the EIRA© and tooling support is available via the CarTool©[3]. Finally, the following four SATs (beta versions at the time of the writing of this document) are available as well:

- Open Data SAT (v1.0.0 beta)
- eHI SAT (v1.0.0 beta) - Human Interfaces
- eID SAT (v1.0.0 beta)
- eDelivery SAT (v1.0.0 beta)

2 INTRODUCTION TO SOLUTION ARCHITECTURE TEMPLATES

2.1 The European Interoperability Reference Architecture

The Solution Architecture Templates that are described in this document are based on the "European Interoperability Reference Architecture (EIRA©)", a four-view reference architecture for delivering interoperable digital public services across borders and sectors. It defines the required capabilities for promoting interoperability as a set of architecture building blocks (ABBs). The EIRA© has four main characteristics:

- **Common terminology to achieve a minimum level of coordination**: It provides a set of well-defined ABBs that provide a minimal common understanding of the most important building blocks needed to build interoperable public services.
- **Reference architecture for delivering digital public services**: It offers a framework to categorise (re)usable solution building blocks (SBBs) of an e-Government solution. It allows portfolio managers to rationalise, manage and document their portfolio of solutions.
- **Technology- and product-neutral and a service-oriented architecture (SOA) style**: The EIRA© adopts a service-oriented architecture style and promotes ArchiMate® as a modelling notation. In fact, the EIRA© ABBs can be seen as an extension of the model concepts in ArchiMate®.
- **Alignment with EIF and TOGAF®**: The EIRA© is aligned with the New European Interoperability Framework (EIF)[4] and complies with the context given in the European Interoperability Framework – Implementation Strategy (EIF-IS)[5]. The views of the EIRA© correspond to the interoperability levels in the EIF: legal, organisational, semantic and technical interoperability. Within The Open Group Architecture Framework (TOGAF®)[6] and the Enterprise Architecture Continuum, EIRA© focuses on the architecture continuum. It re-uses terminology and paradigms from TOGAF® such as architecture patterns, building blocks and views.

The EIRA© uses views to model the different aspects of interoperability. Although interoperability almost always involves exchanging data between ICT systems, interoperability is a wider concept and encompasses the ability of organisations to work together towards mutually beneficial and commonly agreed goals. The different views that are used to express this interoperability are "Legal view", "Organisational view", "Semantic view" and "Technical view". An additional view called "EIF Underlying Principles view" contains goals and principles that describe the context in which European public services are designed and implemented.

- **EIRA© view**: The EIRA© consists of several views, including one view for each of the EIF interoperability levels.
- **EIRA© viewpoints**: The EIRA© defines several viewpoints containing a selection of building blocks from the different views, to address the needs of specific stakeholders. SATs should include at least the High-level viewpoint, which provides an introductory view by highlighting the focal ABBs of the SAT.
- **Architecture Building Block**: Based on the TOGAF® definition, an architecture building block is an abstract component that captures architecture requirements and that directs and guides the development of solution building blocks. An ABB represents a (potentially re-usable) component of legal, organisational, semantic or technical capability that can be combined with other architecture building blocks. An architecture building block describes generic characteristics and functionalities. Architecture building blocks are used to describe reference architectures, solution architecture templates or solution architectures of a specific solutions.
- **Solution Building Block**: Based on the TOGAF® definition, a solution building block is a concrete element that implements the required capabilities of one or more architecture building blocks. On the technical view, a solution building block is a specific product or software component.

More information on the EIRA© can be found on Joinup[7] where documentation is available, the model can be downloaded and other supporting material like the CarTool© and other SATs are available for consultation.

2.2 What is a solution architecture template

A Solution Architecture Template (SAT) is a specification extending the EIRA© providing support to solution architects in a specific solution domain in the form of a template that can be used to design related solutions. An SAT contains the following elements:

- A motivation in the form of principles and requirements.
- A goal and a description of the supported functionalities.
- A sub-set of the EIRA© core Architecture Building Blocks (ABBs) covering the four EIF layers.
- A set of specific ABBs extending EIRA©'s views enabling specific functionalities to be provided by implementations derived from the SAT.
- The interoperability specifications of selected ABBs and a narrative for each EIRA© view.

[7] https://joinup.ec.europa.eu/asset/eia/

2.3 Benefits of an SAT

The benefits of a SAT are the following:

- Provides architects with a common approach to cope with a specific interoperability challenge. It also places the focus on the key-points you need to consider when building a solution in a specific problem area.
- An architect can create a solution architecture by mapping existing Solution Building Blocks (SBBs) to an SAT, based on the interoperability specifications that are provided. This is done by realising the ABBs identified in the SAT by specific SBBs.
- When an architect creates an SAT, he/she can define the interoperability specifications for the SAT's ABBs and moreover recommend specific SBBs so that the selection of specific solutions is facilitated and is ensured to have more interoperable results.
- An SAT can be created within and across the different views of the EIRA©. An SAT can then support architects specialised in different interoperability levels.

2. PROPOSED SOLUTION

This dissertation proposes a new method for object-oriented framework design and instantiation.

In [19] Roberts and Johnson state that "Developing reusable frameworks cannot occur by simply sitting down and thinking about the problem domain. No one has the insight to come up with the proper abstractions." They propose the development of concrete examples in order to understand the domain. The design strategy presented here is quite similar, analyzing concrete applications as viewpoints [8, 6] of a domain and deriving the final framework from the analysis of these viewpoints.

The approach to framework design is based on the idea that any design can be divided into two parts: the kernel sub-system design and the hot-spot sub-system design. The kernel sub-system design is common to all the applications that the framework may generate, and the hot-spot sub-system design describes the different characteristics of each application that can be supported by the framework. The hot-spot sub-system uses the method and the information provided by the kernel sub-system and may extend it.

The kernel structure is defined by analyzing the viewpoints design representations to produce a resulting design representation that reflects a structure that is common to all chosen viewpoints. This part of the design approach is based on a domain-dependent semantic analysis of the design diagrams to elicit the common features of the framework design structure, and is formally described in this dissertation.

The elements that are not in the kernel are the ones that vary for each application, and depend on the use of the framework. These elements define the framework hot-spots [17, 20] that must be adapted to each related application. We defined new relationship in object-oriented design, called the hot-spot relationship, to specify all the hot-spots in the system.

The semantics of this new relationship is given by the design patterns essentials [18]. This implies that the hot-spot relationship is in fact a meta-relationship that is implemented through a design pattern that is generated taking into account the hot-spot flexibility requirements. The hot-spot cards guides this generation process, providing a systematic way for generating design patterns based on flexibility properties.

The most common way to instantiate a framework is to inherit from some abstract classes defined in the framework hierarchy and write the code that is called by the framework itself. However, it is not always easy to identify which code and where it should be written since frameworks class hierarchies can be very complex, especially for non-expert users.

Therefore, the "common" instantiation process is not always the ideal way to describe a particular application. This happens because of several facts:

- the language used to build and use the framework is a wide spectrum language, where it is not always easy to express the user intentions;
- the complexity of framework class hierarchies and the difficulty of finding the points where the code should be written, that are the framework hot-spots or flexible points.

The instantiation method proposes a different process, where a particular application is described by domain specific languages (DSLs) [12] that are designed precisely for the framework domain. In this way, the technique basic idea is to capture domain concepts in a DSL, which will help the framework user to build code in an easier way, without worrying about implementation decisions and remaining focused on the problem domain. The specification written in the DSL is translated to the framework instantiation code through the use of transformational systems [16, 4].