

Software Architecture

Abridged presentation

An overview of what every software architect should know
[HTTP://www.shimon.us](http://www.shimon.us)

Shimon Rothschild
Shimon@Shimon.us
Tel: (1)310.651.8200

What is Software Architecture

This presentation highlights areas that a software architect need be proficient. Specific technology implementations depend on the organization and problem. The core knowledge outlined here is independent of any specific industry or technology.

Copyright ©2007 all rights reserved. This work may not be shown or distributed without written permission.

The Four Cornerstones

- Communication
- Foundation
- Technology
- Implementation

Whatever the industry or the methodology, these cornerstones exist.

Communication

- ◉ Understand the problem
- ◉ Talk to the development team
- ◉ Project progress and risk identification
- ◉ Support training and sales

Half of all software bugs are attributed to poor communication.

Foundation

- ◉ Process for delivery
- ◉ Validation and verification
- ◉ Design for flexibility
- ◉ Metrics

If the design can't be delivered or can't be tested
then how can it be useful?

Technology

- ◉ Data centric
- ◉ Object centric
- ◉ Trends
- ◉ Innovation

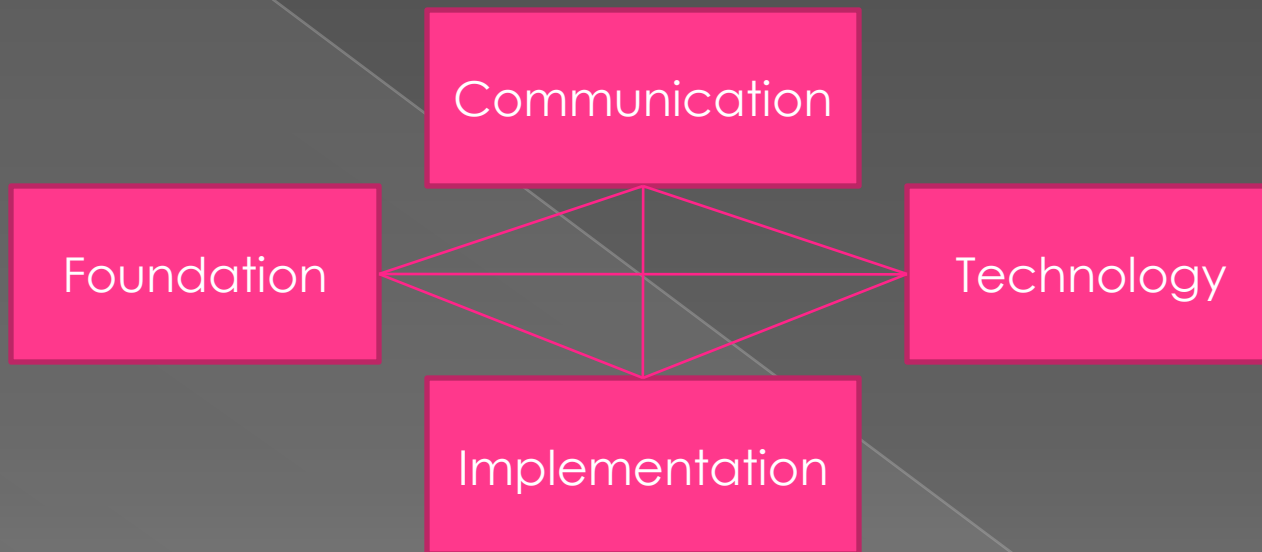
The programmers playground, but keep it in context.
It's one of four cornerstones.

Implementation

- ◉ Source for answers
- ◉ Lead by example
- ◉ Evaluate alternatives
- ◉ Cost / risk scenarios

Integrate implementation with software architecture because change is inevitable.

Each Supporting the Others



Details will change.
The rules will not.

Communication

- ◎ **Understand the problem**
- ◎ Talk to the development team
- ◎ Project progress and risk identification
- ◎ Support training and sales

Separate core functionality from current implementation.
Understand the business requirements and the impact of technology.

Get the Requirements

- ◉ Stakeholders
- ◉ Users
- ◉ Support staff

Stakeholders – business rules
User – current implementation
Support staff – current limitations

Communication

- Understand the problem
- **Talk to the development team**
- Project progress and risk identification
- Support training and sales

Each role in the team has a unique vocabulary.

Appropriate Terminology

- ◉ Business analysts
- ◉ Quality assurance
- ◉ Software developers

Each use a distinct and precise terminology to convey ideas and processes.

Communication

- Understand the problem
- Talk to the development team
- **Project progress and risk identification**
- Support training and sales

Accurate, balanced and timely notification increases accurate forecasting.

Timely Notification

- ◉ Project progress
- ◉ Potential problems and work around
- ◉ Discovered opportunities
- ◉ Cost / risk exposure

Set multiple level notifications to monitor progress and uncertainty.
Evaluate impact of opportunities that arise.

