Meta-ArchitecturE Document

ARCHITECTURE IN RESPONSE TO:

Team-Chat Feature Request

a submission by:

Chris Cargile

GAYATHRI SARATHI

NICKLAUS RHODES

DELIVERED:

February 12, 2014

**Table of Contents**

[How To Use This document Template 3](#_Toc379555488)

[Architectural Vision 3](#_Toc379555489)

[Principles 4](#_Toc379555490)

[Styles 5](#_Toc379555491)

[Patterns & interconnection Mechanisms 5](#_Toc379555492)

[Pattern [NAME] 5](#_Toc379555493)

[Interconnection Mechanism [name] 6](#_Toc379555494)

[Philosophies & Preferences 6](#_Toc379555495)

[Guidelines & policies 6](#_Toc379555496)

[Additional Information 6](#_Toc379555497)

Meta-ArchitecturE document

team tom-chat

# How To Use This document Template

Meta-Architecture as an activity is gathering ideas from other architectures and past experience so that you don’t repeat mistakes and waste time reinventing solutions.

Team Tom-chat has discovered that given the status of the art and science surrounding software engineering, certain choices can regularly lead to more highly-valued outcomes in a project. To ensure the team-focused chat feature the team will produce is as elegant, effective, efficient, dependable, evolvable and scalable as possible, Tom-chat will seek to apply these decisions, in practice.

As the style of the Tom-chat chat functionality is intended to embody stylistic elements representative of a highly-distributed architecture, it will do well to uphold principles which enforce such decisions. The ideas in their broadest sense which Tom-chat will ascribe to for these purposes include: physically separating software components of the requester from the provider; making service providers unaware of requester identities; insulating requesters from each other, to allow dynamic associations of requesters to providers for the chat service; and allowing multiple chat server modules to come into play to achieve a more balanced load across the provider instances.

These embodiment of these decisions may be expressed in the form of:

* A RESTful chat program implementation: the program design will be representative of principles associated with the client/server model; it will rely on the HTTP GET/POST mechanisms for transport of chat messages
* A multi-tiered design implementation: physical separation of entities is achieved via a web based container serving client requests and passing the data along with transport to the database connector at the point a request is registered with the container. The server component in the container will forward data in the requests asynchronously, to be written to the storage location designated by the database server-side storage engine, via a database connector. However, data committed to long-term storage will not be synchronously forwarded to clients, which are the observers, in this system. Rather, the database values will be recorded as a log for chat sessions.
* Requesters will be insulated from one another via threading within the same server (.servlet/.jsp) that has a single server module (worker). Multiple server modules can be assigned to handle REQUESTs to the web-server/container, in a more robust implementation of the functionality, but we suffice that only a single server module will be needed, given a team size of less than 5 members.
* Alternative approaches to the integrated webserver/container environment that accepts and responds to requests using a STATELESS-, HTTP/1.1, request/response-, client/server implementation include: using a design in which state is maintained using URL rewriting or storing, retrieving, and polling the datastore for updates, to then forward those to clients; using alternative protocols or component configurations that allow for responding to chat-based messages via adding on as a new component of the receiving web-server’s native extension API or using different, more sophisticated messaging architectures to achieve a similar effect, such as JMS employing one of various JMS architectural models, including publish-subscribe, for example.

The level of formalism should suit the project it describes. Making these ideas explicit, rather than leaving them implicit, helps ensure that a team shares a unified vision and goal.

# Architectural Vision

Describe your vision for the architecture here. What is the architecture going to do for your team, or your business?.This should be drawn from your vision for the project.

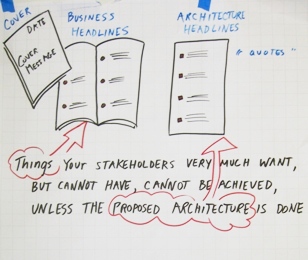


Figure 1. Cover story vision for the project

Figure 1. is used by permission from Bredemeyer Consulting.

# Principles

By “principle” we mean “a behavior or practice that leads to good things”. Example: “Conceptual Integrity can be achieved by uniformly applying a limited number of design forms”.

Guidelines on from for architectural principles:

* Each principle clearly states a fundamental belief of the organization.
* Each principle should be stated in a way that you will know if the architecture has the characteristics expressed by the principle
* Each principle should have a counterargument; that is, they should not be platitudes or general features that are desirable regardless of the system.
* Principles should be simply stated and understandable
* Principles need to be rationalized, stating why the principle is preferred; draw on business-related factors where possible
* The implications of adopting the principle should be identified if possible

You should include one table for each principle. To include new principles, cut and paste copies of the template below.

|  |  |
| --- | --- |
| Principle Name |  |
| Description |  |
| Rationale/Benefits |  |
| Implications |  |
| Counterargument |  |

# Styles

Here is definition of style that helps distinguish it from Patterns, although at this level there is not much distinction. Style is determined by

* A set of component types that perform some function at runtime
* A topological layout of these components indicating their runtime relationships
* A set of connectors that mediate communication, coordination, or cooperation among components (function call, RPC, data stream, socket, etc.)
* A distinctive name that describes the collection.

We can identify styles by answering the following questions, from Shaw and Garlan:

* What is the design vocabulary—the types of components and connectors?
* What are the allowable structural patterns?
* What is the underlying computational model?
* What are the essential invariants of the style?
* What are common examples of its use?
* What are the advantages and disadvantages of using the style?
* What are some common specializations of the style?

# Patterns & interconnection Mechanisms

Three types of patterns are of interest:

* Structural Patterns
* Interconnection Patterns
* Decoupling Patterns

These follow directly from describing architecture in terms of a set of components and their relationships.

Patterns are described in terms of a Pattern Form (see below), and one or more CRC “cards” (see below), which describe each of the major pieces of the pattern. Cut and paste copies of the Pattern Form and CRC table (as needed), for each Pattern or Interconnect Mechanism described. You may not think of every pattern the first tie through. You may struggle with rationale. That’s OK. Plan time in your project to revisit this document.

If you are thinking of following a reference architecture, this is the place to describe and consider it.

## Pattern [NAME]

### Summary Description –

### Context of USE (INTENT)

### Problem Statement

### Solution Description

### Variants and Related Patterns

### Known Uses -

### Consequences

Rationale:

|  |  |
| --- | --- |
| Class: | Collaborator: |
| Responsibility: |  |

## Interconnection Mechanism [name]

This is the place to consider standard interconnection technology and solutions that you may use, such as COM, CORBA, JavaBeans, Mule Enterprise Service Bus, MS Kinect.

# Philosophies & Preferences

Often there are ideas, philosophies or cultures that affect teams and projects. If it doesn’t fit anywhere else, put it here.

# Guidelines & policies

* Here you want to put general guidelines and policies that:
* Might affect the architecture & design of the system
* Don’t fit other places in this document

# Additional Information

Include comments or ideas that you want here, but that don’t fit anywhere else. Keep this section to a minimum, because you want a document people can use easily, not a tome of esoteric lore.