Seminar Design Patterns und Anti-Patterns

Architecture Anti-Patterns

Arkadiusz Gabryś
Matrikel-Nr. 21895081
Department of Compuer Science

July 20, 2015

Abstract — Content of this document describes three common Architectural Anti-Patterns. Sources and causes, symptoms and consequences, examples and solutions will be discussed.

Contents

1 Introduction

1.1 Software Architecture

Good architecture is a critical factor in process of transforming high level models into actual implementation [3]. Unfortunately the engineering discipline of software architecture is relatively immature and there is even no official definition [4]. SEI provides three types of software architecture definitions [5]:

- Modern definition – The set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both.
- Classic definitions – An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization.
- Bibliographic definitions – Software architecture is the study of the large-scale structure and performance of software systems. Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information.

The common aspect of those definitions is the placement of software architecture as a layer between high level abstract model and specific implementation.

1.2 Architecture Anti-Patterns

Anti-Pattern is a commonly used solution that results in bad consequences It describes the practice itself as well as solution [6]. Architecture Anti-Patterns are those patterns that occur at architecture level, so called system-level or enterprise-level [1].

In this document, while describing anti-pattern, firstly the general form of a bad solution will be presented both with it causes. Next the main symptoms and consequences and after that some real life example will be presented to describe example solution. In case of anti-patterns there occur some exceptions - situations when an anti-pattern can be accepted - and that will be the last chapter for each one of them.

This whole project (excluding graphics) was created using LaTeX technology and its sources are available at https://github.com/gabr/dpap_aap

2 Stovepipe Enterprise

2.1 Form & Causes

Stovepipe Enterprise, also know as Island of Automation [8] is a situation when multiple system within enterprise are designed independently at every level [9]. Those systems has potential to share data, functionality or whole subsystems but they don't.

Stovepipe metaphor isn't accidental. Stovepipe is the pipe which conducts smoke from a coal or wood-burning stove to it's chimney. There are two problems with such pipes and two similar problems will occur in case of stovepipe enterprise anti-pattern. The first problem is that burning wood produces corrosive substances that erode metal, so pipe must be constantly maintained and repaired in order to avoid leakage. Second problem is fact that stovepipes are never connected with each other to create one system. If one would have a two coals he would have two separate stovepipes to maintain.

This analogy perfectly fits into stovepipe enterprise anti-pattern where we have separated systems which has to be maintained separately because of they layer separation. This and other problems will be discussed in the next chapter.

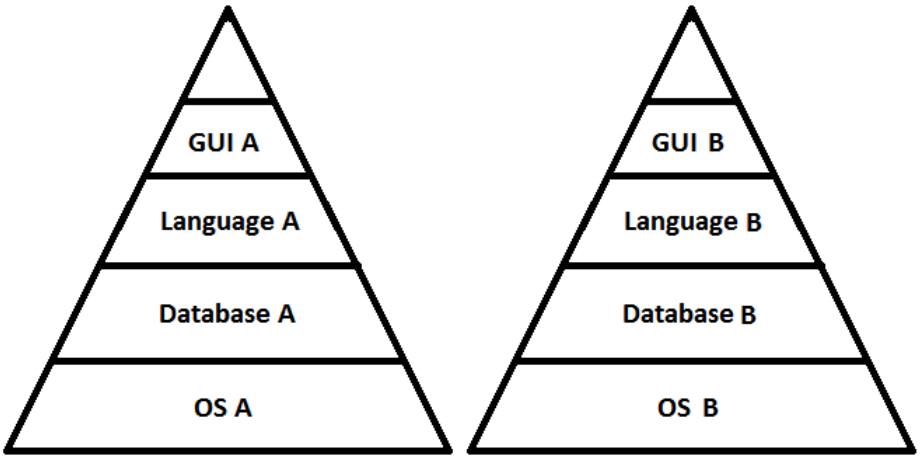


Figure 1: Stovepipe Enterprise Systems - example visualization

Causes of such situations are in general lac of standards, lac of communication and laziness. When a company have global system standards, technology strategy and system profiles then even without teems and enterprise institutions communication such anti-pattern could be avoided. Lack of communication, lack of knowledge about technology standards being used in company and absence of horizontal interfaces in system integration solutions are the main causes of the stovepipe enterprise anti-pattern [1]. It can also came from rapid expansion of the company and systems - from few small applications to big enterprise systems [2].

