Architectures for Different Purposes

L07 PA1308

Mikael Svahnberg[1]

[1]Mikael.Svahnberg@bth.se
School of Computing
Blekinge Institute of Technology

September 25, 2012

Architectures for different purposes

- Regular desktop applications
- Embedded applications
- Enterprise applications
- Applications for Android / IOS
- Cloud applications
- Service-Oriented Architectures (SOA)
- Software Ecosystems
- . . .

What can differ?

- Instantiation into different viewpoints?
  - Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
  - Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
  - Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
  - Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
- Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
- Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
- Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
- Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
- Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
- Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
- Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
- Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

What can differ?

- Instantiation into different viewpoints?
- Trivial, the views will differ between each application anyway.
- Factors, Issues and Strategies?
- Also trivial, for the same reason.
- The *importance* of certain factors (e.g. the importance of certain quality requirements).
- *Typical choices* of strategies for a particular domain.
- Typical choices of architecture styles for a particular domain (may be subordinate to the aforementioned)

Embedded Applications I

This is a large and diverse field, and there are many quality requirements that may be in focus for different applications. However, there are some overall constraints that hold true for *most* applications in this domain:

- **Hardware cost**
  - Keep memory footprint low
  - Keep CPU usage low
  - Optimise for wear and tear
- **Hardware availability for entire product life expectancy**
  - Keep memory and CPU usage low.
  - Cull system regularly to remove stuff that is no longer needed.
  - Low-cost growth mechanisms.

Embedded Applications II

- **Testability**
  - Testable software – the usual stuff
  - Testable hardware – system test software, test harness, etc.
  - Report error states (e.g. through flashing diodes).
- **Reliability**
  - Error detection
  - Error recovery
  - Report error states visibly (e.g. on-line, flashing diodes)

