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
Introduction

- **Jan Schoonderbeek**
  - Infrastructure Architect at Sogeti Nederland B.V.
  - "Sanman" 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
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.

- 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 (1)

- The beneficiaries:
  - Infrastructure designers, Service managers, Enterprise architects, Operations
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.
A short characterization of the DYA|Infrastructure methodology, which serves as the backbone of the knowledge management system.

- Methodology for (de)composition
- Metamodel for infrastructure facilities
  - “dimensions”
  - Suggested set of artefacts
- Accompanied by best practices, guidelines and architecture products
- Extremely boring for all but infrastructure specialists
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
This screenshot shows the KLM/AF infrastructure repository, which is not accessible from the Internet.
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.
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”.

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
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!
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 “propertize” 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.