A Systematic Method for Identification of Anti-patterns in Service Oriented System Development

Mohammad Ali Torkamani*, Hamid Bagheri**
* R&D Department, Iranian Telecommunication Manufacturing Company, Shiraz, Iran
** Information Technology, Kurdistan University, Sanandaj, Iran

Article Info

Article history:

Received Aug 7, 2013
Revised Dec 19, 2013
Accepted Jan 5, 2013

Keyword:

Anti-patterns
Check list
Repository
Service-Oriented Architecture

ABSTRACT

Service-Oriented Architecture is one of the popular software architecture's patterns used for developing lots of modern systems. However, it has been involved in many failures. Anti-patterns are solutions which have good view, but in fact they are wrong solutions that cause failure of systems. There are a lot of anti-patterns for SOA and new anti-patterns are revealed every day. Anti-patterns have their own reasons for being formed and also they are appeared in special area of the problem. As human's mind is restricted and it can process a limited number of states (piece of information) therefore identification of anti-patterns will be difficult for architects. In this paper, we propose a systematic method based on repository of anti-patterns along with a check list to identify anti-patterns of SOA. This method will assist architects to easily detect and avoid anti-patterns in development process and so escape from risks which related to anti-patterns. Furthermore, in this paper, we present a repository of forty five general anti-patterns in SOA. Reviewing these anti-patterns will help developers to work with clear understanding of patterns in phases of software development and so avoid from many potential problems. Also, our method is evaluated in action.

Corresponding Author:

Mohammad Ali Torkamani
R&D Department
Iranian Telecommunication Manufacturing Company, Shiraz, Iran
Email: torkamani_ali@yahoo.com

1. INTRODUCTION

There are attractive and good view solutions in the processing of service-oriented development which in fact are wrong methods and so result to failure of projects. These solutions are called anti-patterns. Architects should have considerable information about all of anti-patterns and continuously update their knowledge through study new resources about anti-patterns in order to avoid from wrong solution (anti-patterns) and finish their projects successfully. An anti-pattern can cause to increase in cost and result in failure of project in the worst case. As numbers of anti-patterns are so much and everyday new anti-patterns are revealing thus discovery of anti-patterns will be so difficult for architects. These anti-patters hardly menace systems based on service-oriented architectures.

Unfortunately, many developers do not pay attention to these anti-patterns and always go through their knowledge and experiences in their past projects and also available tools. On the other hands, managers as stakeholders of projects are people who do not have technical expertise in developing service-oriented systems. Therefore, there is a need for a systematic method for identification of anti-patterns. In this paper, a systematic method for identification of anti-patterns in service-oriented development is presented. Our method use check lists and repository of anti-patterns for discovery of anti-patterns.

The rest of this paper is structured as follows. Section 2 will be discussing related concepts and repository of service-oriented architecture’s anti-patterns will be presented in Section 3. Moreover, we describe some noticeable anti-patterns in this section which result is other anti-patterns. The systematic method for identification of service-oriented architecture's anti-patterns will be explained in section 4. In Section 5, we will have a case study which shows how our method works and finally we will have conclusion.

2. ANTI-PATTERNS IN SERVICE-ORIENTED ARCHITECTURE

An anti-pattern is attractive and known solution for solving a problem which is not practical and usable. In other words, anti-patterns are solutions which result in failure instead of successful [1]. There is a language or format for expressing anti-patterns which includes following items [2, 3]:

- Name: a name for introduction of the anti-pattern
- Bad solution: something that happens or unsuitable solution which is related to the anti-pattern.
- Symptoms: guide of sign of problem
- Consequence: result of applying the anti-pattern
- Root cause: description of how the pattern is used wrongly and cause to failure.
- Proposed solution: repaired solution which solve the problem and warranty usability.

Anti-patterns are categorized to three groups named software development, software architecture, and project manager anti-patterns [4]. Some anti-patterns in object-orientation are available in service-oriented architecture. However, considering this point that relation between service-oriented systems and business is stronger than that in object-orientation, leads to anti-patters which are specific to SOA [5]. A categorization which is completely in association with SOA is defined by IBM as following:

- SOA Adoption anti-patterns: These anti-patterns are caused to delay in adaption or inconsistency consumer and business with SOA. “So, What is New?” and “Big Bang” anti-patterns are examples for this category.
- Service identification & design anti-patterns: These anti-patterns are shown when developers perform design and discovery of services as a part of SOA's project. “Web services as SOA” is a sample from this category.
- Service realization anti-patterns: These anti-patterns present the worst experience for realization of services. Many of these anti-patterns are concentrated on web services which are typical method for realization of SOA. For example, “Conversation Services” and “Point to Point services” anti-patterns are instances of this category.

