Architecture Matters

Erik Wilde, School of Information, UC Berkeley

March 7, 2011

http://dret.net/netdret/publications#integrative-spring11

Contents

- Abstract .................................................................................................................. 2
- Precious Snowflakes ............................................................................................... 3
- Architectural Challenges ......................................................................................... 4
- 1 Examples
  - Synchronization .................................................................................................. 6
  - So you want to be a platform? .............................................................................. 7
- 2 SOA Metamodels
  - What's your Service? .......................................................................................... 9
  - The Great SOA Debate ....................................................................................... 10
  - Designing A System ............................................................................................ 11
  - Designing Systems ............................................................................................. 12
  - Designing Ecosystems ........................................................................................ 13
  - Applying Established Patterns ............................................................................ 14
  - Focusing on the Service ..................................................................................... 15
  - Extending Existing Models ................................................................................ 16
  - Tiled Feed Example ............................................................................................. 17
- 3 Data Metamodels
  - The 5 Metamodel Realms .................................................................................... 19
  - Picking a Metamodel ........................................................................................... 20
- Thanks! Questions? .................................................................................................. 21
Abstract

A good business model and good service design are necessary requirements to implement a successful information-oriented project or business, but implementation issues still can have a great impact on the chances of success. Technical design and implementation are not a mechanical process of defining an abstract model, and then pressing the "generate system code" button. Important architectural issues cover questions of data model design and implementation (which metamodel is used, what are the planned extensibility and openness points, how well is extensibility and openness planned and supported in both the data model and the code base), service design and implementation (what is the service implementation technology, what are the implicit assumptions about technologies that have to be supported by any user of the service), and platform strategy (what is the server-side platform, what is the client-side platform, what are the exit strategies for those platforms).

Precious Snowflakes

- Many applications are not as unique as you might think
  - reusing design patterns and technologies is a good idea
  - avoiding reuse should be justified by good reasons
- Take the Web's HTML as an inspiring example
  - HTML is not all that great as a document format
  - it was good enough as a container for a lot of useful content
  - the network effect far outweighs its functional shortcomings
- Simplicity and easy applicability are valuable
  - simplicity means a lower barrier to entry
  - easy applicability means higher chances to create network effects
Architectural Challenges

- **Effectiveness over Efficiency**
  - design should result in *effective implementations*
  - *efficiency* depends on design (and thus is one important constraint)
  - *flexibility* and *openness* may be at least as important
- **Refactoring** services and data models is hard
  - hitting the sweet spot between overspecialization and overgeneralization is hard
- **Design has become much harder than it used to be**
  - the platform landscape is changing rapidly
  - the competition landscape is changing rapidly
  - it is very likely that interoperability will become important

Examples

**Synchronization**

- Offline-capabilities can be essential for applications
- Many (even mobile) application have terrible offline support
  - not easy to do well but easy to get wrong
- Synchronization is very tightly coupled with the data model
  - things must have a global and stable identity
  - things must have ways how to get created/updated/deleted
  - there must be some way for clients to "query" the update stream
- What are the characteristics of synchronization?
  - few updates (manual reports) or many updates (sensor readings)?
  - complete synchronization or only with a limited horizon?
  - are "overwrites" possible and can they be concurrent?
  - how critical is it to prevent outdated views?
So you want to be a platform?

- Have at least a few possible platform users
  - what are the constants, what are the variable parts?
  - will applications need freedom in adding structured data?
  - if applications can add structured data, can they query it?
- Consider all possible target users of your platform
  - what are the technological requirements to become a client?
  - what is the knowledge required to become a client?
  - do you expect few users or do you expect many users?
- "Eat your own dog food"
  - build your own application/interface using your own API

SOA Metamodels

What's your Service?

- "Building Apps" or "Designing Services"?
  - apps provide access to services (focus on information-intensive services)
  - services provide the substance apps can build on
- The "app landscape" changes faster than ever before
  - Web technology is developing quickly with trends such as HTML5
  - Web clients are developing rapidly with new device classes such as smartphones and tablets
  - the platform landscape changes with non-Web platforms such as iOS and Android
  - making choices in that spectrum is hard and likely to change
- Service Design is what really matters
  - what is the (abstract) service you are delivering?
  - what is the (concrete) service you are implementing?
  - the "touch point" is important but not the only thing
The Great SOA Debate

- **Service Oriented Architecture (SOA)** is a very abstract term
  - how to identify “components” that can be recombined and replaced?
  - how easily can components be changed/switched if that becomes necessary?
  - how easy can the **component landscape** be changed?
- "Better an A team with a B plan than a B team with an A plan"
  - chances are that the original plan needs some adjustment
  - how easy is it to change to a different plan without a complete redesign?
  - maybe parts that originally were internal now should be externalized
- Startups have problems similar to large corporations
  - quickly changing environment and constraints and requirements
  - ability to outsource and insource according to business needs

Designing A System
Designing Systems
Designing Ecosystems

Ecosystem Applications

Resource Ecosystem

Applying Established Patterns
Focusing on the Service
Extending Existing Models

Tiled Feed Example
Data Metamodels

The 5 Metamodel Realms

- Relational
  - SQL
- Trees
  - XML, SGML
- Generalized Graphs
  - RDF
- Key/Value
  - NoSQL, HTML5 Web Storage
- Vectors
  - bag of words

Picking a Metamodel

- Avoid the "framework trap"
  - think long and hard before picking a particular framework
- Moving between metamodel realms is hard
  - design and implementation have to be changed
- Constraints should help picking a realm
  - questions: granularity, openness, extensibility, cohesiveness, perspectives
- Simpler metamodels are not necessarily simpler solutions
- Service design and metamodel design depend on each other
Thanks! Questions?