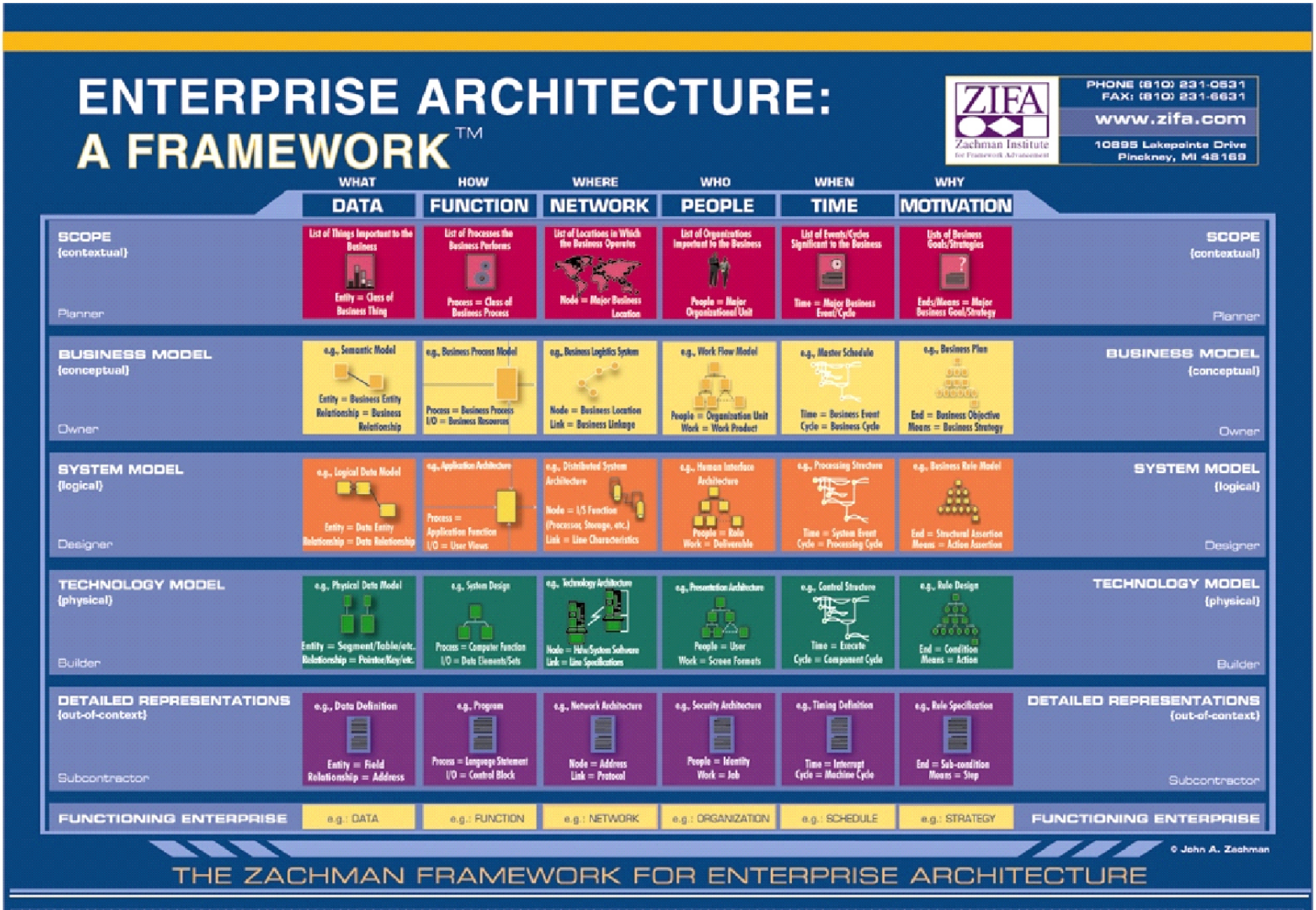
Embedded Applications III

- **Energy consumption**
  - Low-effort computing
  - Reduce display time and display size (if any)
  - Powersave modes
  - Lazy evaluations – deferred processing
- **Network communications**
  - Standard communications platform
  - Robust transfers (e.g. in outdoor environments)
  - Operate by “dead reckoning”

Enterprise Applications I

- Enterprise architectures only has very little to do with software architecture – and yet it has *everything* to do with the software architecture.
- Organisational, Technological, and Product factors can be considered subsets, or limited views, of the enterprise architecture.

Enterprise Applications II: Zachman Framework



Enterprise Applications III: List of Concerns

- Persistent Data[1]
- Large amounts of data
- Large scale concurrent access
- Many data views (user interface screens)
- Need to integrate with other enterprise applications
- Multiple interpretations of data (conceptual dissonance)
- Complex business logic rules (business “illogic”)
- Various types of enterrpise systems
  $\{B, C\}2\{B, C\} \wedge B, C \in \{s, m, l, xl, xxl\}$

[1] M. J. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston MA, 2003.

Enterprise Applications IV: Typical Architecture Styles

- Function-centric (Transaction script)
- Domain concept-centric (Domain model)
- Data representation-centric (Table module)

. . . I could go on, but the book (Fowler 2003) is rather thick and full of patterns that dig deeper and deeper into the application.

Android/IOS Applications

Somewhere between embedded and desktop applications. Any number of quality concerns may be relevant for any application. The platform itself imposes some concerns.

Hardware:

- Restrict battery usage
- Small screen
- Unorthodox input methods (e.g. thumb)
- Not the fastest CPU, restricted RAM.
- Variety of hardware available on a particular phone model.

Software:

- Interruptible applications (e.g. for phone calls)
- Interoperable applications
- Reuse wherever possible, yet enable customisation.
- Intuitive user experience that supports how users operate their handheld device (e.g., avoid deep meny hierarchies).
- Varity of services available on a particular phone.

A/IOS Applications: Addressing Concerns

- Battery usage
  - Event-driven applications (BroadcastReceivers and IntentFilters)
- Interruptible applications, “flat” user interfaces
  - Loosely connected applications
  - Event-driven applications
  - Application as a set of screens, each screen a separate process
  - Separate Activities (interaction-based) from Services (runs in the background)
  - Persistent storage as a system service
- interoperable applications, reuse wherever possible, variety of services available, yet customisable
  - Late binding
  - Loosely connected applications
  - Event-driven applications
  - Common communication platform: Intent and IntentFilters.
  - Separate Activities and Services from ContentProviders.

Cloud Applications

- The concept of a cloud application is simple: It is essentially a client-server solution, where rather than maintaining the server yourself, you rent virtual servers from a cloud vendor.
- *One* definition[2] of a cloud service
  - The service is accessible via a web browser or web services API
  - Zero capital expenditure is necessary to get started
  - You pay only for what you use as you use it.
- Another definition [3]
  - Pooled Resources, Virtualisation, Elasticity, Automation, Metered Billing

[2] G. Reese, *Cloud Application Architectures*, O'Reilly, 2009.
[3] Rosenberg et al., *The Cloud at your Service*, Manning Publications co., 2011.

Levels of Cloud Services

- Software as a Service (SaaS)
  - e.g. Google Docs, Yahoo!, SalesForce.com, Valtira, etc.
- Platform as a Service (PaaS)
  - e.g. Google App Engine, Microsoft Azure, etc.
- Infrastructure as a Service (IaaS)
  - e.g. Amazon Elastic Compute Cloud (EC2), Microsoft Azure, RackSpace, etc.
- ...

Factors that “push” you towards the cloud

- Transference – Move your on-site solution as-is to the cloud for e.g. economic reasons.
  - Challenges: Setting up a similar environment in the cloud as you have locally.
- Internet Scale – Scaling up to handle more users.
  - Challenges: Database design may become a bottleneck.
- Burst Compute – Large swings in capacity requirements.
  - Challenges: Strategy for load balancing, database access.
- Elastic Storage – Scaling up to handle (much) more data.
  - Challenges: need also to consider where the data is processed.