According to mentioned classification, it is clear that anti-patterns can be seen in the whole of software development phases. Anti-patterns usually cause to costly and complicated architecture which its consequence is costly and difficult implementation. Even, using and applying anti-patterns in a system can break a project. Therefore, identification and management of anti-patterns has high priority in the process of development. Moreover, this approach is one of substantial methods which architects can be sure that their architecture is valid and appropriate. On the other hand, if architects use a private or customized solution, it has much probability to become a wrong solution and finally become an anti-pattern. Furthermore, if architects use a new technology or approach and have less experience with service-orientation, it has again high probability that they are faced with defeat.

2.1. Measuring Risk of Anti-Patterns

As anti-patterns are caused to increasing in cost of project and even failure, measuring risk of anti-patterns and using risk management techniques have substantial role in projects which are based on SOA. SOA's anti-patterns have various happening probabilities and can consequence to variable deficiencies. Risk of several anti-patterns calculated based on these two factors in [5]. For each anti-pattern, these two factors may have less or much value which allocated based on experiment of a developer. By considering these two factors, we will have three categories of anti-patterns which have less, much, and very much risk probabilities respectively. Risk management of anti-patterns is done in this way that list of anti-patterns are provided and then risk of them are calculated. They are sorted by their value of risk and called based on their risk values. Anti-patterns with very much risk value are called critical anti-patterns which risk management must be done for them. It is often necessary that risk management is also performed for anti-patterns which have much risk value.

3. REPOSITORY OF SOA ANTI-PATTERNS

By considering the mentioned items in Section 2, all developers need a comprehensive repository or reference of anti-patterns. In order to satisfy this need, a number of resources reviewed and 45 anti-patterns of SOA, which are typical and important, extracted from [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. These anti-patterns listed in Table 1. The considerable point about this list is that number of anti-patterns is more and they are increasing continuously but indicated anti-patterns in Table 1 just covers the most popular and important anti-patterns of SOA. Architects can complete the repository by studying and investigating new resources and best practice of other developers.

Table 1. Repository of SOA Anti-patterns

row	Anti-pattern
1	Point to Point Web Services [1]
2	Too many Cooks in the SOA [9]
3	CRUD (Create, Read, Update, Delete) Interface [9]
4	Loosey Goosey [9]
5	Chatty Interfaces [1]
	Dialogue Services[3]
6	Web Services Equal SO [2]
	Service Proliferation Syndrome
7	Component-less Services [9]
8	Interface Bloat [9]
9	SOA Over standardized or Standardization Paralysis [6]
10	Vendor-Lock-In anti-pattern [5]
	Everything Must Be New
	No Legacy
11	Nothing New [6]
	So, What's New?
12	Fine-Grained Interfaces, Fine-Grained Services [6]
13	Sand Pile [5]
14	Atomic Services [5]
15	On-Line Only [5]
16	From Application Silos to SOA Silos [10]
17	Build It and They Will Come [10]
18	Over-Engineer Reference Architectures [10]
19	Undefined Baseline for Business ROI [10,13]
	Thermal Runaway
20	Web Service Sprawl [10]
21	Armchair Architecture from the Ivory Tower [10,13]
22	Scattered SOA Strategy Entangled with IT Strategy [10]
23	Expecting a Free Ride on the SOA Train [11]
24	Where’s the Money? [10]
	EAI 2.0 SOA Equals
25	The Shiny Nickel [8]
	Magpie [8]
26	The Technology Altar [8]
27	Percolating Process [8]
28	Splitting Hairs [8]
29	IT2B [8]
	The people's republic of IT
30	DIY Transport [8]
31	Nobody Home [8]
32	UBER service [8]
33	A Million Services all in a row [8]
34	Architectural Stovepipe [8]
35	Defensive SOA [8]
	Optimistic SOA
36	Integrating Distributed and Heterogeneous Data Sources in SOAs [11]
37	Data Replication [12]
38	Technology bandwidth [2]
39	Misbehaving Registries [2]
40	Business Process for Ever [5]
41	Cyberspace [5]
42	Grey Services [5]
43	Big Bang [9]
	Bite more than you can chew
44	The Butterfly Effect [13]
45	Catch-22 [13]