2.2 Symptoms & Consequences

The main symptoms and consequences of the stovepipe enterprise anti-pattern are [1]:

- Lack of software reuse between enterprise systems
- Lack of interoperability between enterprise systems
- Brittle systems
- Monolithic systems

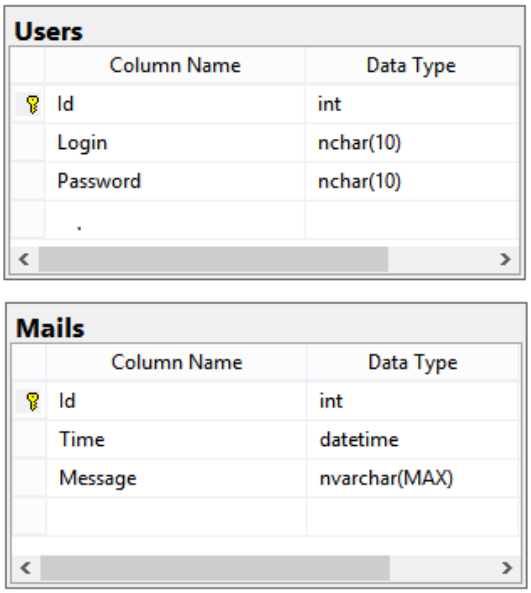
- Undocumented architectures
- Inability to extend systems
- Excessive maintenance costs
- Employee turnover may causes project discontinuity and maintenance problems

Many of those consequences can be notices at the early stages of system creation. If some subsystem can be maintained only by one employ this is probably separate "island" of the system. If system architecture is known only to system team and there is no enterprise standard then undocumented architecture problem will occur. Also no one else outside the team would be able to quickly join to the team or connect two different systems.

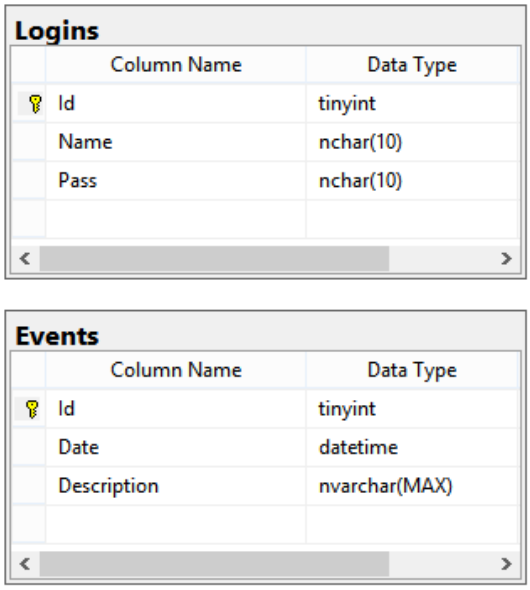
2.3 Example - authorization system

As an example of the stovepipe enterprise anti-pattern will be considered a e-mail system and calendar system within one company.

Both systems besides they unique data had to store information about they users and passwords. In enterprise where are no standards on database layer defined we can expect that those systems can use different technologies, naming conventions and will probably not share the information. Figure 2 shows tables for those two systems. In case where those systems uses different databases technologies the gap between database layer is even higher.



(a) E-mail system - example tables in MySQL database



(b) Calendar system - example tables in MSSSL database

Figure 2: Stovepipe Enterprise Systems database example - different tables in different databases technologies

Not only the the separation of tables is the symptom of the stovepipe enterprise anti-pattern but also the different naming conventions and *Id* field data type. Such systems require double work in case of changing any user data in any of those two systems. So it is inconvenient both in development and in administration.

