ISSN 1884-0760

GRACE TECHNICAL REPORTS

Proceedings of the 1st Asian Conference on Pattern Languages of Programs (AsianPLoP 2010)

Volume I

Hironori WASHIZAKI Nobukazu YOSHIOKA
(editors)

GRACE-TR-2010-01 March 16, 2010

CENTER FOR GLOBAL RESEARCH IN
ADVANCED SOFTWARE SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF INFORMATICS
2-1-2 HITOTSUBASHI, CHIYODA-KU, TOKYO, JAPAN

WWW page: http://grace-center.jp/

Proceedings

AsianPLoP 2010

1st Asian Conference on Pattern Languages of Programs

Volume I

Tokyo, Japan, March 16-17, 2010, Co-located with

The GRACE International Symposium on Advanced Software Engineering

Edited by

Hironori Washizaki and Nobukazu Yoshioka

Sponsored by

GRACE Center of the National Institute of Informatics (NII)

IPSJ/SIGSE Patterns Working Group

ACM Japan Chapter

Supported by

eXtreme Programming Japan Users Group (XPJUG)

Conference Committee

Conference Chair

Eiichi Hanyuda, Mamezou, Co. Ltd.

Program Co-Chairs

Hironori Washizaki, Waseda University / National Institute of Informatics GRACE Center

Nobukazu Yoshioka, National Institute of Informatics

Program Committee

Eduardo B. Fernandez, Florida Atlantic University
Joseph W. Yoder, The Refactory Inc and Joe Yoder Enterprises
Norihiro Yoshida, Osaka University
Masao Tomono, KameNet Inc.
Koido Ryo, eXtreme Programming Japan User's Groups
Terunobu 'Terry' Fujino, InArcadia, Ltd.
Kiminobu Kodama, Information System Research Institute
Kenji Hiranabe, Eiwa System Management, Inc.
Masaru Amano, Eiwa System Management, Inc.
Takeshi Kakeda, Eiwa System Management, Inc.
Kenichiroh Ohta, IBM Japan
Yuriko Sawatani, IBM Japan
Akira Sakakibara, IBM Japan
Takao Okubo, Fujitsu Limited, Japan
Okita Naoyuki, Yokogawa Electric Corporation
Akio Kawai, Object Design Laboratory, Inc.
Yann-Gaël Guéhéneuc, Canada Research Chair on Software Patterns and Patterns of Software, École Polytechnique de Montréal
Yuji Yamano, OGIS International, Inc.
Yoichi Hasegawa, Technoport
Takashi Iba, Keio University
Takashi Kobayashi, Nagoya University
Dinesha K V, IIIT Bangalore
Raj Datta, MindTree Ltd.
Eric Platon, Cirius Technologies
Foutse Khomh, DIRO, Universite de Montreal, QC
Tsukasa Takemura, NSD CO., LTD.

Message from PC Co-Chairs

Welcome to the 1st Asian Conference on Pattern Languages of Programs, AsianPLoP 2010. AsianPLoP takes place at the first time, as a premier event for pattern authors and users to gather, discuss and learn more about patterns and software development in the Asia region as well as other regions.

The purpose of AsianPLoP is to promote development of patterns, pattern languages, technologies and experiences of patterns primarily about software; however, these for domains outside software are also welcome.

In AsianPLoP 2010, various patterns, pattern languages and related techniques will be discussed. Topics include software design, services, security, interaction, pedagogy and organizational change. Most of papers will be workshopped in the traditional PLoP Writer's Workshop format. We received 16 paper submissions. After the rigorous shepherding processes, 13 papers have been accepted for Writer's Workshops and 3 papers for Writing Groups. Moreover the 1st program incorporates one invited talk and one tutorial.

We thank program committee members. They reviewed and conducted shepherding processes for papers carefully and fairly. Moreover we thank our sponsors, supporters and the Hillside Group for their kind supports. We hope that the 1st conference of AsianPLoP is successful, and will contribute the development of this filed.

Hironori Washizaki and Nobukazu Yoshioka

Program Co-Chairs

Invited Talk

Title: A Timeless Way Of Communicating

Presenter: Joshua Kerievsky (Industrial Logic, Inc.)

Abstract

If you pick up the masterpiece, "A Pattern Language", by Christopher Alexander et. al, you will discover a book filled with engaging photographs, hand-drawn sketches, big bold, hard-to-miss text, memorable stories and scholarly notes for the academically minded. One can quickly "surf" this book by focusing only on pattern titles, images and headlines or one can dive deep into the book by reading the detailed text of each pattern. In short, A Pattern Language uses a timeless way of communicating, a form that engages people and provides numerous pathways for accessing the knowledge. As authors of software-related pattern languages, we must understand what it takes to make our own works endure. In this talk, we will analyze the form and content of real-world software patterns/pattern languages, looking for what makes them succeed or fail at engaging the reader and providing knowledge pathways. If you are interested in crafting great pattern languages, this talk will help you discover some essential ingredients.

Biography

Joshua Kerievsky is founder of Industrial Logic, Inc., an early pioneer and expert in Extreme Programming (XP), author of the best-selling, Jolt Cola Award-winning book Refactoring to Patterns , thought leader behind Industrial XP, a state-of-the-art synthesis of XP and Agile Project Management and an innovator of Agile eLearning, which helps organizations “Scale Agility Faster.” Joshua has over 20 years of experience in software development and loves coaching agile project communities, helping executives understand and manage technical debt, leading excellent workshops, and building software products (because it enables him to “walk the agile talk” as an entrepreneur, manager, customer and programmer).

Tutorial

Title: Pattern Writing: The Straight Scoop

Presenter: Joseph W. Yoder (The Refactory, Inc.)

Abstract

Writing Patterns can be a difficult task and getting started sometimes is the most difficult step. Pattern ideas start to emerge from experience practitioners but if you don't have the experience of writing patterns, it can be daunting on how to capture these experiences and start outlining your patterns. This tutorial will discuss ideas on How to Write Patterns. We will discuss Patterns for Writing Patterns and outline some different processes that beginning pattern writers can use to start the process of capturing their patterns. We will also examine some different pattern forms and workshop on some pattern writing.

Contents: Volume I

Contents: Volume II

A pattern for the WS-Trust standard for web services

Ola Ajaj and Eduardo B. Fernandez
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431-0991 USA
oajaj@fau.edu, ed@cse.fau.edu

Abstract: Web services intend to provide an application integration technology that can be successfully used over the Internet in a secure, interoperable and trusted manner. One of the main functionalities of web services is providing secure messaging, where the web services exchange security credentials (either directly or indirectly). However, each party needs to determine if they can trust the asserted credentials of the other party. Moreover, the dynamic interaction between the web services requires specifying trust relationships in an explicit way for all parties. Without a clear definition of how web services could manage secure communications and establish trust relationships with other partners, malicious web services could use their business interactions to perform illegal actions. The WS-Trust standard defines how to establish trust between interacting parties; we present here a pattern for this standard. WS-Trust defines a security token service and a trust engine which are used by web services to authenticate other web services. Using the functions defined in WS-Trust, applications can engage in secure communication after establishing trust.

1. Introduction

Without a clear definition of how web services can manage secure communications and establish trust relationships with other partners, it would be hard to perform any kind of interaction. WS-Trust is a standard to support the establishment of trust relationships.

Using web services requires that we exchange credentials to define the rights of each participant. This exchange is based on trust and builds further trust. Trust is based on security and other policies to enable requesting and obtaining credentials within different trust domains. Both parties need to determine if they can "trust" the asserted credentials of the other party. The goal of the WS-Trust standard is to enable applications to construct trusted message exchanges. This trust is realized through the exchange and brokering of security tokens [oas09].

The motivation toward WS-Trust is supported by the fact that there are different formats for security tokens (e.g. X.509 certificates, Kerberos tickets, SAML assertions, XACML policies, etc.), and it's unlikely to expect that an endpoint will understand each of these options. Additionally, there is no guarantee that there will be an intersection between the sets of supported security token formats of different actors who are willing to exchange messages using the WS-Security standard [Mad03].

Web services standards are rather complex and verbose and it is not easy for designers and users to understand their key points. By expressing web services security mechanisms and standards as patterns, we can verify if an existing product implementing a given security mechanism supports some specific standard [Fer06]. Inversely, a product vendor can use the standards to guide the development of the product. By expressing standards as patterns, we can compare them and understand them better. For example, we can discover overlapping and inconsistent aspects between them. We have produced patterns to describe SAML, XACML, WS-Policy, WS-Security, XML Encryption, XML Digital Signature, and others. A standard defines a generic architecture and this is a basic feature of any pattern; it can then be confirmed as a best practice by looking at products that implement the standard (and implicitly the pattern).

Section 2 shows a pattern that describes this standard. Section 3 ends the paper with some conclusions.

2. A Pattern for WS-Trust

Intent

WS-Trust defines a security token service and a trust engine which are used by web services to authenticate other web services. Using the functions defined in WS-Trust, applications can engage in secure communication after establishing trust.

Example

The *Ajiad* travel agency offers its travel services through several different business portals to provide travel tickets, hotel and car rental services to its customers. *Ajiad* needs to establish trust relationships with its partners through these portals.

The *Ajiad* supports different business relationships and needs to be able to determine which travel services to invoke for which customer. Without a well-defined structure, *Ajiad* will not be able to know if a partner is trusted or not, or to automate the trust relationships quickly and securely with its partners, which may lead to losing a valuable business goal of offering integrated travel services as a part of the customer's portal environment.

Context

Distributed applications need to establish secure and trusted relationships between them to perform some work in a web-service environment which may be unreliable and/or insecure (e.g. the Internet). The concept of "Trusting A" mainly means "considering true the assertions made by A", which does not necessarily correspond to the intuitive idea of trust in its colloquial use.

WS-Security begins with the assumption that, if one of the parties uses a particular type of security token within the WS-Security header, then the other party will be able to interpret and process this token. A fundamental issue that WS-Security did not address is how two entities (a SOAP client and SOAP Service) can agree on the nature and characteristics of the security tokens that are the fundamentals of WS-Security.

Problem

Establishing security relationships is fundamental for the interoperation of distributed systems. Without applying relevant trust relationships expressed in the same way between the involved parties, web services have no means to assure security and interoperability in their integration. How can we define a way for the parties to trust each other's security credentials?

The possible solution is constrained by the following forces:

- ***Knowledge:*** In human relationships, we are concerned with first knowing a person before we trust her. That attitude applies also to web services. We need to have a structure that encapsulates some knowledge about the unit we intend to trust.

- ***Policy consideration*:** The web service policy contains all the required assertions and conditions that should be met to use that web service. The trust structure should consider this policy for verification purposes.

- ***Confidentiality and Integrity*:** Policies may include sensitive information. Malicious consumers may acquire sensitive information, fingerprint the service and infer service vulnerabilities. This implies that the policy itself should be protected.

- ***Message integrity***: The data to be transferred between the partners through messages may be private data that need to be protected. Attackers may try to modify or replace these messages.

- ***Time Validity***: For protection purposes, any interactions or means of communications (including the trust relationships) between the web services should have a time limit, that determines for how long the trust relationship is valid.

Solution

We define explicitly an artifact (security token) that implies trust. This artifact implies what kinds of assertions are required to make trustworthy interactions between the involved web services.

We should verify the claims and information sent by the requester in order to obtain the required security token that becomes a proof enough to establish a trust relationship with its target partners.