4. IDENTIFICATION OF ANTI-PATTERNS IN THE PROCESS OF SOA

Our proposed method is based on check lists and repository of anti-patterns. Table 2 presents proposed check list for anti-patterns. This check list is result of analysis anti-patterns indicated in Table1. Indeed, the chick list is composed of several questions which using of them to identifying anti-patterns is useful. These questions should be answered with “Yes”, “No”. If response of the questions becomes positive so it is hopeful that the selected approach is far from any anti-patterns. However, if one of responses becomes negative so we can certainly consider that we will encounter anti-patterns. This method will be useful for people who don't have any experience with anti-patterns or don't have technical view. Indeed, we will have a high level abstract of anti-patterns. Column 3 of Table 2 presents possible anti-patterns related to check lists.

This column shows that if the answers of these questions are "No" which anti-patterns will happen. Anti-patterns which are introduced in this column are just samples. Architect can complete the table based on his or her experience and knowledge and studying unsuccessful experience of other organizations as well. This table is well suited for team working, thus developers through brain storming anti-pattern sessions can exchange ideas and discuss these issues. This process enhances the creativity and communication and leads to collective mind of coordinators. In proposed model, which is shown in Figure 1, assumes that organization has anti-pattern repository. When an anti-pattern is discovered or architect find out new anti-pattern, the repository should be updated and new anti-pattern should be added. An experienced architect attempts to update anti-pattern repository in order to use it efficiently by studying anti-patterns in other organizations or unsuccessful experience of other developers and studying recent published resources about anti-patterns as well. Even the Chief Architect may employ someone as an "anti-pattern specialist" especially in big projects. The responsibilities of this person are: research, collect, design, detect, evaluate, risk management for anti-patterns, present Anti Pattern Risk Document and update anti-pattern repository. Another responsibility of anti-pattern specialist is produce required checklist, so we propose a new architect product which means "Anti Pattern Risk Analysis Document". Now, the question is if an organization has an updated anti-pattern repository and has enough experience in Service Oriented Architecture, chick list is needed? The answer is yes, because human's mind is restricted and it can process a limited amount of information. On the other hand, the number of anti-pattern in Service oriented Architecture are extremely high and according to [2], the much SOA is big, the more anti pattern will be discovered. Check list categorize anti-patterns and make anti-pattern finding in repository easy. Every question in checklist is related to some anti-patterns so checklist plays an important role in layering and stereo typing. Suppose you are an architect and you have 300 anti-pattern descriptions in the repository. Comparing and evaluating of 300 anti-patterns with architecture are not straightforward.

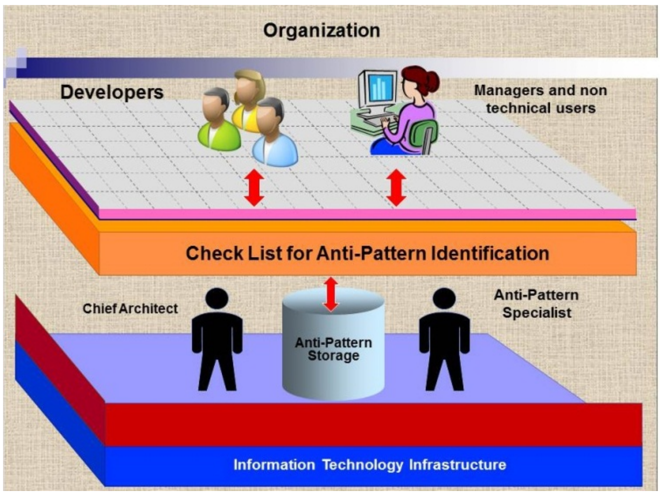
Organization
Developers
Managers and non technical users
Check List for Anti-Pattern Identification
Chief Architect
Anti-Pattern Storage
Anti-Pattern Specialist
Information Technology Infrastructure

