Design Patterns in N-tier Architecture

Viachaslau Kachurka, Brest State Technical University

Abstract

The efficiency of solution of practical computational problems depends heavily on system architecture design. In case of distributed and modulated system with the connections between elements lying over bad and unreliable network hybrid design patterns should be considered for overhead reducing. In this paper such pattern called MVA-F is presented. It is based on the standard patterns of MVC family in combination with a “Façade” pattern. Its main advantage is basic-level stability due to one point of entrance to model layer for different parts of upper layers. The comparison with standard patterns made on synthetic tests proves that this approach can be used in the systems which are used in unstable environment.

1. Introduction

In software engineering an important role is played by architecture choice and when the final decision is made about the design in base of newly programmed software.

For a standalone application, the architecture choice can affect (except the engineering difficulty) only the computational performance of application at the terminal computer [1].

In case of usual simple web-application, when the design should conform the basic model “client-server” [2], a price of architecture errors increases – the application uses the resources not only of the terminal client, but also of the server(s) and of the network infrastructure.

And when application works not only in simple “client-server” model, the problem of dispatching the elements in highly dissociated and distributed system occurs [6].

In general, software architect has to make a decision between some widely used and explored [3-5] practices; but in presence of additional conditions the choice becomes unobvious.

Consider a problem $P$ of software design, which has two key conditions:

1. A computational system should be distributed over network;
2. Network is, with high probability, either unreliable or even unusable.

The problem can be stated as – which design pattern should we follow to decrease engineering difficulty and minimize time of request dispatching?

Considering the second condition, the main goal of solving problem $P$ is to find the most harmless environment for user which will work with most efficiency in a network with a high package loss rate.

2. Existing design patterns

2.1 N-tier Architecture

N-tier application architecture provides a model for developers to create a flexible and reusable application. In most cases a 3-tier architecture is implied, - presentation tier, logic tier and data tier, - where the elements are divided and distributed physically.

But in general a design which uses N-tier architecture can consist of many dedicated tiers, each distributed over network, as shown on Figure 1.

![Fig. 1. Example of the N-tier architecture usage](image)

The most valuable positive side of such architecture is the flexibility in scaling, programming the independent modules etc., while it also creates a lot of pitfalls [6].

The usage of this architecture is mostly either formal, when infrastructure already provides most of the tiers (Client’s browser + Web-server + DBMS), or dictated by outer reasons, like security reasons, problem specific conditions, etc. [6]

N-tier architecture satisfies the first condition of problem $P$, leaving the second condition unmatched.
2.2 MVC Family


The main idea of these patterns is to divide every element of the application (in case of N-tier, every tier can be MVC/MVP) into exact parts with strictly dedicated roles. In case of MVC it is:

- Model – manipulates data and provides business-logic of element [7];
- View – uses data from model to create a representation, shown to end-user;
- Controller – dispatches the connection between Model and View [7].

MVP pattern uses another division, into 3 main (Model, View, Presenter) and 3 middle (Selections, Commands, Interactors) parts (as shown on Figure 2), where the middle parts are used only for linking the main parts, and, not like in MVC, the most business-logic is done in Presenter, which also dispatches the whole process [4].

![Fig. 2. MVC, MVP, MVA patterns](image)

Also there is a not well known pattern MVA [5], Model-View-Adapter, where model is data-only abstraction, view is the end-user representation of data, and between them there is a variable number of Adapters, every one of which changes the data. Adapters provide business logic.

This pattern is better suited for “classic”, non-Web applications, where user interacts only with View.

Also, due to linear direction of dataflow, this design can be easily physically divided, so in the “naïve” approach this design is most suitable for N-tier architectures by default.

2.3 Synthetic N-tier tests

To choose a pattern to use with N-tier architecture to solve the problem \( P \), a set of synthetic tests \( T \) has been created. Each test \( T(PAT, PLn, C, N, K\%) \) measures time, which a data manipulation operation (selection, in conditions of different connectivity) over network takes, where \( PAT \) is tested pattern, \( K\% \) is package loss rate, \( PLn \)-number of testing platform, \( C \) – used compiler and \( N \) is number of operations, done in a row.

Actually we can measure how much time will pass between creating a request by user till having the results acquired and shown properly. In tests \( T \) such time is measured as time between first request sending and the \( N \)-th successful data presentation for user, where \( N \) should be large enough to decrease the “infrastructure noise” influence.

To decrease influence of “virtual machines”, interpreters or some other middleware, we chose a compilable language (C++), which can be distributed over different platforms without changes (Qt framework) [9].

Compilers – MS Visual CPP Compiler, with default options (only OS Windows), GCC with default options (optimization O0 – without optimization), GCC with optimization level O2.