Structure

Figure 1 describes the structure of this pattern. **Claim** is a statement made about the attributes of a client, service or other resource (e.g. name, identity, key, group, privilege, capability, etc.). Claims are assertions, for example: "I am Joman", "I am an authenticated user and I am authorized to print in printer P". Claims are used to validate the requests made by a sender and need to be verified.

A **Security Token** is a collection of claims. It is possible to add signatures to tokens. **Security Token** also is a generalization of two types: **Signed Security Token** that is cryptographically endorsed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket) and **Proof-of-Possession (PoP)**

Token that contains a secret *data* parameter that can be used to prove authorized use of an associated security token and provides the function of adding digital signature. Usually, the proof-of-possession information is encrypted with a key known only to the recipient of the PoP token.

The **Security Token Service (STS)** is a web service that issues security tokens. It makes decisions based on evidence that it trusts. The **STS** is responsible for generating security tokens and, providing challenges for the requester to ensure message freshness (the message has not been replayed and is currently valid), verification of authorized use of a security token, and finally establishing, extending and removing trust in a domain of services. The **STS** is the heart of WS-Trust and forms the basis of trust brokering. The main output of the **STS** is a trust relationship between the requester and the receiver expressed as a security token. It represents the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes in a secure, reliable and time-relevant manner.

Each **STS** has a **Trust Engine** that evaluates the security-related aspects of a message using security mechanisms and includes policies to verify the requester's assertions. The **Trust Engine** is responsible for verifying security tokens and verifying claims against policies. A **Policy** is a collection of policy assertions that have their own name, references, and ID. Policies form the basic conditions to establish a trust relationship. Verifying the requester's claims against policy assertions generates an approval to use the target service. A policy may reference another policy (ies), in order to check the tokens sent by the requester or verified by the receiver.

Dynamics

We describe the dynamic aspects of the WS-Trust using sequence diagrams for the use cases "*create security token*" and "*access a resource using a token*".

Create a security token (Figure 2):

Summary: STS creates a security token using the claims provided by the requester.
Actors: A Requester
Precondition: The STS has the required policy to verify the requester claims and the requester provides parameters in form of *claims* and *RequestType* signed by a *signature*.
Description:

a. The requester requests a security token by sending the required *claims* and *RequestType* signed by a *Signature* to the STS. The signature verifies that the request is legitimate.
b. The STS contacts the Trust Engine to check the requester's claims.
c. The Trust Engine contacts the web service's policy to verify the claims including attributes and security token issuers of the requester.
d. Once approved, the STS creates a security token containing the requested claims.
e. The STS sends back its *SecurityTokenResponse* with a security token issued for the requester.

Postcondition: The requester has a security token that can be used to access resources in a trusted unit.

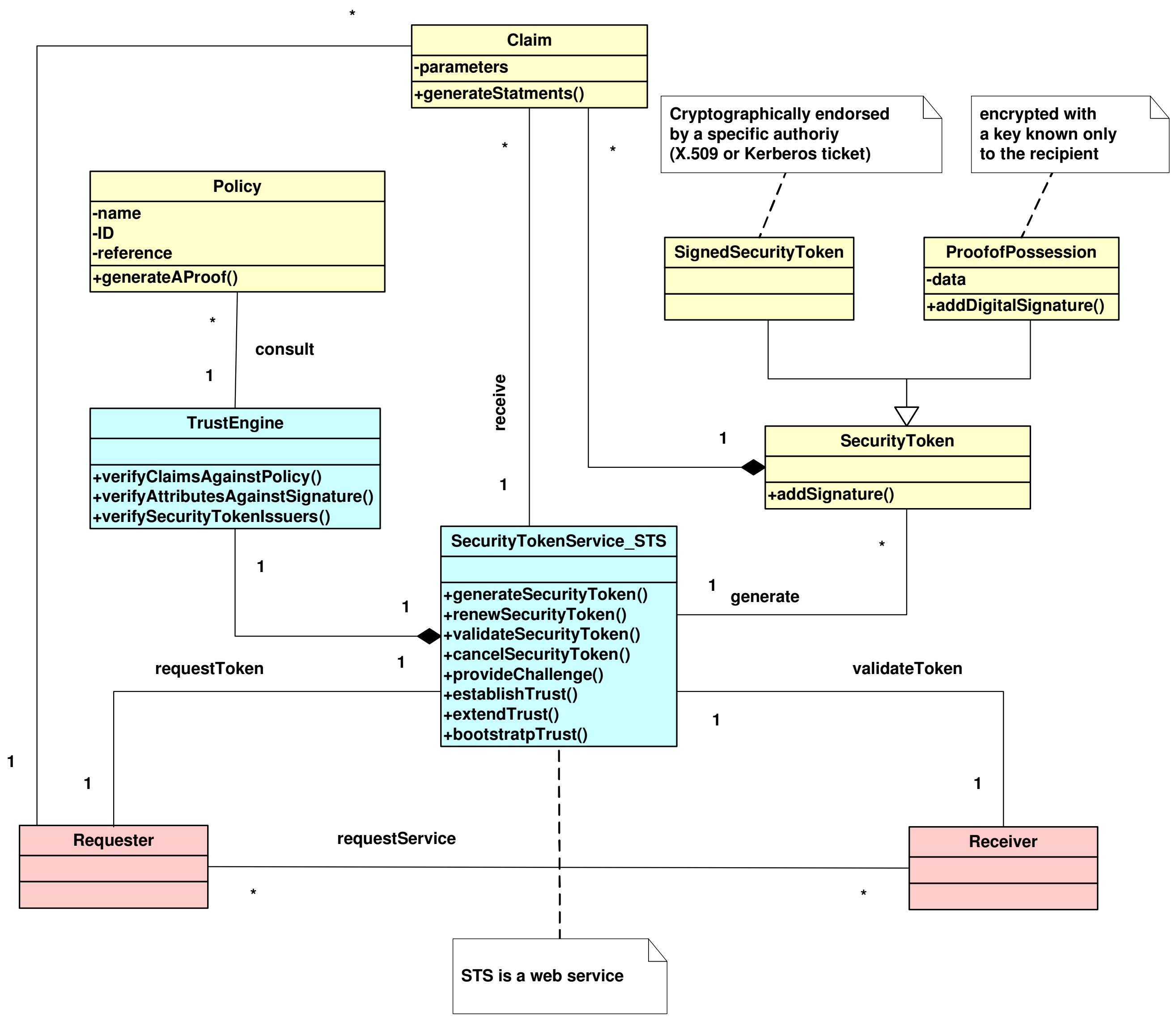


Figure 1: Class Diagram for the WS-Trust Pattern

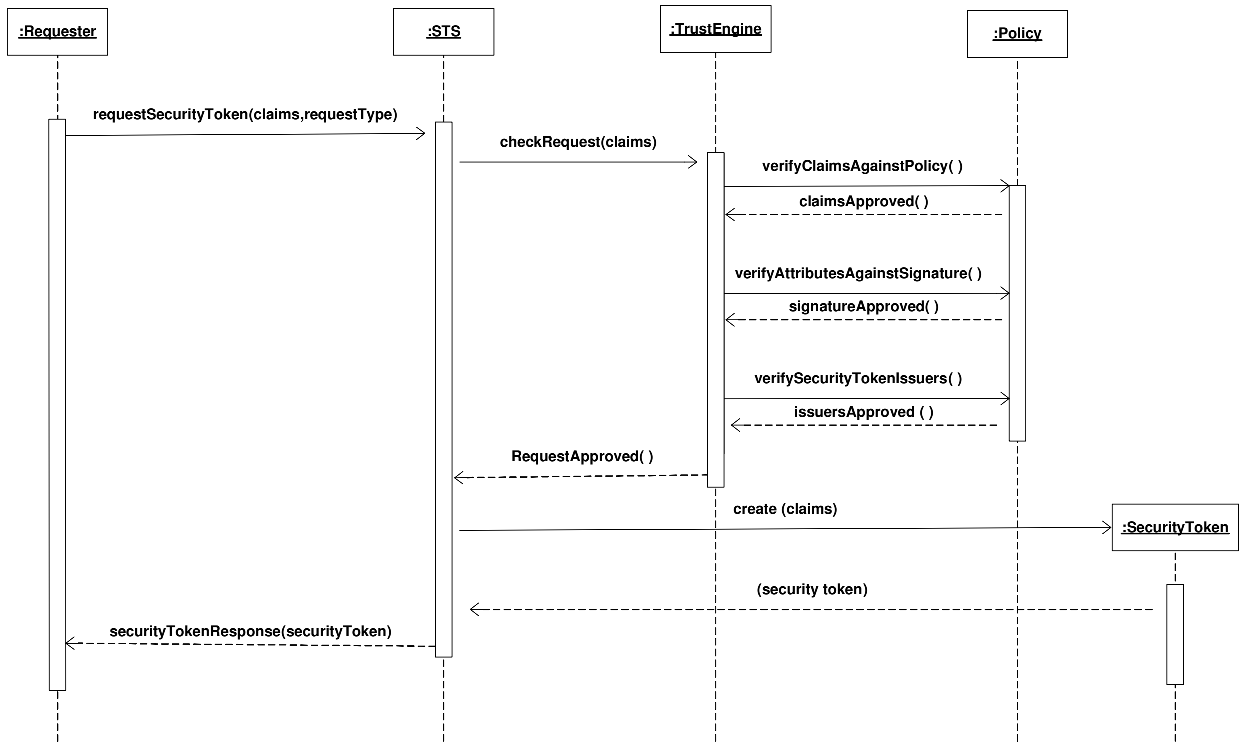


Figure 2: Sequence Diagram creating a security token

Access a resource using a token (Figure 3):

Summary: A STS allows the use of resources by establishing trust by verifying *proofOfClaims* sent by the requester.
Actors: A Requester
Precondition: The Trust Engine has the required policy to verify the requester' security token.
Description:

a. The requester asks for a service access by providing the required security token.
b. The receiver sends the security token to the STS for verification.
c. The STS use its Trust engine to verify the security token claims.
d. Once approved, the STS notifies the receiver that the security token is valid and verified.
e. The receiver gives the requester a token that implies the right to use the service.

Postcondition: The requester has a security token that can be used to access services in a Receiver web service.

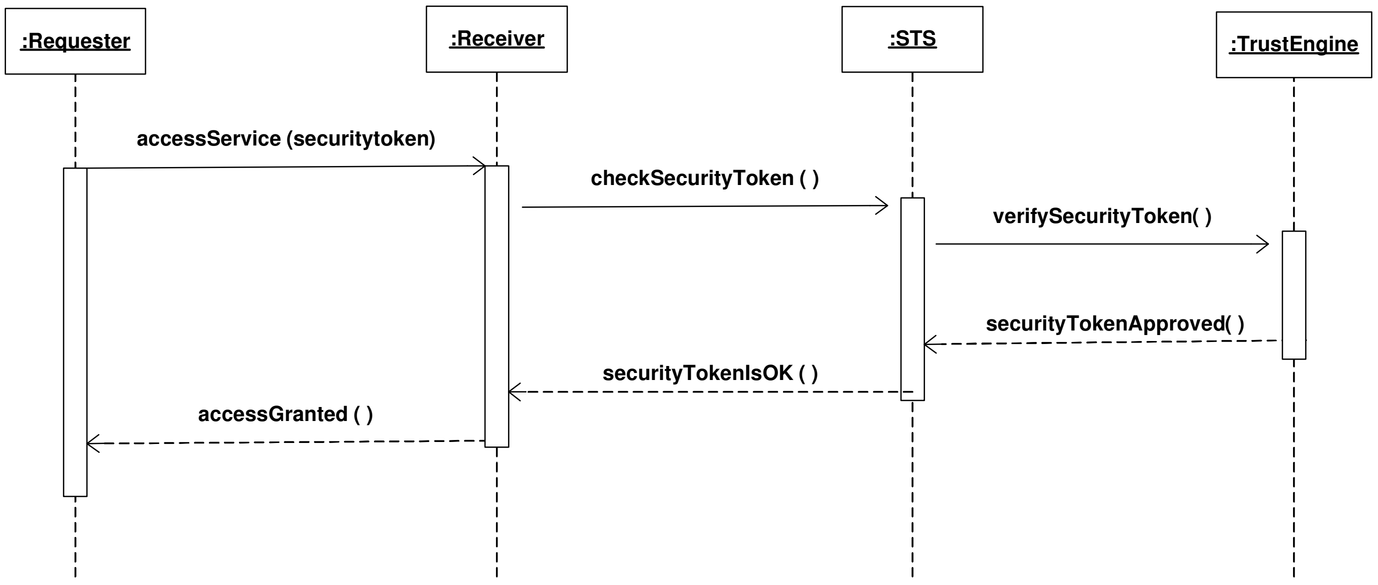


Figure 3: Sequence Diagram accessing a resource using a token

Implementation

In this solution, the concept of trust is realized by obtaining a security token from the web service (in our diagram, the Security Token Service) and submitting it to the receiver who in turn validates that security token through the same web service. Upon approval, the receiver establishes a valid trust relationship with the receiver that lasts as long as the security token is valid.