Figure 1. Check List place for Anti-Pattern identification

Checklist by asking some questions makes the problem domain small and you are not forced to consider the whole domain. Every question in checklist is related to some anti-patterns, so instead of considering 300 anti-patterns small numbers of anti-patterns, which are in the same group, should be take into account. It is completely clear that considering 4 anti-patterns are more straightforward than 300 anti-patterns. Some anti-patterns may belong to different anti-patterns for instance, "Fine Grained Interface" and "Fine Grained Service" anti-pattern are related to questions number 1 and 9 in table 2. According to different researches, for example [6], the mentioned anti patterns, which are the most important anti-patterns in SOA, can cause so many anti-patterns to happen. Checklist makes the process of finding prominent and important anti-pattern easier; it means anti-patterns which relate to more than 2 questions in table 2 are more significant.

5. PROPOSED MODEL IN ACTION

Iranian Telecommunication Manufacturing Company (ITMC) is a company operating in Electrical engineering and ICT areas. ITMC besides some products in electrical and communication area produces so many web based software, for instance Finance and Human Resources, for itself and other central telecommunications. Right now personnel in Training and Research unit in cooperation with MIS unit are developing website for R&D based on SQL Server and Visual C#. The website has main sections; Research and Training. In training section, users can view training courses and seminars. Besides that users can register in new courses and apply for vacancies. Also applicants, who want to work as an internship in ITMC, should register all processes through website without attending. Also people who take courses in ITMC should receive their certificates by websites. Managers should be able to connect to website and get different reports by local network. Shareholders who are resident in different cities should take benefits of facilities produced by website.

In research section, comprehensive information should be mentioned about works done in ITMC for example papers, research plans, projects, seminars and presentations. Besides that, website must have a section for registering new ideas and research plans. If someone has an idea and wants to be supported by ITMC should register his or her idea.

Proposed model is applied in this project and results are shown in table 3. In column two in table 3, question number related to table 2 is written. Answer of the question is shown in column three in this table and anti-pattern is recognized in column four. In column five actions to encounter related anti-pattern is shown and finally the results are revealed in column six.

As it is clearly seen, four anti-patterns are recognized in this project. It is obvious if these anti-patterns don’t detect and manage; they will increase the cost of organization and will cause undesirable side effects.

Results of applying proposed model in ITMC project are: reduce costs, enhance system efficiency, change developer’s view, detect organization’s software assets and reuse them in future projects. As it is clearly seen that proposed model can detect prevalent anti-patterns easily. Architects can study more anti-patterns and complete table 2 so that they can discover more anti-patterns. According to current case study, applying proposed approach in all organization in different level of SOA maturity and different developing experiences is highly beneficial.

Table 2. Checklist for anti-pattern identification in SOA

Row	Question	Possible anti-pattern	Actions should be done
1	Is organization ready to accept SOA architecture? Are number of services and fine-grained proper?	According to Gartner [7], one of the reasons of failure in SOA is lack of concern in service number and not preparing for SOA. This is due to the large numbers of services and fine grained (“Fine Grained Interface” and “Fine Grained Service” anti-pattern). According to [6], (“Fine Grained Interface” and “Fine Grained Service” anti-pattern can cause so many other anti-patterns.	– Raising awareness about fine grained services – Not using inappropriate standard
2	Has organization any experience in SOA?	“So, what’s new?” anti-pattern can appear due to lack of dominance of developers about SOA and derivation of Object Oriented [6].	– Emphasize on how SOA is different from earlier solutions. – The facilities provided by an ESB, Which is an essential part of SOA, such as Transport Services, Mediation Services, and Event Services, are examples of new capabilities made available by SOA. – Provide successful examples that will demonstrate the success and feasibility of implementing a SOA solution
3	Are selected architecture and technology at the same direction with business goal?	“The Shiny Nickel” anti-pattern shows that there isn’t pay attention to where the organization is going and huge investment is made in technology. Technology is not at the same direction with organization goals.	To tackle this problem, we have to know where we want to go. It means where your organization wants to go in future? Also deep understanding of phases, Standards and technologies are needed to reach your destination.