2.4 Solution*

[1] Coordination of technologies at each level is necessary to both avoid and solve stovepipe enterprise problem. Defining standards at each level is the first step and it should be done in order from the lowest level (high-level architecture) to the highest (API specifications, extensions etc).

Topic is new but the problem is old, so many large enterprises developed conventions for the definitions of object-oriented architectures that can be applied to many organizations. The key is to define detailed interoperability [2] conventions across systems of large-scale architectures. And at the same time address technology strategy and requirements. Experience has shown that four requirements models and four specification models has to be defined in order to properly scope interoperability between layer.

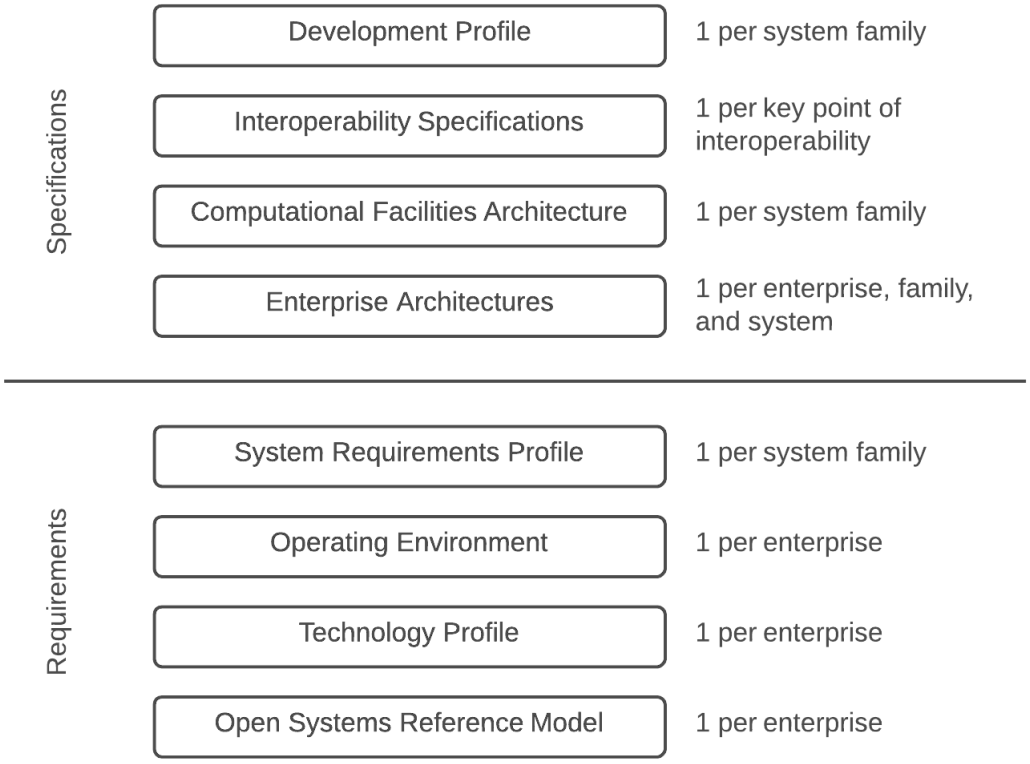


Figure 3: Scoping of interoperability [1]

Detailed description of each layer will not be discussed.

[1] Due to lack of other source the whole chapter is based on [1]

[2] Property of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, without any restricted access or implementation. [7]

Based on that, the solution for the example database layer show on figure 2 can be provided. The first step will be to define one database technology, then the naming conventions and types standards. After that one table for all user for both systems can be defined as it is shown on figure 4.