In order to assure effective implementation, we need to take in consideration the following:

- To communicate trust, a service requires proof, such as a signature to prove knowledge of a security token or set of security tokens. A service itself can generate tokens or it can rely on a separate STS to issue a security token with its own trust statement.

- Although the messages exchanged between the involved entities are protected by WS-Security; still three issues related to security tokens are possible: security token format incompatibility, security token trust, and namespace differences. The WS-Trust pattern addresses these issues by defining a request/response protocol (in which the client sends *RequestSecurityToken* and receives *RequestSecurityTokenResponse)* and introducing a Security Token Service (STS) which is another web service.

- Based on the credential provided by the requester, there are different aspects of requesting a security token (RST), each of which has a unique format that the requester should follow:

- The issuance process: formed as *RequestSecurityToken (RequestType, Claims).*This is our use case Create a security token in the Dynamics section.
- The renewal process: formed as *RequestSecurityToken (RequestType, RenewTarget).*
- The cancel process: formed *RequestSecurityToken (RequestType, CancelTarget).*
  By the way, the cancelled token is no longer valid for authentication and authorization.
- The validate process: formed as *RequestSecurityToken (RequestType, ValidateTarget).*

.

The WS-Trust specification was created as part of the Global XML Web Services Architecture (GXA) framework, which is a protocol framework designed to provide a consistent model for building infrastructure-level protocols for web services and applications [Box02]. It was authored by Microsoft, IBM, Verisign, and RSA Security and was approved by OASIS as a standard in March 2007.

Example Resolved

Ajiad now has the ability to automate its trust relationships with its partners by managing the registration tasks for all its partners and issuing customers a unique ID's. In this case, *Ajiad* provides a mediator between the customers and its participant partners and plays the role of negotiator and third-party player who is trying to satisfy both sides.

Ajiad now can offer a Security Token Service for its business partners, who may find useful ways to take advantage of credit processing and other services offered by *Ajiad*, which now has new business opportunities.

Consequences

The WS-Trust pattern presents the following *advantages:*

- ***Security.*** By extending the WS-Security mechanisms, we can handle security issues such as security tokens (the possibility of a token substitution attack), and signing (where all private elements should be included in the scope of the signature and where this signature must include a timestamp).

- ***Trust.*** With this solution, we have the choice of implementing the WS-Policy framework to support trust partners by expressing and exchanging their statements of trust. The description of this expected behavior within the security space can also be expressed as a trust policy.

- ***Confidentiality.*** We can achieve confidentiality of users' information. Since Policy providers now can use mechanisms provided by other web services specifications such as WS-Security [ibm09b] to secure access to the policy, XML Digital Signature [w3c08] to authenticate sensitive information, and WS-Metadata Exchange [w3c09].

- All the security tokens exchanged between the involved parties are signed and stamped with unique keys that are known only to the recipients.

- ***Time validity***. We can specify time constraints in the parameters of a security token issued by STS. This constraint will specify for how long that security token is valid. Upon expiring, the security token's holder may renew or cancel it.

The WS-Trust pattern presents the following *liabilities:*

- The efficiency of WS-Trust may suffer from the repeated round-trips for multiple token requests. We need to make an effort to reduce the number of messages exchanged.

- The WS-Trust standard is a lengthy document and several details were left to avoid making the pattern too complex. The interested reader can find more details in the WS-Trust Standard web page [oas09].

Known Uses

- DataPower's XS40 XML Security Gateway [dat05] is a device for securing web services that provides web services access control, message filtering and field-level encryption. It centralizes policy enforcement, supporting standards such as WS-Security, WS-Trust, WS-Policy and XACML.
- SecureSpan™ XML Firewall [lay09] enforces WS* and WS-I standards to centralize security and access requirements in policies that can be run as a shared service in front of applications.
- Vordel Security Token Service [vor09] is used to issue security tokens and to convert security tokens from one format to another. The security tokens created by an STS are bound to the messages travelling between web services..
- PingTrust, a standalone WS-Trust Security Token Server [pin06] creates and validates security tokens that are bound into SOAP messages according to the Web Services Security (WSS) standard.

Related Patterns

- The *Trust* Analysis Pattern, [Fay04]. The objective of this pattern is to provide a conceptual model that embodies the abstract aspects of trust to make it applicable to different domains and applications.

- The *Credential* Pattern [Mor06]. This pattern addresses the problem of exchanging data between trust boundaries and how to resolve the problem of authenticating and authorizing a principal's identity over different systems.

- The *Circle of Trust* pattern allows the formation of trust relationships among service providers in order for their subjects to access an integrated and more secure environment [Del07]. The WS-Trust pattern could be used to establish trust between providers.

3. Conclusions

This pattern describes how to build trust relationshsips and how existing trust relationships may be used as the basis for brokering trust through the creation of security token issuance services. These security token issuance services build on WS-Security to transfer the requisite security tokens in a manner that ensures their integrity and confidentiality.

Future work will include designing patterns for other web services standards such as WS-Federation and WS-SecureConversation that depend on WS-Trust as a prerequisite foundation. This will give us a good chance to analyze and discover how WS-Trust fits with other web services standards and how much it could simplify the implementation of theses specifications in real-life business applications.

Acknowledgements

We thank our shepherd Takao Okubo for his comments. We also thank our PC member Yann-Gael Gueheneuc for his discussion about the value of patterns describing standards.

References

[Box02] D. Box, *Understanding GXA,* Microsoft Corporation, http://msdn.microsoft.com/en-us/library/aa479664.aspx - Last accessed on December 15, 2009

[dat05] IBM Corporation, WebSphere DataPower XML Security Gateway XS40, http://www-01.ibm.com/software/integration/datapower/xs40/ – Last accessed at November 25, 2009

[Del07] N. Delessy, E.B.Fernandez, and M.M. Larrondo-Petrie, "A pattern language for identity management", *Procs. of the 2nd IEEE Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007),* March 4-9, Guadeloupe, French Caribbean.

[Fay04] M.E.Fayad, and H. Hamza, "The Trust Analysis Pattern*", in Proceedings of the Fourth Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP 2004),* Porto Das Dunas, Ceara, Brazil. August 10-13, 2004. http://sugarloafplop2004.ufc.br/acceptedPapers/ww/WW_1.pdf - Last accessed on December 15, 2009

[Fer06] E.B.Fernandez and N. Delessy, ""Using patterns to understand and compare web services security products and standards", *Proceedings of the Int. Conference on Web Applications and Services (ICIW'06*), Guadeloupe, February 2006. IEEE Comp. Society, 2006.

[ibm09a] Security in a Web Services World: A Proposed Architecture and Roadmap, http://download.boulder.ibm.com/ibmdl/pub/software/dw/library/ws-secmap.pdf - Last accessed on December 3, 2009

[ibm09b] IBM Corporation, Web Services Security 2004, http://www.ibm.com/developerworks/library/specification/ws-secure/ – Last accessed on December 07, 2009

[lay09] Layer 7 Technologies, The SecureSpan XML Firewall, http://www.layer7tech.com/main/products/xml-firewall.html – Last accessed on December 09, 2009

[Mad03] WS-Trust: Interoperable Security for Web Services, by Paul Madsen, http://www.xml.com/pub/a/ws/2003/06/24/ws-trust.html - Last accessed on November 30, 2009

[Mor06] P. Morrison and E. B. Fernandez, "The credentials pattern*", in Proceedings of the 2006 conference on Pattern languages of programs (PLoP 2006),* Portland, OR, USA. October 21–23, 2006. http://portal.acm.org/citation.cfm?id=1415472.1415483 - Last accessed on December 15, 2009

[oas06] OASIS, Web Services Security: (WS-Security 2004), http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf - Last accessed on December 15, 2009

[oas09] OASIS Standard, WS-Trust 1.4, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.pdf - Last accessed on December 07, 2009

[pin06] Ping Identity Corporation, PingTrust, a standalone Security Token Server, http://www.pingidentity.com/about-us/news-press.cfm?customel_datapageid_1173=1404 - Last accessed on December 15, 2009

[vor09] Vordel Limited, Vordel STS, http://www.vordel.com/solutions/security_token_services.html - Last accessed on December 15, 2009

[w3c07] W3C, Web Services Policy 1.5 – Framework, 4 September 2007, http://www.w3.org/TR/ws-policy/- Last accessed on December 15, 2009

[w3c08] W3C Working Group, XML Signature Syntax and Processing (Second Edition) 2008, http://www.w3.org/TR/ws-gloss/ – Last accessed on December 15, 2009

[w3c09] W3C Working Draft 2009, Web Services Metadata Exchange, http://www.w3.org/TR/ws-gloss/ – Last accessed on December 15, 2009

A Worm misuse pattern

Eduardo B. Fernandez[1], Nobukazu Yoshioka[2], and Hironori Washizaki[3]

1 Dept. of Comp. Science and Eng., Florida Atlantic University, Boca Raton, FL, USA, ed@cse.fau.edu
2 National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan, nobukazu@nii.ac.jp
3 Waseda University / GRACE Center, National Institute of Informatics, 3-4-1, Okubo, Shinjuku-ku, Tokyo, Japan, washizaki@waseda.jp

Abstract

We have proposed a new type of pattern, the *misuse pattern*. This pattern describes, from the point of view of the attacker, how a type of attack or misuse is performed (what system units it uses and how), provides ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and helps analyzing the attack once it has happened by indicating where can we find forensics data as well as what type of data. A catalog of misuse patterns is needed to let designers evaluate their designs with respect to possible threats. We present here a misuse pattern for a generic worm, which describes the essential and typical characteristics of this type of malware. We consider how to stop this malware and we also discuss some examples and variations.