Row	Question	Possible anti-pattern	Actions should be done
4	Does selected approach satisfy SOA principles?	"Chatty Interfaces" anti-pattern Violate Loose coupling, Statelessness and Abstraction principles[9]	The most effective approach is to redesign along with combination of closely related information
	Do the Services have appropriate performance? Do large amount of information exchange in some services?	"Interface Bloat" anti-pattern breaks the loose couple and combinability principles. Low performance and exchange overload information are an indication that this anti-pattern is beingapplied.	– Paying attention to SOA principles and avoidance of solution which violate SOA principles – Divide particular service into some several services.
5	Is Proposed approach financially affordable?	Some anti-patterns lead to complicated and costly architecture?	technical and financial analysis of solutions
6	Is proposed approach support predecessor technology?	"The Shiny Nickel" anti-pattern puts aside previous technologies [8].	We should take advantage of organization assets in previous projects and technologies should be applied which support the current organization technology
		"SOA Over standardized (Standardization Paralysis)" and "Fine Grained Interface" and "Fine Grained Service" anti-pattern are another samples relate to this question.	– Paying attention in selecting new standards – Avoidance of selecting complicated and costly standards
7	Does anti-patterns storage exist in your organization?	All possible anti-patterns in SOA	
8	Does anti-pattern storage update regularly? Do developers study latest resources about anti-patterns?	All anti-patterns that do not exist in anti-pattern storage.	Create an anti-pattern storage and try to keep it update
9	Are services fine grained appropriate?	"Fine Grained Interface" and "Fine Grained Service" anti-patterns	Review service fine grained
10	Does investment focus on SOA architecture and technology or on the business?	The "Technology Alter" anti-pattern focus is on the SOA and technology instead of business. Indeed business goal demote to second rank. In "Technology Bandwidth", architect is seen from technology point of view rather than business. One consequence of this anti-pattern is increasing in IT cost and there is no payback [2].	– Paying attention to business goal and aligning technology with business goal. – Information technology should be at the service of business.
11	Does your organization depend on particular technology? Has organization bias in particular technology?	"Big Bang" anti-pattern	CIO should avoid dependence on particular technology. Technology selection should be based on organization requirement and technology benefits.

Table 3. Results of case study

Row	Question	Answer	Anti-pattern	Actions	Results
1	2	No, after presenting 2 seminars about SOA principles for ITMC personnel, they said there is nothing new and we have done it before!	So, what's new	Presenting more contents about SOA for developers Avoidance of Object Oriented solutions	The observance of SOA principles in project and avoidance of several anti-patterns.
2	4	"Service combinability" is not considered	Interface Bloat	Some services exchange large amount of data so that they divided to several services.	Efficiency of system was increased and resulted in customer satisfaction
3	6	No, some applications in ITMC were developed by Delphi 6 but they are not used.	Shiny Nickel	Reusability of older codes	100 million Rials cost reduced.
4	11	Developers emphasize on C# much more than needed	Big Bang	Presentation about advantages of java programming and reusability of older codes.	Group formed with 3 members to take advantages of older codes Developers became interested in Java

6. CONCLUSION

There are attractive and good view solutions in the processing of service-oriented development which in fact are wrong methods and thus result to failure of projects. These solutions are called anti-patterns. As there are large numbers of anti-patterns and day by days we face new anti-patterns, thus discovery of anti-patterns will be so difficult for architects. These anti-patters hardly menace systems based on service-oriented architectures. , managers as stakeholders of projects are people who do not have technical expertise in developing service-oriented systems. Therefore, there is a need for a systematic method for identification of anti-patterns. In our paper we introduced new method based on check lists and repository of anti-patterns for discovery of anti-patterns, so that developers can discover anti-patterns in service oriented architecture and do risk management. Check list is used as guidance for anti-pattern storage and is highly beneficial for organizations in order to detect anti-patterns.

Even an organization has an updated anti-pattern repository and has enough experience in Service Oriented Architecture, chick list is needed because human's mind is restricted and it can process a limited amount of information. On the other hand the number of anti-pattern in Service oriented Architecture are extremely high and according to [2], the much SOA is big, the more anti pattern will be discovered. Check list categorize anti-patterns and make anti-pattern finding in repository easy. Every question in checklist is related to some anti-patterns so checklist plays an important role in layering and stereo typing. Suppose you are an architect and you have 300 anti-pattern descriptions in the repository. Comparing and evaluating of 300 anti-patterns with architecture are not straightforward. Some anti-patterns may belong to different anti-patterns for instance, “Fine Grained Interface” and “Fine Grained Service” anti-pattern are related to questions number 1 and 9 in table 2. Checklist makes the process of finding prominent and important anti-pattern easier; it means anti-patterns which relate to more than 2 questions in table 2 are more significant. Architect can complete the suggested table in this paper based on his or her experience and knowledge and studying unsuccessful experience of other organizations as well. This table is well suited for team working, thus developers through brain storming ant-pattern sessions can exchange ideas and discuss these issues.

