


Managing ICT Infrastructure Architecture knowledge

A “Case Study” for Semantic MediaWiki

Amsterdam, 2010-09-19
jan.schoonderbeek@sogeti.nl



This presentation is a case study of the use of Semantic MediaWiki for knowledge management.

Well, “case study”? This presentation is a showcase, based on personal experience, but without in-depth methodical investigation.

Program

- **Introduction**

- Presenter, presentation goals

- **Infrastructure Architecture**

- What is it, why do companies need it
- DYA|Infrastructure, the methodology

- **DYA|Infrastructure Repository (DIR)**

- Overview
- Noteworthy uses of SMW

- **The Information Model**

- Why do we need it
- How to create it

- **Jan Schoonderbeek**

- Infrastructure Architect at Sogeti Nederland B.V.
- "Saruman" on #semantic-mediawiki

- **Presentation:**

- 25 minutes (12 slides)
- Interactive (ask away!)
- Goal:
 - to show how we use SMW to manage knowledge
 - to introduce the concept of an information model

Infrastructure architecture

- **The problem:**

- Lots of concurrent projects, each demanding their own little bit of infrastructure
- Lack of reuse, lack of coherent development, lack of standards

- **The solution: Infrastructure architecture**

- Frames ICT infrastructure development
- Promotes reuse of ICT facilities and use of standards
- Separates functionality from technical components (!)

- **The beneficiaries:**

- Infrastructure designers, Service managers, Enterprise architects, Operations



This presentation is not the right platform to explain ICT infrastructure architecture in full, but to appreciate the knowledge system built for it, we quickly glance over the essence of infrastructure architecture anyway.

Information overload?



A little rest point, for when the afternoon is getting too long...



An empty slide, to let all the new information from the last slide sink in... After all this presentation is the last one on the second day..

- **Methodology for (de)composition**
- **Metamodel for infrastructure facilities**
 - 5 "dimensions"
 - Suggested set of artefacts
- **Accompanied by best practices, guidelines and architecture products**
- **Extremely boring for all but infrastructure specialists**

A short characterization of the DYAIInfrastructure methodology, which serves as the backbone of the knowledge management system

MW.BV.Identity Validation.SAP



Page maturity
This page has matured over **2 (years)**

MW.BV	Identity Validation SAP	Version: 0.21	
Document type	Building block Variant	Owner: S.A.U. Jansick	

Variant Coordinates

This Building Block Variant...
belongs to Working Area: **Middelware (MW)**
is a variant of Building Block Type: **Identity validation**
is intended for use in Environment: **Data Centre - Public cloud**

<https://dya-knowledge.sogeti.nl/dir>

The purpose is to authenticate a user that requests access to a SAP application against the proprietary SAP® Authentication Application either in a local solution (i.e.) or a central (cloud) solution (SUA).
If Single Sign-On is required, CUA (and TAN) are required. The user base is also a determinant between UA and CUA. If the user base is internal (SmartMart own or hired personnel) either the UA or CUA option can be used. If the user base is external or a mix of internal and external UA is used, Single Sign is then only possible if an exception is granted for the external users.

Characteristics & indications of use case



This slide shows the top half of a single page in the repository; if anyone is curious, the content of the repository is publicly accessible at the given URL

Customer implementation: KLM/AirFrance



Main Page

This is the architecture wiki describing the general infrastructure architecture of KLM/AF. It is based on an application of UTA/Infrastructure, and is decomposed and modelled using the UTA/Infrastructure Building Blocks Model. The methodology is described here:

To see for yourself how the Reference Architecture and Product Catalogue are constructed for KLM/AF, go ahead and use the Sidebar to the left to explore the infrastructure architecture repository. In the process, you will be able to explore and witness the working of the Building Blocks Model in this pioneering architecture implementation.

A quick overview of all classes of pages included in the wiki can be found from the categories listing.

Create an architecture artefact

The following forms are available to create (if you have sufficient permissions) or view architecture artefacts:

- Element form
- Building Block Variant form
- Building Block Type form
- Pattern Type form
- Environment form
- Working Area form

If you want to create a new page from scratch:

Statistics per 16-02-2010

This repository currently has:

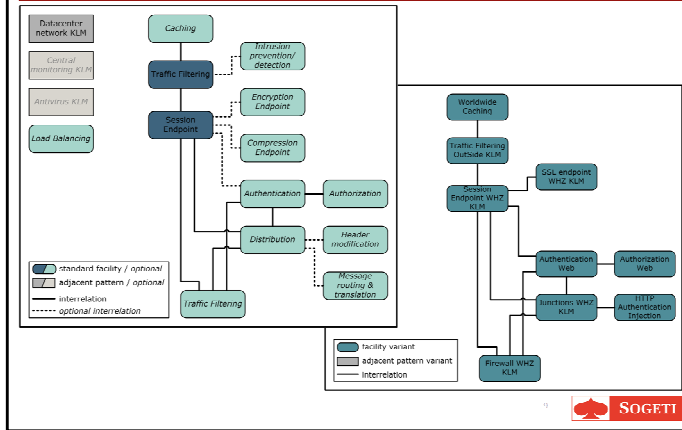
- 5 Working Areas defined
- 14 Environments under these Working Areas
- 71 Building Block Types in these 5 environments
- 58 Building Block Variants based on these Building Block Types
- 20 Elements that are used in the Building Block Variants
- 23 Pattern Types that define common infrastructure solutions
- 14 Pattern Variants that realize these solutions

Specific group related numbers:

- KLM has 21 Building Block Variants and 6 Pattern Variants
- AF has 2 Building Block Variants and 1 Pattern Variants
- 31 artefacts are used

This screenshot shows the KLM/AF infrastructure repository, which is *not* accessible from the Internet.