Introduction

In order to design a secure system, we first need to understand the possible threats to the system. Without this understanding we may produce a system that is more expensive than necessary, it is hard to administer, and has a large performance overhead. We have proposed a systematic approach to threat identification starting from the analysis of the activities in the use cases of the system and postulating possible threats [Bra08]. This method identifies high-level threats such as "the customer can be an impostor", but once the system is designed we need to see how the chosen components could be used by the attacker to reach her objectives. For this purpose we proposed the use of *misuse patterns* (which we called initially attack patterns) [Fer07]. A misuse pattern describes, from the point of view of the attacker, how a type of attack is performed (what units it uses and how), analyzes the ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and describes how to trace the attack once it has happened by appropriate collection and observation of forensics data. It also describes precisely the context where the attack may occur. We built a catalog of misuse patterns for VoIP [Pel09] and we characterized precisely some aspects of misuse patterns [Fer09]. We describe this type of patterns using a template based on the one used in [Bus96], which is commonly used for architectural patterns as well as security patterns. This catalog is not only useful to test a new system but also to evaluate an existing system.

To make misuse patterns of practical value we need a catalog of typical attacks. As we said above, until now we have only misuse patterns for VoIP environments, this is our first misuse pattern of a more general scope.

Worm

Intent
Propagate to as many places as possible (or to specific systems), usually indicating its presence, and maybe performing some damage.

Context
Sites connected through the Internet or another type of network. The Internet provides a variety of services such as email, file transfer, and web services (Figure 1). Any of these services can be used for propagation. Both fixed and wireless networks can be used by the worm. Portable storage devices such as memory sticks can also propagate worms.

Problem
A worm tries to take advantage of any input to invade a system. Users might open attachments carrying worms and some ports of a system may be unprotected or have vulnerabilities; all of these give the worm a chance to invade. Mail systems and file transfer systems for example, include lists of addresses which can be used by the worm to find places where to propagate. Many systems do not control access to their system directories and do not restrict Internet traffic, which facilitates a worm invasion.

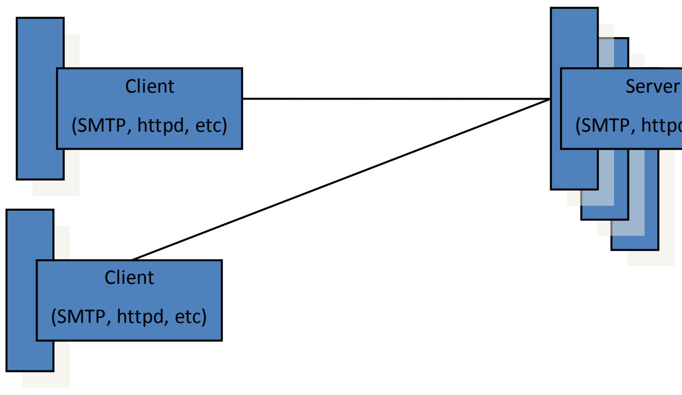


Figure 1. Context for worm propagation

The solution is affected by the following **forces :**

- *Objectives*. Its objectives may be political, monetary, or vandalism. A political worm typically tries to produce damage to an antagonist; a monetary worm tries to reach many places to collect information or drop spyware; a vandal worm tries to destroy or damage information.

- *Reach.* Try to reach as many places as possible or to specific sites. For most worms, reaching many places is a basic objective.

- *Presence manifestation.* Try to show its presence in the system so victims know about it. Exceptions to this are cases where the objective is to drop spyware.

- *Credit.* To embed an identification or mark so that the creator can take credit for it.

- *Misuse.* Perform some destruction and/or other misuses (confidentiality, integrity, or availability). The misuse may be delayed (time bomb).

- *Obfuscation.* Try to hide its structure to make harder its detection and removal.

- *Collateral damage*. In addition to specific misuses, the worm may require costly operations for its removal, stopping or disrupting business activities. Its propagation may affect the normal traffic in the network.

- *Latency*. Its propagation must be as fast as possible to avoid detection and countermeasures.

- *Activation.* This can be done by *e*nticing offers which may tempt users to open email attachments or download procedures (social engineering). Other possibilities are invading through unprotected ports or taking advantage of vulnerabilities.

Solution

Attach a core portion of the worm to email messages or to files. When the user opens the message attachments or executes the file the core of the worm starts executing. Alternatively, invade through an unprotected or flawed port. Download remaining portions from complementary network sites. Use some procedure to hide the structure of the worm. Perform its mission and propagate. Figure 2 shows the propagation of a typical worm; speed comes from a tree-like propagation.

Structure

Figure 3 shows a class diagram of the units involved. Class **Node** represents any node in the network, defined by its address (URL in the Internet). Any node can be the *origin* of a worm and any node can be its *target* (and be invaded). Some nodes are *complementary sites* from which commands or other parts of the worm may be retrieved. Class **Worm** represents the worm itself, including procedures for initial setup, to bring complementary parts, to hide the worm, to perform its mission, and to propagate.

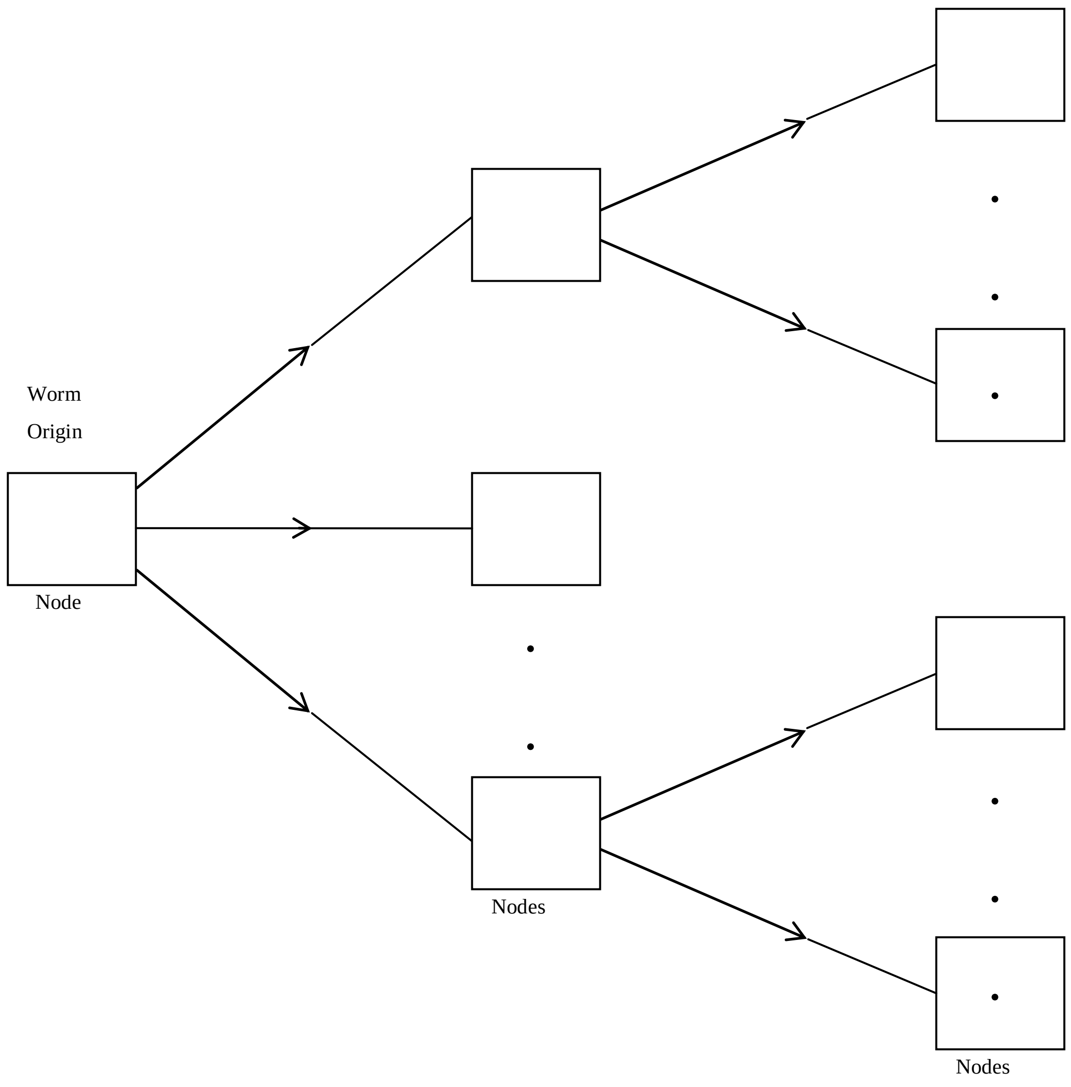


Figure 2 Worm propagation

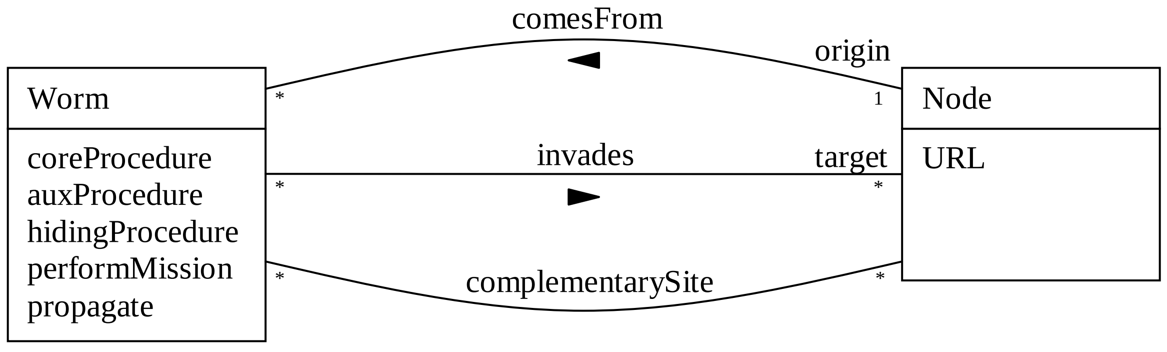


Figure 3. Class diagram for the Worm pattern.

Dynamics

Use cases for a worm may include Create a Worm, Remove a Worm, and Activate a Worm. Create and Remove are specific to the type of worm (see Variants). We describe here Activate a Worm because it is the most important for defenders. Its scenario (Figure 4) includes:

- *Triggering:* After the attacker sends a message, a target (user) may activate an executable procedure with a core part of the worm.
- *Assembly:* Download remaining parts via the Internet (optional)
- *Obfuscation*: Use some procedure to hide the parts of the worm, e.g. encryption or dispersion.
- *Address Search*: Find destination addresses as new targets for propagation. Addresses may also be generated randomly.
- *Manifestation*: Display some messages (optional)
- *Propagation:* Send the core part via the connection to another node in the address list. This operation is repeated for all the found or generated addresses.