Another suggestion which introduced in our paper is that Chief Architect may employ someone as an “anti-pattern specialist” especially in big projects. The responsibilities of this person are: research, collect, design, detect, evaluate, risk management for anti-patterns, present Anti Pattern Risk Document and update anti-pattern repository. Another responsibility of anti-pattern specialist is produce required checklist, so we propose a new architect product which means “Anti Pattern Risk Analysis Document”.

Proposed model is applied in Iranian Telecommunication Manufacturing Company (ITMC) project and results are shown in table 3. Results of applying proposed model in ITMC project are reduce costs, enhance system efficiency, change developer’s view, detect organization’s software assets and reuse them in future projects.

REFERENCES

[1] Noori M and Riazi M. “Anti-Pattern in Service oriented Architecture”. First Information Technology Conference, Iran, 2010, http://www.ibm.com/developerworks/webservices/library/ws-antipatterns/, 2010-11-14

[2] Carrato T and Srinivasan H. “Learning from Mistakes: A guide to SOA anti-patterns - how to benefit from known unworkable solutions”. 2007,http://soa.sys-con.com/node/318437, 2010-11-14.

[3] Brown WJ, Malveau RC, McCormick HW and Mowbray TJ. “AntiPatterns - Refactoring Software, Architectures and Projects in Crisis”. John Wiley & Sons, 1998.

[4] Kr´al J and Zemliˇcka M. “The Most Important Service-Oriented Antipatterns”. *IEEE*. 2007.

[5] Kr´al J and Zemliˇcka M. “Enterprise Architecture Antipatt”. *IEEE*. 2009.

[6] Kavis M. “Top 10 reasons why people are making SOA fail”. July 2008. http://www.cio.com/article/438413, 2010-11-14.

[7] Jones S. “SOA anti-patterns”. http://www.infoq.com/articles/SOA-anti-patterns, Jun 19, 2006.

[8] Biswas P. “Principles of Service Orientation: What is SOA as illustrated by what it is not”. 2010, www.percepsys.com/blog/WhatIsNotSOA.pdf

[9] Oracle. “Oracle White Paper in Enterprise Architecture—SOA Anti-Patterns: How Not to Do Service-Oriented Architecture”. 2010.

[10] Hacigumus H. “Anti-Patterns: Integrating Distributed and Heterogeneous Data Sources in SOAs”. *IEEE*.2008.

[11] Microsoft MSDM, Data Replication as an Enterprise SOA Antipattern, 2013, http://msdn.microsoft.com/en-us/library/bb245678.aspx

[12] Chatterjee S. “Enterprise Architecture Antipatterns: The reason behind so many dots”. Wednesday, June 04, 2008.

[13] http://www.cioindex.com/nm/templates/itstrategy.aspx?articleid=70280

[14] Torkamani MA, Abdelzad V, Bashavard S. “Anti-pattern repository in service oriented architecture”. Second International Conference in Computer (CICIS’11), Zanjan, 2011

BIOGRAPHIES OF AUTHORS

Mohammad Ali Torkamani was born in Iran, Shiraz City, in 1975. He received the M.S. degree in software engineering from the Shahid Beheshti University, in 2011. He is the author of 15 books (in Persian), more than 35 articles. His research interests include software architecture, Ultra Large Scale systems, cryptography and Network security and holds one patent.
He is working in R&D Department of Iranian Telecommunication Manufacturing Company now. Also, he is currently teaching at the University of Applied Science and Technology in Shiraz.

Hamid Bagheri was born in sanandaj, Iran, in 1979. He received the B.S. and M.S. degrees in Software engineering from the Shahid Beheshti University, Tehran, in 2011. Since 2009, he has been working in Information Technology in Kurdistan University. His research interests include Service Oriented Arhcitecture and Ultra large Scale Systems.