A Technical Perspective on Use-Case-Driven Challenges for Software Architectures to Document Study and Variable Information

IASSIST 2013
29.05.2013

Matthäus Zloch
GESIS, Germany
matthaeus.zloch@gesis.org

Thomas Bosch
GESIS, Germany
thomas.bosch@gesis.org
boschthomas@blogspot.com

Dennis Wegener
GESIS, Germany
dennis.wegener@gesis.org
Outline

• What has already been said

• Challenges for MISSY Software Developers
• MISSY Software Architecture
• Implementation of DISCO
• Persistence Strategies
Thomas Presentation

- General information about MISSY
- Next generation MISSY
- Software architecture overview
- Presentation layer and MISSY use cases
- Business logic
  - data model
  - DDI-RDF Discovery Vocabulary
MISSY for Software Developers

CHALLENGES AND REQUIREMENTS
Requirements to Software Developers

• Focus lies on software reusability
  • must be stable and reliable
  • API must be clean and easy to extend
• Flexible Web Application Framework and modern architecture
  • Service-oriented
  • Use of Semantic Web technologies
• Complex data model to represent use-cases (seen in previous presentation)
Requirements to Software Developers

• Define and implement a common data model and

• Different Persistence Strategies

• Creation of an abstract framework and architecture
  • Should be well designed to be able to be extended and reusable
  • Available as open source software
  • Independent of end-user system
Design Goals for the MISSY

SOFTWARE ARCHITECTURE
Software Architecture – Design Goals

• Separation of
  • Model, i.e. concepts and real life objects, that represents the use case
  • (Physical) Storage mechanisms
  • Logic that controls and provides services to manipulate the data
  • The representation of information itself

• The key is to have logically separated parts, where people might work independently but collaboratively

• Creation of a reusable and extendable abstract API
Software Architecture

- State-of-the-art technologies to develop software
  - Multitier architecture
  - Model-View-Controller (MVC-Pattern)
  - Maven Projects + Modules

- Multitier architecture separates the project into logical parts
  - Presentation, application processing, data, persistence, …
MVC – Interactions

Model manipulates

View accesses

Controller controls

Model manipulates
View Technologies

Controller

Model

View

controls

manipulates

accesses

<FreeMarker>

The Apache Velocity Project
http://velocity.apache.org/

JavaServer Pages

Groovy

...
Data Model

Controller \(\rightarrow\) Model

- controls
- manipulates

View

- accesses

<FreeMarker>

<ddi>

The Apache Velocity Project
http://velocity.apache.org/

JavaServer Pages

Groovy
Missy Technologies

![Diagram of Model-View-Controller (MVC) architecture with arrows indicating manipulates, controls, and accesses.]
Implementation of

THE DATA MODEL
Data Model

- DDI-RDF Discovery Vocabulary DISCO
  - designed for the discovery use-case
  - provides object types, properties and data type properties designed for discovery use-case

- We use DISCO as the internal data model
  - Implemented in Java
  - Maps all object properties available

- Subclass relationships through Java native inheritance
Extendible Data Model

- DISCO does not cover all use cases
- Projects may have individual needs
- DISCO-model objects may be extended
Extendible Data Model

• DISCO does not cover all use cases
• Projects may have individual needs
• DISCO-model objects may be extended

Provide this as an API!!
disco-model api
Multilevel-Model

disco-model api

gesis-model api

project1-model

project2-model
Data is stored via

PERSISTENCE STRATEGIES
Persistence-Layer – Strategies

• The application itself does not need to know how the data is (physically) stored
  • Methods are provided to access and store objects through data access objects
  • Actual implementation is “hidden” to the upper layers

• A strategy is an implementation of the actual type of persistence or physical storage, respectively
  • e.g. DDI-L-XML, DDI-RDF, XML-DB, Relational-DB, etc.
Persistence-Layer – Strategies

- The application itself does not need to know how the data is (physically) stored
  - Methods are provided to access and store objects through data access objects
  - Actual implementation is “hidden” to the upper layers
- A strategy is an implementation of the actual type of persistence or physical storage, respectively
  - e.g. DDI-L-XML, DDI-RDF, XML-DB, Relational-DB, etc.
Persistence-Layer – Strategies

- The application itself does not need to know how the data is (physically) stored
  - Methods are provided to access and store objects through data access objects
  - Actual implementation is “hidden” to the upper layers

- A strategy is an implementation of the actual type of persistence or physical storage, respectively
  - e.g. DDI-L-XML, DDI-RDF, XML-DB, Relational-DB, etc.

- Due to performance:
Persistence-Layer – Strategies / Modules

- disco-persistence-api
  - Defines persistence functionality for model components regardless of the actual type of physical persistence

- disco-persistence-relational
  - Implements the persistence functionality defined in disco-persistence-api with respect to the usage of relational DBs

- disco-persistence-xml
  - Implements the persistence functionality defined in disco-persistence-api with respect to the usage of DDI-XML

- disco-persistence-rdf
  - Implements the persistence functionality defined in disco-persistence-api with respect to the usage of the disco-specification
Persistence-Layer – Strategies / Modules

• **disco-persistence-api**
  • **Defines** persistence functionality for model components regardless of the actual type of physical persistence

• **disco-persistence-relational**
  • **Implements** the persistence functionality defined in disco-persistence-api with respect to the usage of relational DBs

• **disco-persistence-xml**
  • **Implements** the persistence functionality defined in disco-persistence-api with respect to the usage of DDI-XML

• **disco-persistence-rdf**
  • **Implements** the persistence functionality defined in disco-persistence-api with respect to the usage of the disco-specification
Hibernate Example
Inheritance of Properties

from DDI-Identifiable
Inheritance of Properties

from disco-api
Declaration of own Properties
Missy Project API – Modules

- **presentation**
  - missy-editor-web

- **business**
  - missy-editor-core
  - disco-model
  - missy-model

- **persistence**
  - disco-persistence-api
  - disco-persistence-relational
  - missy-persistence-api
  - missy-persistence-relational
Towards a DDI Backend Architecture
Sharing DDI-related Software Modules

Mathias Ziech, Thomas Bosch, Dennis Wegener
(mathias.ziech, thomas.bosch, dennis.wegener)@gesis.org

Data Model (DDI Discovery Vocabulary + MISSY)

Microdata Information System (MISSY)
- Study documentation on variable level
- Studies: German microcensus, EU-SILC, EU-LFS, EVS, ...

Data models
- DDI Discovery Vocabulary as a basis
- Project-specific data models

MISSY Web – Variable Details

Multitier Software Architecture
- Reusable modules
- Usage of diverse software development patterns (e.g., MVC, strategy, DAO)
Thank you for your attention

**Matthäus Zloch**
Team Architecture  
GESIS, Germany  
matthaeus.zloch@gesis.org

**The Missy Project**
http://github.com/missy-project