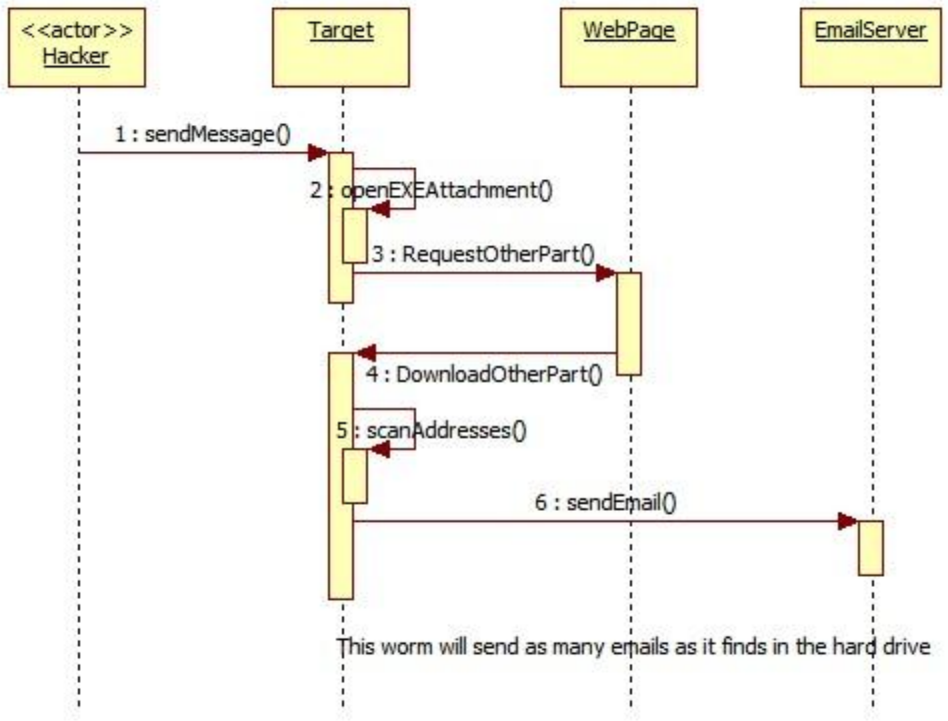


Figure 4. Sequence diagram for activating a worm

Variants
A *passive worm* requires a user to activate an executable program and it usually propagates through email. Melissa, ILOVEYOU, Anna Kournikova, and Bagle are examples of this type.

An *active worm* takes advantage of some system flaw to provoke a buffer overflow or another attack to get in through some port. It may scan looking for unprotected ports. Code Red is an active worm. Storm can be active or passive [Smi08].

A *virus* attaches itself to some program (infects an executable file) and when the user executes this program it gets activated. Jerusalem, Christmas, and Chernobyl are examples of viruses.

Some worms have several versions with different purposes; for example, Storm has variants that perform different types of misuses, including targeted spam and DDoS attacks [Smi08].

Some worms are *multimode* (multivector) worms, which can use a variety of ways to invade their targets; for example the Storm virus infects computers using multiple payloads [Smi08].

Known uses
Typical examples of worms include:

- *ILOVEYOU* [ILO, wor09]. This was an email attachment worm that appeared in 2000. It relied in social engineering to entice users to open the attachment. It also used specific weaknesses of Microsoft Windows. It propagated using the addresses in the address book of the mail system.

- *Bagle*. It was a mass-mailing worm written in assembly language [bag] and affecting all versions of Windows. After activation, it copies itself to the Windows system directory and downloads a SMTP engine to mail its core to other nodes as an attachment (see the Implementation section for its typical behavior).

- *Code Red* [Ber01]. It appeared in July of 2001. It propagated through port 80, indicated its presence by defacing web pages, propagated using a random IP address generator, and later would activate a denial of service attack from infected sites.

- *Nimda* [nim]. Nimda is a multivector worm that can use several ways to propagate: email, visiting an infected site, seeking out vulnerable servers to upload files, or through the network.

- *Slapper* [Arc03]. Can launch denial of service attacks. Propagates finding addresses in files. The nodes invaded by the worm communicate using a P2P protocol to collaborate in their misuses.

- *Conficker* [con09, wor09]. This is a multivector worm with an autoupdate facility (signed updates) and encrypted communications. It downloads parts of the worm from some Internet sites.

These worms are really worm types from where many variants can be derived. It is possible to define separate patterns for each type of the generic Worm pattern. For example, the Slapper worm and the Apache Scalper operate in a similar way [wor09], the Conficker is really a series of worms [wor09].

Implementation
We show a typical implementation of the Bagle worm. It follows very closely the sequence diagram of Figure 4. A scenario in a Microsoft environment would include:

- A user invokes an executable code by clicking a MS Word file, then automatically VBA macro code is interpreted.
- The worm downloads the remaining parts from a web server via the Internet.
- The worm finds target addresses in the Outlook address book using VBA and a SMTP server name from outlook settings.
- The worm displays some messages using a VBA function.
- The worm opens a SMTP connection to mail its core to the next target. This operation is repeated for all the found addresses.

Active worms take advantage of vulnerabilities such as buffer overflows and can get in through port 80 or unprotected ports. In the case of worms such as Code Red the core of the worm was sent to the input buffer of port 80 in Microsoft's IIS server [Ber01]. A virus or worm may send a web address link as an instant message to all the contacts of the invaded site and if the recipients answer, they bring the virus to their sites.

Consequences
This misuse has the following *advantages* for the attacker:

- *Objectives*. Its economic objectives can be reached if the worm has a long reach and clever social engineering. Its political objectives can be reached if the worm reaches the intended audience and manifests its presence and reasons. Its vandalism objectives can be obtained if the worm does considerable damage.

- *Reach.* If the system has easily accessible address lists the worm can find many new targets. Random address generation is not so effective.

- *Manifestation of its presence.* A good procedure for display can make its presence well noticed. This may intimidate its victims, which brings satisfaction to the attacker.

- *Credit.* The mark should be distinctive but not identify the attacker. The creator can get negative recognition for his effort.

- *Misuse.* A worm can perform destruction and/or other misuses (confidentiality, integrity, denial of service, drop spyware or spam).

- *Obfuscation.* Encryption and dispersion can make harder its detection and removal. Some worms mutate, i.e. they change their structure when they propagate.

- *Side effects*. A fast-propagating worm can produce a lot of traffic and if it is hard to detect its cost increases.

- *Latency*. A fast-propagating worm can do much damage before being stopped.

- *Activation*. Good ways to activate the worm are necessary since all its objectives depend on this step.

A worm also can have some *liabilities* for the attacker:

- A worm can be used to detect infected nodes or to destroy viruses or other worms.

Countermeasures
The following policies and their corresponding mechanisms (realized as patterns), can stop or mitigate the worm:

- *Policy about attachments*: Users should be trained to recognize trustable attachments and they should be forbidden to open unknown or suspicious attachments.

- *Need-to-know* policy to define access by system processes to resources. For example, address lists should use authorization to control access to their contents.

- *Control of network communications*: Connections should be established with only trusted addresses (control through the firewalls). This policy may avoid downloads from complementary sites.

- *Intrusion detection:* An IDS can detect some attacks in real time and alert the firewall to stop it.

- *Use of antivirus software*: Can help detect and clean worms after the fact

- *Backups.* Checkpointing files and keeping backup images of them is a fundamental precaution against data destruction or unauthorized modification.

- *Specialized hardware.* Process communication controls in the operating system can be enforced through specialized hardware [Shi00]. It is possible to define partitions in the operating system that can be enforced by hardware and will prevent a worm from performing its actions.

Forensics

The pieces of the worm may be scattered in different units within a site. The specific places to look for worm components depend on the specific variant or type of worm. The places where worms normally penetrate include mail attachments, files, unprotected ports, and these must be inspected. One should also look for the specific parts of the work, e.g. core procedure, obfuscation procedure, etc.

Web logs can help in finding parts that might have been downloaded. GUIs may have log records of the use of procedures to display the worm announcements. Units that contain addresses may contain indications of search.

Related patterns

- *Authorization and Reference Monitor*. These patterns together can prevent access to address lists, thus stopping the worm propagation [Sch06].
- *Firewall*. Can filter attempts to download further pieces of the worm [Sch06].
- *Intrusion Detection*. Can detect a worm invasion in real time and collaborate with the firewall to block its traffic [Fer05].

Acknowledgements

We thank our shepherd, Tsukasa Takemura, for his useful comments that significantly improved the quality of the paper. We also thank Eiiti Hanyuda for supervising our paper shepherding.

References

[Arc03] I. Arce and E. Levy, An analysis of the Slapper worm", *IEEE Security and Privacy*, Jan./Feb. 2003. 82-87.

[bag] "Bagle (computer worm), *http://en.wikipedia.org/wiki/Bagle_(computer_worm)*

[Ber01] H. Berghel, "The Code Red worm", *Comm. of the ACM*, vol. 44, No 12, December 2001, 15-19.

[Bra08] F. Braz, E.B.Fernandez, and M. VanHilst, "Eliciting security requirements through misuse activities" *Procs. of the 2nd Int. Workshop on Secure Systems Methodologies using Patterns (SPattern'07*). In conjunction with the 4th International Conference onTrust, Privacy & Security in Digital Business (TrustBus'07), Turin, Italy, September 1-5, 2008. 328-333.

[con] "Conficker", http://en.wikipedia.org/wiki/Conficker

[Fer05] E.B.Fernandez and A. Kumar, "A security pattern for rule-based intrusion detection", *Proceedings of the Nordic Conference on Pattern Languages of Programs, Viking PLoP 2005,* Otaniemi, Finland, 23-25 September 2005.

[Fer07] E.B. Fernandez, J.C. Pelaez, and M.M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool", *Procs. of the Third Annual IFIP WG 11.9 Int. Conf. on Digital Forensics*, Orlando, FL, Jan. 29-31, 2007. Chapter 24 in *Advances in Digital Forensics III*, P. Craiger and S. Shenoi (Eds.), Springer/IFIP, 2007, 345-357.

[Fer09] E.B. Fernandez, N. Yoshioka and H. Washizaki, "Modeling misuse patterns", *Procs. of the 4th Int. Workshop on Dependability Aspects of Data Warehousing and Mining Applications (DAWAM 2009),* in conjunction with the *4th Int.Conf. on Availability,* Reliability, and Security (ARES 2009). March 16-19, 2009, Fukuoka, Japan.

[ILO] "ILOVEYOU", http://en.wikipedia.org/wiki/ILOVEYOU

[Nim] "F-Secure Virus-descriptions:Nimda", http://www.f-secure.com/v-descs/nimda.shmtl

[Pel09] J. Pelaez, E.B.Fernandez, and M.M. Larrondo-Petrie, "Misuse patterns in VoIP", *Security and Communication Networks Journal*. Wiley, published online**:** 15 Apr 2009 http://www3.interscience.wiley.com/journal/117905275/issue

[Sch06] M. Schumacher, E. B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering,* Wiley 2006.

[Shi00] T. Shinagawa, K. Kono, T. Masuda, " *Exploiting Segmentation Mechanism for Protecting Against Malicious Mobile Code*", Tech. Report 00-02, Dept. of Information Science, University of Tokyo, May 2000.

[Smi08] B. Smith, "A Storm (worm) is brewing", *Computer,* IEEE February 2008, 20-22.

[wor09] "Worm evolution", May 2009, http://www.digitalthreat.net/?p=17

Design Decision Topology Model for Pattern Relationship Analysis

Kiran Kumar, Prabhakar T.V.
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur, India
{vkirankr, tvp}@iitk.ac.in

Abstract

Software design patterns are solutions to recurring design problems. Analyzing and managing the large and ever increasing number of design patterns is a problem. Non-uniform and incomplete pattern descriptions further complicate the task.

Existing literature defines different pattern relationship types and many relationships among patterns. These relationships are analyzed based on designer's experience and their formal basis is unclear. We propose a novel graph based model to capture the semantics of a design pattern using design decisions and their side-effects. The relationships are analyzed using various graph properties which enable automation of relationship analysis.

1. Introduction

A design pattern describes a particular recurring design problem that arises in a specific design context, and presents a well-proven generic scheme for its solution [9, 5]. Patterns are increasingly being used not only to capture and disseminate best practices, but also to turn named patterns into a shared vocabulary for expressing and communicating technical knowledge [9, 5, and 13]. The large number of existing and continuously increasing patterns (one source states that there are 250 patterns for Human-Computer interaction alone [19]) introduce new problems to designer who use them - like the management of a pattern knowledge base.