Platforms – 5 different by performance computers with multiple OS.

Simulated packet losses – not higher than 5%, as shown by T. Bonald [8].

Examples of test results in tables 1, 2:

<table>
<thead>
<tr>
<th>Tab.1. Example of test results for MVC without package loss (K = 0%) in milliseconds per request processing, whereas in tests N = 100, 1000, 5000, 10000 (median)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
</tr>
<tr>
<td>#1</td>
</tr>
<tr>
<td>#2</td>
</tr>
<tr>
<td>#3</td>
</tr>
<tr>
<td>#4</td>
</tr>
<tr>
<td>#5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tab.2. Example of test results for MVP on different packet losses rates in milliseconds per operation (median)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
</tr>
<tr>
<td>#1</td>
</tr>
<tr>
<td>#2</td>
</tr>
<tr>
<td>#3</td>
</tr>
<tr>
<td>#4</td>
</tr>
<tr>
<td>#5</td>
</tr>
</tbody>
</table>

Due to uncomparability of different platforms in our case, we can use only relative, not absolute
values – in how many times time per operation has increased due to package loss:

\[
R(PAT, PL, K%) = \frac{\text{Median}(T(PAT, PL, C.N, H\%))}{\text{Median}(T(PAT, PL, C.N, H\%))}
\]

where \( T(0\%) \) is time per operation without package losses, ms; \( T(K\%) \) - time per operation in conditions with K% package losses, ms.

![Fig. 3. Rate of time consumption with K% packet loss of MVC, MVP, MVA patterns](image)

The value which we can actually compare (Figure 3) is a unified mean rate \( R' \):

\[
R'(PAT, K\%) = \frac{\sum_{i=1}^{N} R(PAT, PLi, K\%)}{\text{Median}(PLi)}
\]

In this case we can use the mean due to already normalized (mediated) values.

MVC shows the best performance, MVP shows strange pike at 2.5% while still having the same efficiency at high package loss rate, while MVA wastes much more time.

### 3. Hybrid pattern MVA-F

To add some efficiency to MVA pattern we can add a new layer which will dispatch requests and filter the “dead ones” efficiently, to reduce time consumption.

This new layer can be called “Façade” due to its comparability with existing pattern of the point of one entrance (Figure 4).

It dispatches all the requests from views/adapters to models and can efficiently delete “dead requests” or restart them so that adapters (and views, and user) do not know about network latency etc.

On the other hand this adds more difficulty into pattern, which needs more time to implement.

Synthetic tests of MVA-F were done in the same conditions as other patterns (Figure 5).

The time consumption for MVA-F tests shows on higher package loss rates the time efficiency of its usage compared to standard patterns.

![Fig. 4. MVA-F concept](image)

### Tab.3.

<table>
<thead>
<tr>
<th>Platform</th>
<th>0%</th>
<th>0.1%</th>
<th>1%</th>
<th>2.5%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>7.83</td>
<td>7.84</td>
<td>7.95</td>
<td>8.02</td>
<td>8.26</td>
</tr>
<tr>
<td>#2</td>
<td>16.78</td>
<td>16.8</td>
<td>17.04</td>
<td>17.23</td>
<td>17.64</td>
</tr>
<tr>
<td>#3</td>
<td>11.21</td>
<td>11.22</td>
<td>11.35</td>
<td>11.52</td>
<td>11.81</td>
</tr>
<tr>
<td>#4</td>
<td>12.88</td>
<td>12.89</td>
<td>13.05</td>
<td>13.23</td>
<td>13.55</td>
</tr>
<tr>
<td>#5</td>
<td>10.51</td>
<td>10.52</td>
<td>10.67</td>
<td>10.77</td>
<td>11.08</td>
</tr>
</tbody>
</table>

![Fig. 5. Rate of time consumption with K% of packet loss of MVA-F compared to other patterns](image)
Conclusions

As it is shown in p.3, the first of widely known and used practices – MVC – shows good efficiency compared to its “heirs” in conditions of problem P. But the hybrid pattern MVA-F, proposed here, has better time efficiency (reduces inefficiency of its predecessor), while being a more sophisticated to create but more easy to be divided into tiers, if needed.

Usage of hybrid pattern in N-tier software over unreliable network is already tested in real problem solutions, where it has shown its efficiency. Its usage is highly recommended when following conditions are met: N-tier architecture, unreliable network, and possibility to divide the every tier into multiple when needed.

Bibliography


Author:

Viachaslau Kachurka
Brest State Technical University
Moskovskaya str., 267
224017 Brest
tel. +375 29 2045651
e-mail: vakochurko@bstu.by