Creating infrastructure patterns

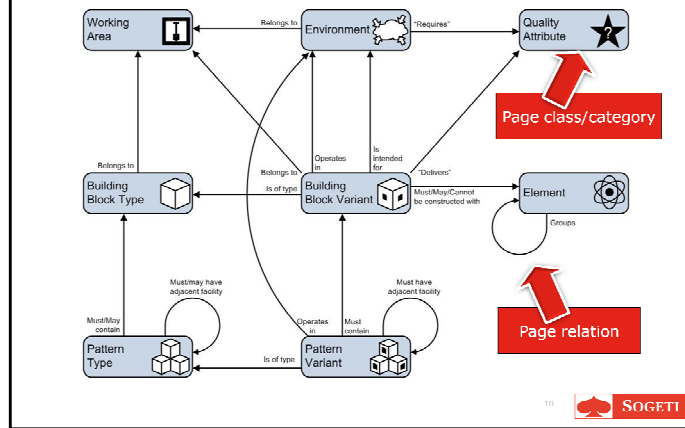


This is basically what the repository is for: creating patterns of infrastructure functions.

The example on the left is a “pattern type” (generic mould) that can be used to create application access protection facilities

The picture on the right is a “pattern variant” (a more specialized version of the generic mould) that describes the facility that provides application access protection to the KLM webfarm.

Knowledge system build-up



The repository contains different types of architecture artefacts. This picture illustrates the interrelations between 8 such types. Each blue rectangle is a class of architecture artefact, which is recognized as an SMW Category. The arrows indicate possible semantic relations. Thus we can see that in the Repository, one can find that “this Building Block Variant” “Belongs to” “that Working Area”.

What SMW brings to the table here...

- **Forms with autoselection (using SF)**
- **Automatic, up-to-date lists**
- **Up-to-date statistics**
- **(programmed) inheritance**
- **Checks based on reasoning**
- **... and more features to be unleashed in the future**

Forms: e.g. Building Block Variant creation

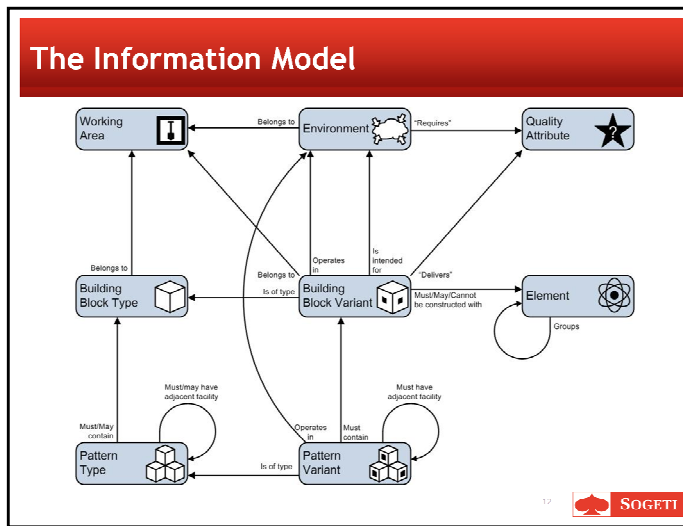
Lists: e.g. Building Block Types

Statistics: e.g. front page counters

Inheritance: e.g. Building Block Type icon

Checks: e.g. Pattern Variant check on Building Block Variant placement in an Environment

The Information Model



OK so THIS is – in my opinion, anyway – a representation of an information model. Why do we need one?

-To keep track of the relations that we have available, e.g. for queries

-To help us consider the completeness of our representation of the recorded knowledge

Having an information model is slightly more important for abstract knowledge domains (e.g. wiki's that are "dressed" as on-line applications) than for concrete knowledge domains (e.g. wiki's that record information about a specific topic, like game characters, towns, graves etc), but I believe they ALWAYS help!

How to create an Information Model

- **Determine the limits of your knowledge domain**
- **Consider which artefacts can...**
 - .. "live" by themselves on a separate page
 - .. be considered to "be" something

→ **Create categories to group the artefacts**
- **Consider the properties each artefact has**
- **Record your information model**
 - ... to keep things clear for yourself
 - ... to advertise the model to your users
- **Go with the flow of setting up your wiki**

Ad 1) Do not attempt to annotate everything and then some. Limit the domain you'll model, at least initially

Ad 2) It can be hard to determine if information belongs in a section of a page, or should be put in its own separate page, and then referenced from the first page. It can also be hard (at least initially) to see if some aspect is a class or a relation.

Ad 3) Don't "proptize" too much information. It is doubtful that a semantic wiki about beer needs a property for the middle name of the owner of the factory that delivers the paper to the printer of the adhesive labels for the beer bottles. On the other hand, don't be afraid to introduce "artificial" properties that help your knowledge system, e.g. the DIR's page version number, or the ordinal number for a quality attribute level (Availability "high" corresponds to Availability ordinal "30")

Ad 4) The picture on the last slide shows how a graphic representation can help you keep track of 8 classes and 16 relations.

Ad 5) Don't overdo it! Don't try to write out the whole information model in full before starting your wiki (but conversely, don't just make everything up as you go along, only recording it at the last possible moment). Think carefully about one aspect, structure it semantically, implement it in a mockup page, then create the templates and forms, and update your information model