We propose a graph based model called Design Decision Topology Model (DDTM) to deal with the relationship analysis problem. The objective of this model is to reduce pattern *semantics* to *syntax*- a graph which delivers the pattern functionality (quality) through elementary functionality (quality) – nodes of the graph are elementary functional functionality and edges are dependencies. Conceptually, the *DDTM* technique is analogous to the *Decision view* [8, 16, 25] in the architecture domain. The utility of the *DDTM* for a pattern can be derived from the utility of Decision view for architecture. Researchers of architecture domain [8, 16, 25] propose *Decision views* to enable:

- Enriching architecture description
- Codifying crosscutting and intertwined design decisions present in multiple views.
- Traceability of quality requirements.
- Providing thumbnail or compact forms of the architecture.

Traceability and *Thumbnail* problems are considered important at pattern level also [6]. We apply architecture level techniques at pattern level to derive the DDTM of a pattern. This representation enriches pattern descriptions, helps analyze quality requirement traceability and relationships amongst patterns.

This model treats each pattern as a micro-architecture and defines the pattern as a topology of a set of design decisions. Using this model, different relationships are analyzed using graph properties. For example,

Patterns A and B are duplicates if Graph(A) $\equiv$ *graph(B),*

Pattern A comprises-of patterns B and C if Graph(B) $\subset$ *Graph(A) AND Graph(C)* $\subset$ *Graph(A).*

The rest of the paper is structured as follows: Section 2 provides the required background terminology. In section 3, we discuss briefly how a pattern is described as a DDTM. In section 4, we demonstrate the tactic topology model which is a kind of DDTM. Section 5 discusses related work, and section 6 concludes the paper suggesting some future directions.

2. Terminology

In this section, we review some software architecture terminology used in this paper.

- **Quality requirement** [27]: is a requirement which is not specifically concerned with the functionality of the software. Quality requirements specify the external constraints the software should meet.

- **Quality Attribute** [2]: is a set of related quality requirements.
- **Design Decision** [15]: is a strategy that is applied to solve a particular part of the problem.
- **Tactic** [2]: A tactic is a design decision that influences the control of a quality attribute parameter. For example, the *Increase available resources* design decision (upgrading 512 MB RAM to 1 GB RAM) controls (minimizes) the response time parameter.
- **Implications/Side-effect** [3, 25]**:** A design decision comes with many implications. For example, a design decision might introduce a need to make other decisions, create new requirements, or modify existing requirements; pose additional constraints to the environment. For example, the *Increase available resources* tactic which is an alternative to achieve *Reduce response time* quality requirement imposes side-effects like *Increase in cost*, *Change in resource management (scheduling) policy* etc.

	Relationship Type	Synonyms	Description
1	*Is-Duplicate-of*	--	Patterns A and B provide same solution to same problem. [14]
2	*Is-an-Alternative-to*	*Similar-to*	A and B are similar patterns, solving the same problem, but proposing different choices. [26, 17]
3	*Comprises*	• Uses • Is-made-of • Decomposes -into	When building a solution for the problem addressed by pattern A, one sub-problem is similar to the problem addressed by B. Therefore, the pattern A uses the pattern B in its solution. [26, 21, 17]
4	*Refines*	• Is-variant-of • Subsumes	Patterns A and B address same problem but pattern A provides more refined (with less side-effects) solution than B. [26, 17]

Table 1: Description of different Relationship types.

3. How to describe a pattern – the Design Decision Topology Model (DDTM)

Analyzing the relationships for a given set of patterns can be considered as 3-step process:

- Analyze design decisions of the patterns.
- Analyze the topology of the design decisions.
- Analyze the relationships from the topologies.

3.1. Analyze design decisions of the patterns.

The key-information of a pattern is usually embedded in the essential sections of the pattern-form/template; *Context*, *Problem*, *Solution, Consequences* sections are considered to be essential sections [9, 5, 6]. Additional sections such as *Implementation*, *Example* etc often appropriate to provide meaningful guidance on where a pattern applies and how to apply it [6]. Hence we use the key-information provided in *Problem*, *Solution, Consequences* sections as clues to analyze design decisions.

3.2. Analyze topology of design decisions.

DDTM provides a structure to the design decisions of a pattern by explicitly representing the dependency among them. DDTM primarily provides rationale for existence of a particular design decision. This information is modeled as edges among design decisions in our graph model. An edge A → B in DDTM means the side-effect of design decision A is resolved by design decision B.

The DDTM of a pattern can be viewed as a graph of design decisions. The *Intent* section specifies the primary design decision(s) of the pattern; they are modeled as source-nodes in the DDTM. Based on the implications of the current stage design decisions, the design decisions in the next stage are analyzed. For example, consider the *Master-slave* pattern [5]. The intent of *Master-slave* pattern is given as: *the master component distributes the work to slave components to support parallel computation*. It specifies *Work partitioning* as the primary design decision; and it is modeled as the source node in the DDTM of *Master-slave* pattern. One of the implications of *Work partitioning* design decision is the work distribution details need to be hidden from clients; hence *Restrict communication paths* design decision is used to overcome this implication. The continuation of the design process leads to an edge from *Work partitioning* to *Restrict communication paths* nodes in the DDTM of *Master-slave* pattern; the label on the edge represents the implication of *Work partitioning* design decision. Figure 1 shows a fragment of DDTM of *Master-slave* pattern.

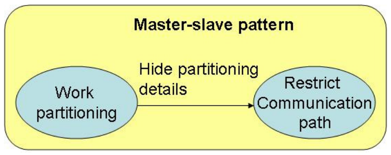


Figure 1: DDTM of Master-slave pattern.

	Relationship Type	Description
1	*Is-Duplicate-of*	Graph(P1) $\equiv$ Graph(P2). (Graph equivalence property)
2	*Is-an-Alternative-to*	Source-Node(P1) = Source-Node(P2) AND Graph(P1) $\neq$ Graph(P2).
3	*Comprises*	Graph(P2) $\subset$ Graph(P1). (Sub-graph property)
4	*Refines*	Source-Node(P1) = Source-Node(P2) AND Graph(P2) $\subset$ Graph(P1).

Table 2: Graph rules of different Relationship types.

3.3. Analyze relationships from the topologies.

Various relationships among patterns can be analyzed using some graph properties, when patterns are described with DDTM. The graph model we propose is currently applicable to analyze four relationship types (see Table 1):
(1) Is-Duplicate-of,
(2) Is-an-Alternative-to,
(3) Comprises, and
(4) Refines.

Table 2 describes how these relationship-types map into properties of graphs.

4. Evaluation

This section demonstrates the utility of DDTM. We define *Tactic Topology Model (TTM)*, an instance of DDTM where the design decisions are tactics. Using TTM we describe some patterns such as *Abstract factory*, *Builder*, *MVC*, *Observer*, and *Publisher-Subscriber* to analyze relationships between them.

4.1. Tactics as design decisions of a pattern – Tactic Topology Model (TTM).

Conceptually, there can be a variety of design decisions that can model the semantics of patterns. Choosing tactics as design decisions has these benefits:

- DDTMs can be communicated easily when its design decisions refer to a standard body of knowledge such as tactics.
- Tactics are classified according to different quality attributes; this allows easy indexing of a tactic when its quality requirement is known.

Tactic Topology Model (TTM) is a kind of DDTM where the design decisions of a pattern are captured through tactics which are more elementary than patterns. Intuitively, if a pattern provides a solution to achieve multiple primitive quality requirements, a tactic provides a solution to achieve a single primitive quality requirement [2].

Bass et al define a catalogue of tactics [2, 4] for various quality attributes. This catalogue seems insufficient to precisely capture the semantics of the considered patterns. Also, Bass et al explicitly mention in [2] that "*the list of tactics is necessarily incomplete*". We defined an additional set of tactics to completely model the tactic topologies for the considered patterns. Table 3 lists the tactics along with the quality requirements they achieve and the quality attributes of the quality requirements.

Bass et al tactics		
Tactic	**Quality requirement**	**Quality Attribute**
Apply Polymorphism	Variant modules need to be exchangeable at runtime.	Substitutability
Restrict communication paths	Hide a set of modules/services.	Modifiability
Maintain Semantic Coherence	High-level decomposition of an application.	Modularity
Maintain Multiple views	Handle multiplicity in user-interface requirements.	Usability
Parameterize representative	Abstraction over variant modules.	Extensibility
Register at runtime	Dependents of an object are known at runtime.	Adaptability
Additional tactics		
Compose whole from parts	Represent module groups.	Maintainability
Notify modification	State change in one object requires state change in other objects.	Adaptability
Enumerate representatives	Abstraction over variant modules.	Extensibility

Table 3: Tactics used in pattern topologies.

4.2. Analyze tactics from pattern description.

The problem part of the pattern provides information of some of its design decisions in the form of its *benefits*. The described benefits or quality requirements are used as clues to recover some of its constituent tactics by mapping the pattern benefits upon the quality requirements of the tactics.

The UML structure in the solution part of the pattern also provides information of some of its design decisions. UML templates of tactics are used to analyze the tactics from pattern UML structure.

In this section, we provide details of analyzing the inherent tactics for the Observer pattern - tactic analysis for other patterns can be found at http://www.cse.iitk.ac.in/users/vkirankr/Patterns_to_Tactics.doc.

Table 4 illustrates the tactic analysis of Observer (a GoF [9]) pattern. Table 5 illustrates the tactic analysis of MVC (a POSA1 [5]) pattern.

It is to be observed from Tables 4 and 5 the differences in pattern description templates of GoF and POSA1 patterns. The template of GoF patterns contains *Intent*, *Motivation*, *Applicability*, *Structure*, *Participants*, *Collaborations*, *Consequences*, *Implementation*, *Sample Code*, *Known Uses*, and *Related Patterns* sections. Whereas, the POSA1 patterns template contain *Intent*, *Example*, *Problem*, *Solution*, *Structure*, *Dynamics*, *Implementation*, *Variants*, *Known Uses*, *Consequences*, *See Also* sections. Although these two templates differ in naming conventions, the two templates are nearly similar to each other. Henninger et al [14] also discusses the similarities and differences among GoF, POSA, and PLML templates.

Applicability section of Observer Pattern description		
Requirement	**Elaboration of requirement**	**Achieved through tactic(s)**
"*When a change to one object requires changing others.*"	State change in one object requires state change in other objects.	Notify modification
"*You don't know how many objects need to be changed.*"	Dependents of an object are known at runtime.	Register at runtime.
"*When an object should be able to notify other objects without making assumptions about who these objects are. In other words, you don't want these objects tightly coupled.*"	Variant modules need to be exchangeable at runtime.	Apply Polymorphism
Consequences section of Observer Pattern description		
"*Abstract coupling between Subject and Observer.*"	Variant modules need to be exchangeable at runtime.	Apply Polymorphism
"*Support for broadcast communication.*"	Variant modules need to be exchangeable at runtime.	Apply Polymorphism

UML diagram of Observer Pattern	
Tactic	**UML Textual form from [9]** (↑ means inheritance)
Register at runtime	Subject.attach(), Subject.detach()
Notify modification	Subject.notify()
Apply Polymorphism	**Instance 1:** • Subject, ConcreteSubject • Subject ↑ ConcreteSubject **Instance 2:** • Observer, ConcreteObserver • Observer ↑ ConcreteObserver

Table 4: Analysis of tactics for Observer pattern [9].