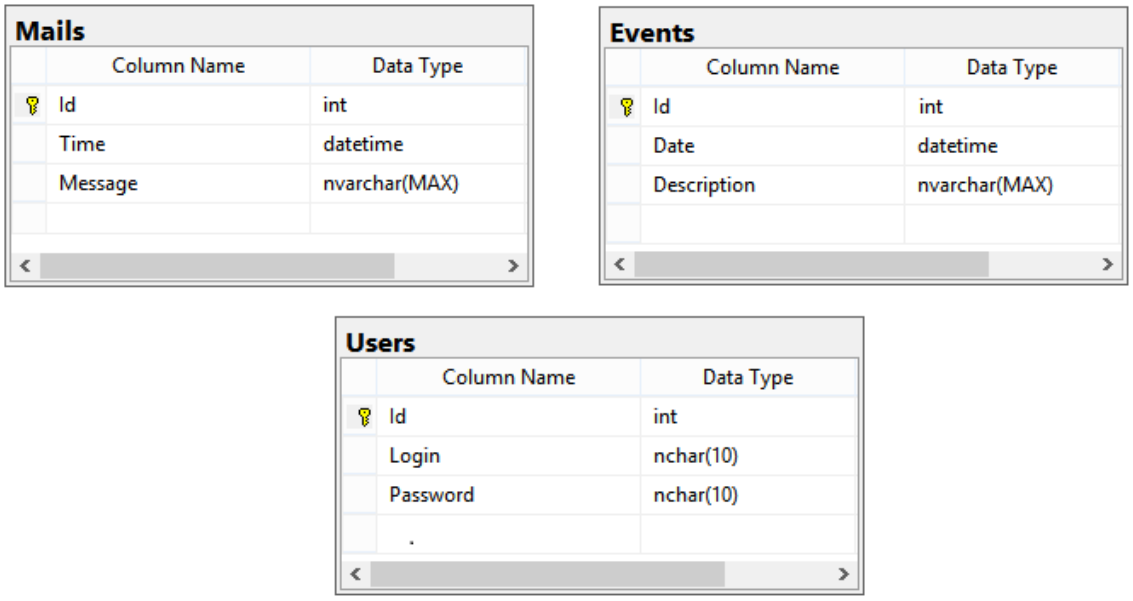


Figure 4: Example database layer - solution

2.5 Exceptions

3 Stovepipe System

3.1 Form & Causes

3.2 Distinction

3.3 Symptoms & Consequences

3.4 Example

3.5 Solution

3.6 Exceptions

4 Vendor Lock-In

4.1 Form & Causes

4.2 Symptoms & Consequences

4.3 Example

4.4 Solution

4.5 Exceptions

5 Summary

List of Figures

References

[1] Alexander Shvets *AntiPatterns The Survival Guide*, published by https://sourcemaking.com

[2] W.H. Immon *Slaying the stovepipe dragon* 24 April 2003 p. 2

[3] *Nordic Workshop on Model Driven Engineering* ISBN: 978-91-7295-985-9, 27-29 August 2007 Ronneby, Sweden p. 5

[4] *Architektura oprogramowania*, web: https://pl.wikipedia.org/wiki/Architektura_oprogramowania 10 July 2015

[5] Software Engineering Institute, Carnegie Mellon University *What is your definition of software architecture?*, web: http://www.sei.cmu.edu/architecture/start/glossary/definition-form.cfm 10 July 2015

[6] Brad Appleton *Patterns and Software: Essential Concepts and Terminology* 19 September 2008 12:06 AM p. 7

[7] AFUL dedicated website for Definition of Interoperability http://interoperability-definition.info/ 20 July 2015

[8] *Stovepipe Anti Pattern* web: http://c2.com/cgi/wiki?StovepipeAntiPattern 11 July 2015

[9] Brown, W., Malveau, R., McCormick *AntiPatterns Refactoring Software, Architectures, and Projects in Crisis* III, H., Mowbray, T., John Wiley & Sons, Inc., 1998