Communication

- Understand the problem
- Talk to the development team
- Project progress and risk identification
- **Support training and sales**

Design to support training and sales.

Early Exposure

- ◉ Material for training
- ◉ Tips and tricks for support
- ◉ Documentation for documentation

Share knowledge with those that will support the product.

Foundation

◎ **Process for delivery**

- ◎ Validation and verification
- ◎ Design for flexibility
- ◎ Metrics

Have a plan to deliver the software.

System for Software Delivery

- ◉ Plan of action with milestones
- ◉ Prioritization to assure an on-time deliverable
- ◉ Tools, training and materials
- ◉ System for secure and redundant code protection

“Things happen” so be prepared.

Foundation

- ◉ Process for delivery
- ◉ **Validation and verification**
- ◉ Design for flexibility
- ◉ Metrics

Validate it is what is expected and verify it works as expected.

Quality and Correctness

- ◉ Check against requirements
- ◉ Rectify inconsistent, conflicting and missing requirements
- ◉ Design to facilitate code testing
- ◉ Follow engineering principles

Validate: Requirements and usability
Verify: Code

Foundation

- Process for delivery
- Validation and verification
- **Design for flexibility**
- Metrics

Allow for change.

“Things Happen”

- Changing requirements
- Incorporate indecision
- Design for minor modification
- Anticipate future needs

When designing for change, the additional cost may be worth it.

Foundation

- Process for delivery
- Validation and verification
- Design for flexibility
- **Metrics**

Have metrics to identify potential risk and opportunity.

Track and Monitor Progress

- Use metrics to learn the true cost of process
- Metrics identify trends
- Metrics initiate alarm management

With metrics it becomes possible to identify problems early.

Technology

- ◎ **Data centric**
- ◎ Object centric
- ◎ Trends
- ◎ Innovation

Type of, source and use of data dictates the software architecture.

Data as King

- ◉ Data as a strategic asset
- ◉ Making data useful and not abused
- ◉ Capture the right data at the right time

Good data has value, bad data is worse than worthless.

Technology

- ◉ Data centric
- ◉ **Object centric**
- ◉ Trends
- ◉ Innovation

Objects are the common language between programmer, end user and stake holder.

Everything is an Object

- ◉ Business objects make sense
- ◉ Separate what (object) from how (rule)
- ◉ OOD is the standard

Objects are easy to understand and easy to maintain.

Technology

- ◉ Data centric
- ◉ Object centric
- ◉ **Trends**
- ◉ Innovation

This industry is full of trends and some are even useful.

What's Hot and What's Not

- ◉ Fad, fashion and style are not equal
- ◉ Marketing educates not dictates
- ◉ Know what the boss is reading
- ◉ Beware of quick fixes and magic potions

Staying current is essential, but discern hype from innovation.

Technology

- ◉ Data centric
- ◉ Object centric
- ◉ Trends
- ◉ **Innovation**

Innovation distinguishes one from the competition.

Intellectual Property

- Use proven processes in innovative ways
- Invent new ways
- Combine existing components into new patterns

Innovate or perish.

Implementation

- ◎ **Source for answers**
- ◎ Lead by example
- ◎ Evaluate alternatives
- ◎ Cost / risk scenarios

How long will it take to find the answer?

The Encyclopedia

- ◉ Know how to find any answer
- ◉ Know the limit of certainty
- ◉ Recognize fact, opinion and assumption

Be a resource that can be easily approached for help.

Implementation

- ◉ Source for answers
- ◉ **Lead by example**
- ◉ Evaluate alternatives
- ◉ Cost / risk scenarios

Be a source of inspiration.

Charge!

- ◉ Deliver more than is asked
- ◉ Be prepared to fill in or assist
- ◉ First in / last out
- ◉ Be positive, positive and positive

The mood and tempo start here.

Implementation

- ◉ Source for answers
- ◉ Lead by example
- ◉ **Evaluate alternatives**
- ◉ Cost / risk scenarios

Change is inevitable and does not have to be bad.

Weigh the Alternatives

- Know how to probe for weakness
- Set metrics to accurately measure alternatives
- Deferring selection late increases probability of selecting the best alternative

First choice is not always the best choice.

Implementation

- ◉ Source for answers
- ◉ Lead by example
- ◉ Evaluate alternatives
- ◉ **Cost / risk scenarios**

Identify and manage uncertainty.

Risk Can Kill

- Generate alternatives to high risk activities
- Identify potential cost exposure to activities of high risk
- Is activity failure an option?

Murphy's law: Disaster will strike at some point, so plan for it

Full Presentation is Available

Order the book:
Purely Practical Software Architecture
Shimon@Shimon.us

[HTTP://www.shimon.us](http://www.shimon.us)

Contact:
Shimon Rothschild
(1) 310.651.8200