Problem section of MVC Pattern description		
Requirement	**Elaboration of requirement**	**Achieved through tactic(s)**
"*Different users place conflicting requirements on the user interface.*"	Handle multiplicity in user-interface requirements.	Maintain Multiple views
"*Building a system with the required flexibility is expensive and error-prone if the user interface is tightly interwoven with the functional core.*"	High-level decomposition of an application.	Maintain Semantic Coherence

“*The same information is presented differently in different windows, for example, in a bar or pie chart*.”	Handle multiplicity in user-interface requirements.	Maintain Multiple views
“*Changes to the user interface should be easy, and even possible at run-time*.”	State change in one object requires state change in other objects.	Notify modification
“*Supporting different 'look and feel' standards or porting the user interface should not affect code in the core of the application*.”	High-level decomposition of an application.	Maintain Semantic Coherence
“*The display and behavior of the application must reflect data manipulations immediately*.”	State change in one object requires state change in other objects.	Notify modification
Consequences section of MVC Pattern description		
“*Multiple views of the same model*.”	Handle multiplicity in user-interface requirements.	Maintain Multiple views
“*Synchronized views*.”	State change in one object requires state change in other objects.	Notify modification
“*'Pluggable' views and controllers*.”	Dependents of an object are known at runtime.	Register at runtime
“*Exchangeability of 'look and feel'*.”	Variant modules need to be exchangeable at runtime.	Apply Polymorphism

Solution section of MVC Pattern description	
Solution description	**Tactic**
“*The MVC architectural pattern comprises three types of participating components: clients, model, views, and controllers*.”	Maintain Semantic Coherence
“*There can be multiple views of the model*.”	Maintain Multiple views
“*The separation of the model from view and controller components allows multiple views of the same model*.”	Maintain Semantic Coherence
“*If the user changes the model via the controller of one view, all other views dependent on this data should reflect the changes. The model therefore notifies all views whenever its data changes*.”	Notify modification
“*The change-propagation mechanism maintains a registry of the dependent components within the model*.”	Register at runtime
“*Changes to the state of the model trigger the change-propagation mechanism*.”	Notify modification

UML diagram of MVC Pattern description	
Tactic	**UML Textual form from [5]** (↑ means inheritance)
Maintain Semantic Coherence	Model, View, Controller
Maintain Multiple views	View
Notify modification	Model.notify()
Register at runtime	Model.attach(), Model.detach()
Apply Polymorphism	• Observer, View, Controller • Observer ↑ View • Observer ↑ Controller

Table 5: Analysis of tactics for MVC pattern [5].

4.3. Analyze topology of tactics.

Figures 2 through 6 illustrate the tactic topologies of Observer, Abstract factory, Builder, MVC, and Publisher-Subscriber patterns respectively. Table 6 provides the intent of these patterns for ready reference. Here, we discuss the topology analysis of Observer pattern; other topologies can be analyzed in the similar way.

The intent of the Observer pattern (refer table 6) specifies the design decision *Automatic update notification* as the primary design decision; hence *Notify modification* tactic is made as the source node in Observer TTM.

References to dependent modules are to be known and *all dependent modules need to implement a uniform interface* in order to notify an update to its dependents. *Register at runtime* and *Apply Polymorphism* tactics achieve the above two quality requirements respectively. Hence, the implication of *Notify modification* tactic provides the rationale for using *Register at runtime* and *Apply Polymorphism* tactics. The TTM of Observer pattern depicted in Figure 5 illustrates the tactic dependency.

4.4. Analyze relationships from topologies.

When patterns are represented as tactic topologies, the relationships among patterns can be analyzed by applying the graph rules given in Table 2.

It is to be noted that the label on the edges in TTM represent the rationale of the dependency between the two tactics. In order to improve understandability, the rationale for the same pair of tactics may differ from one pattern to other. But the rationale can be equalized at some higher level of abstraction. Hence, mismatch in the rationale does not affect the relationships among patterns.

Using these rules, the following relationships can be inferred from figures 5 through 9:

- Abstract factory (Figure 3) and Builder (Figure 4) have the same source node but TTMs are different. Hence from Table 2 Abstract factory is *is-an-Alternative-to* Builder pattern. Zimmer [26] comes to the same conclusion independently.
- The TTM of MVC pattern (Figure 5) includes TTM of the Observer pattern (Figure 2). Hence MVC *comprises-of* the Observer pattern. Avgeriou et al [1] and Buschmann et al [5] propound the same relationship.
- Publisher-Subscriber pattern (Figure 6) *refines* Observer pattern (Figure 2). This relationship is also mentioned by Avgeriou et al [1].

4.5 Utility of the TTM: The TTM is a decision view of a design pattern and is a useful description of how the pattern works.

Traceability: The direct linkage from a quality requirement to its corresponding UML fragment is missing in the pattern description. Analyzing tactics from the description bridges the gap between quality requirement and UML fragment and eases the traceability analysis. For example, consider a quality requirement of *Observer* pattern (Table 4) - *don't know how many objects need to be changed***.** This requirement is mapped to *Register-at-runtime* tactic. Analyzing the UML fragment of *Register at runtime* tactic - *Subject.attach()*, *Subject.detach()*, we obtain the traceability link between the quality requirement - *don't know how many objects need to be changed,* and its UML fragment - *Subject.attach()*, *Subject.detach()*.

Relationships: By comparing the TTMs of *Observer* and *MVC*, we were able to infer that *MVC* uses *Observer*. Similarly as shown in Section 4.4, *Publisher-Subscriber* refines *Observer*.

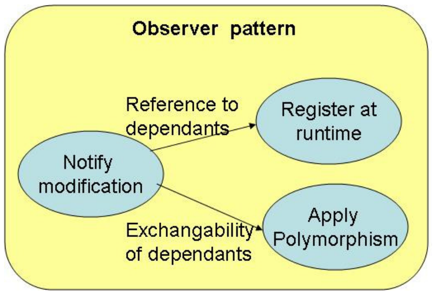


Figure 2: TTM of Observer pattern.

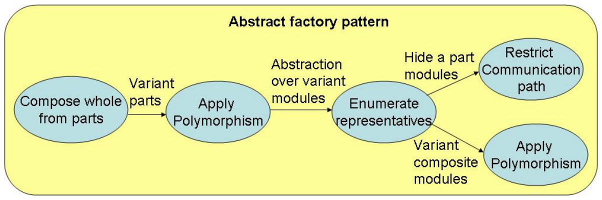


Figure 3: TTM of Abstract factory pattern.

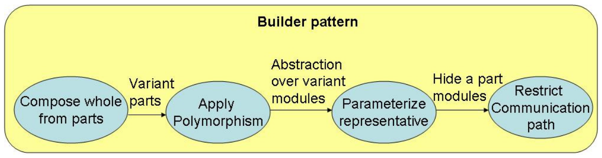


Figure 4: TTM of Builder pattern.

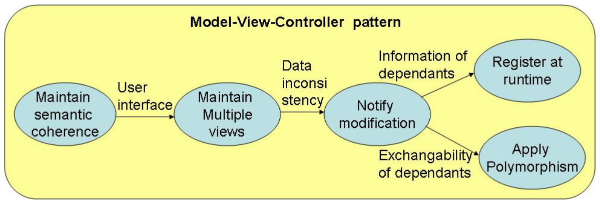


Figure 5: TTM of MVC pattern.

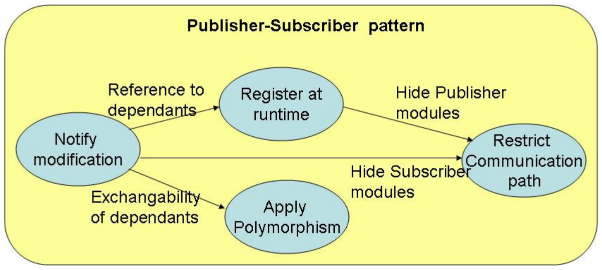


Figure 6: TTM of Publisher-Subscriber pattern.

5. Related Work

Pattern relationship analysis, a sub-problem of the pattern ontology problem, could help in managing the large number of patterns. Zimmer [26] defined three types of relationships and analyzed some relationships between design patterns. Similarly, Avgeriou et. al. in [1] analyzed relationships between architectural patterns. Those relationships were analyzed based on their experience with little or no formal support. Currently, many pattern relationships are specified informally through semantics-free "related to" relationship type [11, 12], but this level of abstraction is unsatisfactory for a designer. Noble in [21] defined and classified various pattern relationship types into primary and secondary categories based on designer needs. Kruchten in [17] defined various relationship types between design decisions.

Analyzing design alternatives (patterns) for a given set of requirements is considered as a knowledge-intensive task [31]. Providing tool support to manage a patterns knowledge base will significantly improve the productivity of the designer. VanHilst and colleagues [32] propose a novel knowledge formalization technique called *Multi-dimensional Concern Matrix* to model the knowledge of *Security* patterns along various stakeholder concerns. This paper also discusses various issues like *Primary dimensions*, *Secondary dimensions* etc to model pattern knowledge. The other advantages of this technique are: gaps in the problem space that lack pattern coverage can be identified easily, and the model is easily extensible to add new dimensions of concern. Our *DDTM* (or TTM) knowledge modeling technique is orthogonal to *Multi-dimensional Concern Matrix* technique. TTM addresses the queries related to relationships between tactics, patterns (and quality requirements) such as:

- What are the patterns which use *Restrict communication path* tactic to improve *Security*? – *Authentication proxy* pattern.
- What are the patterns which use *Compose from parts* tactic? – *Composite*, *Whole-part*, *Abstract factory, Builder* patterns.

It is to be noted that these queries can be modeled using multiple dimensions as: *Tactic*, *Relationship type*, *Quality requirement*.

Similar to our UML templates to analyze tactics, Riele [33] defined UML collaboration templates for patterns to analyze patterns from existing designs of frameworks such as *JUnit*, *Geo system*, *KMU Desktop*, and *JHotDraw*. Riele also defines *Design pattern density* metric to measure what percentage of the framework design can be analyzed through pattern instances. This metric is used to discuss the indications on framework maturity such as: *As the framework matures, the Design pattern density increases* etc.

Non-uniform pattern descriptions [14, 20, 7, 22] and the large number of patterns [14, 13, 19] complicate the problem of pattern relationship analysis. What we need is a uniform pattern description and a formal basis for the analysis of pattern relationships.

Uniform pattern description with natural language is clearly an impractical solution. Modeling pattern semantics through traditional UML semantics seems to be insufficient, modeling patterns precisely using UML is under research [28, 29, 30]. Describing patterns with formal approaches such as eLePUS [23], DiSCO [18], and BPSL [24] is best suited for code generation but not for relationship analysis [14]. Describing patterns as set of property-value pairs [10] enables automation of relationship analysis, but defining a set of properties which is consistent and complete for all the patterns is very difficult.

In recent years, software architecture researchers [25, 16 and 8] proposed that documenting design decisions as first class entities overcomes the *architecture knowledge vaporization* problem.
Currently, for pattern relationship analysis, different formal pattern descriptions exist, such as: eLePUS [23], DiSCO [18], BPSL [24], signs [22], mathematical structures [13], OWL-DL [13]. But these languages are not designer's languages and are hard to use for a designer. Our model helps the designer in ease of describing patterns, enabling the designer to use design decision vocabulary.

6. Conclusions and Future work

. We observe that a decision view of a pattern is useful to analyze relationships amongst patterns. A design decision topology model (DDTM), a decision view, reduces the semantics of a pattern to some syntactic properties of a graph. We propose a particular kind of DDTM, called Tactic Topology Model (TTM) which models design patterns using tactics. This TTM is constructed from the pattern description and its UML diagram. We discussed how our model helps analyze quality requirement traceability and relationships amongst patterns.