Challenge: Internet Scale

Issue:

- Your database's working sets are too large
- Too many writes

Solution:

- Partition the data (Sharding)

Challenge: Cloudbursting

Issue:

- Occasional peaks of traffic that pushes infrastructure over its capacity

Solution:

- Use on-demand capacity (Cloud) for the peaks
- Load-balancer that divides work between in-house servers and cloud servers
- Render static data views

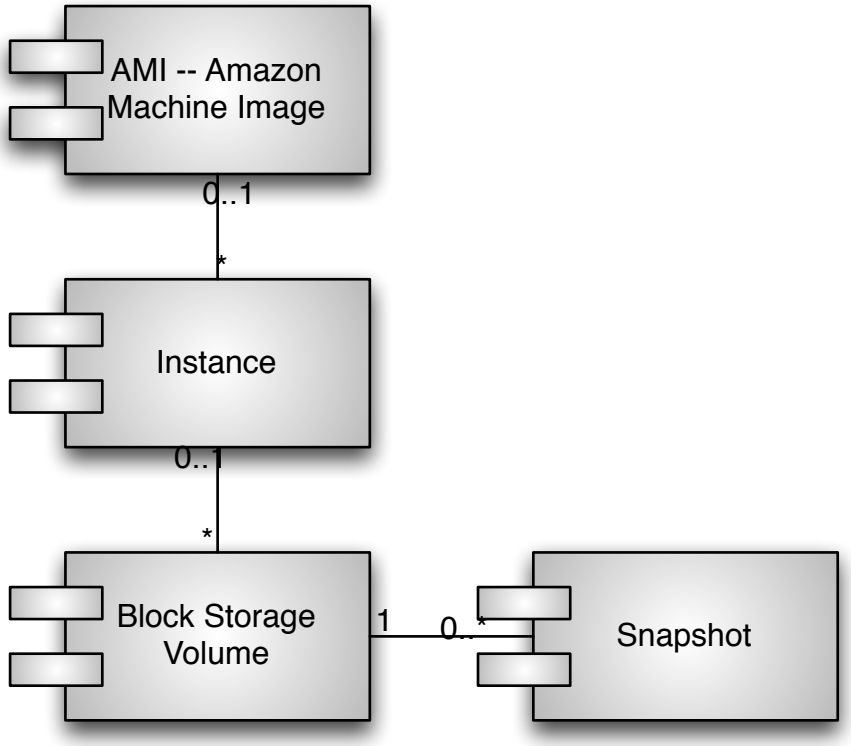
An Overview of Amazon Web Services (AWS)

- Amazon Elastic Cloud Compute (Amazon EC2)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon CloudFront
- Amazon SimpleDB

Please Note

Amazon Web Services is not the only cloud platform available. However, it appears to be among the most popular.

Basic setup of EC2



- Machine Image + Volume Snapsot = Your mold to create instances
- Instance (with an “Elastic IP”) your running instances.
- Instances may have a block storage volume mounted (like a regular HD)

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Challenges

- Avoid allowing Amazon access to the data
  - Solution: Encrypt it at home and load on startup
- Persistent storage that survives even if all your instances go down
  - Solution: Block Storage Volumes? Amazon S3
- Synchronisation and communication between instances
  - Solution: Amazon SQS, convert memory locks to a lock on e.g. a specific row in a DB.
- Startup and mount sequence makes it difficult to do an automatic boot
  - Solution: Hacking startup scripts may be needed
- A whole bunch of other cloud-specific concerns, such as that some governments require data about its citizens to be stored within national borders.
  - Solution (to this particular issue): Amazon CloudFront

Software Ecosystems

- Currently mostly vapourware.
- However, the challenge is real enough.
- Basic challenge:
  - You have a layered architecture
  - You wish to open up for third party development (e.g. plug-ins)
  - Vertical Functionality vs Horizontal Layering
  - How to maintain architecture decisions?
  - How to incorporate the work of others?

In other words

You have a full blown *software ecosystem* where the software organically grows without your control.

Software Ecosystems

- Currently mostly vapourware.
- However, the challenge is real enough.
- Basic challenge:
  - You have a layered architecture
  - You wish to open up for third party development (e.g. plug-ins)
  - Vertical Functionality vs Horizontal Layering
  - How to maintain architecture decisions?
  - How to incorporate the work of others?

In other words

You have a full blown *software ecosystem* where the software organically grows without your control.

Ecosystems Concerns

- Support feature-oriented third party development
- Trust in third party developed code
- Easy integration of third party developed code
- Maintain reusability of system core, avoid redundant code
- Avoid abdicating from strategic product management!

Ecosystems Concerns

- Support feature-oriented third party development
- Trust in third party developed code
- Easy integration of third party developed code
- Maintain reusability of system core, avoid redundant code
- Avoid abdicating from strategic product management!

Summary

- Each application has its own set of unique challenges, but the *class* of applications may also have typical challenges and concerns
- These shape the solutions. Sometimes only a little, sometimes by dictating a certain architecture style.
- In this lecture a select few application classes have been introduced: Embedded, Mobile, Cloud, and Ecosystems.
- Embedded and Mobile application constraints are due to technical limitations.
- Cloud application constraints come from the users.
- Software Ecosystem constraints emerge from the developers.
- Next lecture: Game Architectures, as yet another example of a specific domain.
- Next lecture again: How do companies deal with all this?