Analysis of our Pattern TTMs converges to the same conclusions (relationships between patterns) as described in literature. Our mechanism is amenable for automation and should enable discovery of new relationships.

The work presented in this paper, addresses a sub-problem of a designer's decision support system – ontology of design patterns.

7. Acknowledgements

We are very grateful to our shepherd Eduardo B. Fernandez who had tirelessly read and re-read many versions of the paper and improved both the form and content. Program Committee Member Eric Platon gave several useful and important suggestions which greatly helped. The shepherding process is indeed a very useful practice.

	Pattern	Intent
1	Observer.	Notify the updated state to its dependents automatically, when object state dependency exists between multiple objects.
2	Abstract factory	Separate the representation of a product family from the representation of products. Provide the common interface of product families to create its products and hide the representation of products.
3	Builder	Separate the representation of a complex object from representation of parts. Provide a common construction process to create different complex object representations.
4	Model-View-Controller.	An interactive application is divided into three components – model (core functionality and data), views (user interface), and controllers (handling user input). State changes in a view or in the model are notified to the other views.
5	Publisher-Subscriber.	One publisher notifies any number of subscribers about changes to its state. Publishers register themselves to a broker and subscribers discover publisher from broker.

Table 6: Some patterns from [5, 9].

8. References

[1] P. Avgeriou and U. Zdun, "*Architectural patterns revisited - a pattern language*", In Proceedings of 10th European Conference on Pattern Languages of Programs 2005.

[2] L. Bass, P. Clements, and R.Kazman, "*Software Architecture in Practice*", Second Edition. Addison-Wesley 2003.

[3] Bass. L., Klein. M. and Bachmann. F, "*Quality Attribute Design Primitives and the Attribute Driven Design Method*", In: Proceedings of the Product Family Engineering vol. 4, Springer-Verlag, Berlin.

[4] F. Bachmann, L. Bass, and R. Nord, "*Understanding and Achieving Modifiability in Software Architecture*", CMU/SEI-2007-TR-002.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "*Pattern-Oriented System Architecture: A System of Patterns*", John Wiley & Sons, 1996.

[6] F. Buschmann, K. Henney, D. C. Schmidt, "*Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages*". Wiley & Sons, 2007.

[7] J. Dietrich, and C. Elgar, "*Towards a Web of Patterns*", Proc. Semantic Web Enabled Software Engineering (SWESE), 117-132, Galway, Ireland, 2005.

[8] Dueñas, J.C. and Capilla, R, "*The Decision View of Software Architecture*", Proceedings of the 2nd European Workshop on Software Architecture (EWSA 2005), Springer-Verlag, LNCS 3047, pp. 222-230 (2005).

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "*Design Patterns: Elements of Reusable Object-Oriented Software*", Addison-Wesley, 1994.
[10] S. Hasso and C.R. Carlson, "*Linguistics-based Software Design Patterns Classification*", In Proceedings of the Thirty-Seventh Annual Hawaii International Conference on System Science (HICSS-37). IEEE Computer Society Press, 2004.

[11] S. Henninger, "*An Organizational Learning Method for Applying Usability Guidelines and Patterns*", in Engineering for Human-Computer Interaction (revised papers, EHCI 2001), vol. LNCS 2254, Springer, 2001, pp. 141-155.

[12] S. Henninger, and P. Ashokkumar, "*An Ontology-Based Infrastructure for Usability Design Patterns*", Proc. Semantic Web Enabled Software Engineering (SWESE), Galway, Ireland, pp. 41-55, 2005.

[13] S. Henninger, and P. Ashokkumar, "*An Ontology-Based Metamodel for Software Patterns*", In Proceedings of 18th Int. Conf. on Software Engineering and Knowledge Engineering 2006.

[14] S. Henninger, V. Corrêa, "*Software Pattern Communities: Current Practices and Challenges*", Pattern Languages of Programs (PLoP 07), (submitted), 2007.

[15] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "*Generalizing a model of software architecture design from five industrial approaches*", In Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA5), Pittsburgh, Pennsylvania, 2005.

[16] A. G. J. Jansen and J. Bosch, "*Software architecture as a set of architectural design decisions*", In Proceedings of WICSA 5, pages 109-119, November 2005.

[17] P. Kruchten, "*An ontology of architectural design decisions in software intensive systems*", In 2nd Groningen Workshop on Software Variability, pages 54-61, December 2004.

[18] T. Mikkonen, "*Formalizing Design Patterns*", Int'l Conf. Software Engineering, pp. 115-124, 1998.

[19] S. Montero, P. Díaz, and I. Aedo, "*A Semantic Representation for Domain-Specific Patterns*", Int'l Symp. on Metainformatics, U. K. Wiil, Ed., Springer-Verlag, LNCS 3511, 2005, pp. 129-140.

[20] J. Noble, "*Towards a Pattern Language for Object-Oriented Design*", Proc. of Technology of Object-Oriented Languages and Systems (TOOLS Pacific), 28, IEEE Comp. Soc., pp. 2-13, 1998.

[21] James Noble, "*Classifying relationships between object-oriented design patterns*", In Australian Software Engineering Conference (ASWEC), pages 98-107, 1998.

[22] James Noble, and Robert Biddle, "*Patterns as Signs*", Proceedings of the 16th European Conference on Object-Oriented Programming, p.368-391, June 10-14, 2002.

[23] S. Raje, and S. Chinnasamy, "*eLePUS-A Language for Specification of Software Design Patterns*", Proc. 2001 ACM Symp. Applied Computing, pp. 600-604, 2001.

[24] T. Taibi, and D. C. Ling Ngo, "*Formal Specification of Design Patterns - A Balanced Approach*", Journal of Object Technology, 2(4), pp. 127-140, 2003.

[25] Tyree, J. and Akerman, A, "*Architecture Decisions: Demystifying Architecture*", IEEE Software, vol. 22, no 2, pp. 19-27, (2005).

[26] Walter Zimmer, "*Relationships Between Design Patterns*", J. Coplien and D. Schmidt, editors, Pattern Languages of Program Design, pages 345_364. Addison-Wesley, 1995.

[27] G. Kotonya, I. and Sommerville. "*Requirements Engineering: Processes and Techniques*". John Wiley & Sons, 1998.

[28] G. Sunyé, F. Pennaneac'h, W.M. Ho, A. L. Guennec, and J. M. Jézéquel, "*Using UML action semantics for executable modeling and beyond.*" In Proceedings of the 13th International Conference on Advanced Information Systems Engineering, pages 433-447, 2001.

[29] Robert B. France, Dae-Kyoo Kim, and Sudipto Ghosh, Eunjee Song, "*A UML-Based Pattern Specification Technique*". IEEE Transactions on Software Engineering, pages 193 – 206, 2004.

[30] J. K. H. Mak, C. S. T. Choy, and DPK Lun. "*Precise Modeling of Design Patterns in UML.*" In Proceedings of the 26th International Conference on Software Engineering, pages 252-262, 2005.

[31] Pierre N. Robillard, "*The role of knowledge in software development*", Journal of Communications of the ACM, pages 87-92, 1999.

[32] VanHilst Michael, Fernandez Eduardo B, and Braz Fabricio, "*A Multi-dimensional Classification for Users of Security Patterns*", Journal of Research and Practice in Information Technology, pages 87-97, 2009.

[33] Dirk Riehle, "*Design pattern density defined*", Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications, pages 469-480, 2009.

Learning Patterns: A Pattern Language for Creative Learners II

Takashi Iba

Faculty of Policy Management, Keio University

MIT Center for Collective Intelligence, Massachusetts Institute of Technology

Toko Miyake

Faculty of Environment and Information Studies, Keio University

Abstract

In this paper we present a pattern language for learners who want to learn better without killing their creativity. In order to tell a 'knack' about the way of learning we apply the method of pattern language, which was originally proposed in architectural design and became famous in software design. Our proposed pattern language for creative learners, which we named "Learning Patterns", consists of 40 patterns. Each pattern is described in the same format; pattern number, pattern name, introduction, illustration, context, problem, forces, solution, actions, and related patterns. Although Learning patterns were originally developed in order to support learning of university students, we think it can be applied to any learners in various situations like engineering, business, science, and everyday life due to their fine abstract descriptions as a pattern language. In this paper, we show the overview of 40 patterns and four patterns in detail. Note that other five patterns have been presented in our previous paper at PLoP09 (Iba, *et. al.*, 2009).

1. Introduction

In recent complex society, it is essential to find problems and think of solutions from various point of view with a creative mind. People need to learn ability to practice their ideas, and create new viewpoints and ways of thinking. It is also necessary to construct their own living knowledge based on their situation, not just by memorizing existing information[1]. Under present circumstances, a few people can realize such a creative way of learning, but others do not seem to know how to do so.

As it is well known in the scenes of education, there is a difficult problem how one can teach how to learn. While it is quite easy to show the guideline to follow, it may shut learners out of the chance for thinking their own way of learning themselves. Furthermore, there is another difficulty to provide appropriate guideline for all learners who are under various situations. So, is it possible to provide something to help the learners under various situations to think their way of learning? In this paper, we would like to provide a solution for these problems.

In the following sections, we present a pattern language for learners in order to share several `knacks' against the way of creative learning. It means that we refer the mind and the writing format of pattern language into "learning design", as well as architectural design (Alexander 1977), software design (Beck and Cunningham. 1987; Gamma, *et. al.* 1995), organizational design (Coplien and Harrison 2004; Manns and Rising 2005), and pedagogical design (Anthony 1996; Bergin 2000).

We think that a pattern language is good way to help the student to design their learning, because it focuses on providing a new view for the reader so that they can think. It is quite important that the method is not easy way to get the result without thinking themselves. It is not, however, irresponsible way to leave all up to individual ability. It is considered as the way that tolerates individual ability while making a good use of abstract rules of past experience. The patterns are mainly for the learners, but they are also for the educators. The patterns will become a good tool for sharing the way of thinking.

[1] Against the backdrop, "Project-based Learning" (PBL) and "Learning by doing" are spotlighted, however they are the method to manage the classes or workshops that are usually closed. Although their aim is related to us, our aim is to support learning in everyday life rather than controlled classroom.

2. The Prehistory of the Learning Patterns

Learning patterns, which are presented in this paper, were developed by Learning Patterns Project, Keio University. We have handed out the catalog booklet of learning patterns to undergraduate students (Figure 1). The catalog was handed out to approximately 3,600 students of two faculties: Faculty of Policy Management and Faculty of Environment and Information Studies. These faculties have implemented a unique curriculum that is interdisciplinary and non-graded. It means all undergraduate students can study any kind of academic areas, for example social innovation, public policy, global strategy, environment, life sciences, and information studies, without reference to their grades and experience. Therefore the students should design their own learning, and it is the reason why we made the learning pattern for supporting learning design.

Figure 1: Catalog Booklet of Learning Patterns

3. Patterns Overview

Learning Patterns consist of 40 patterns. Figure 2 shows the overview of the whole language of the learning patterns. Learning patterns is organized in three layers according to the abstract level. In the top layer, there is a root pattern: *Learning Design (0).* This pattern provides an introductory explanation about why and how to use Learning Patterns. Such an explanation is usually provided outside patterns, however we put it in patterns as a self-referential pattern. In the second layer, there are three fundamental patterns: _Making Opportunities for Learning (1)_, _Creative Learning (2)_ and _Open-Process Learning (3)._ These patterns show important attitudes that summarize more specific patterns in next third layer. In the third layer, there are thirty-six patterns as concrete `knack' of learning: _Tornado of Learning (4)_, _Academic Excitement! (5)_, and so on.

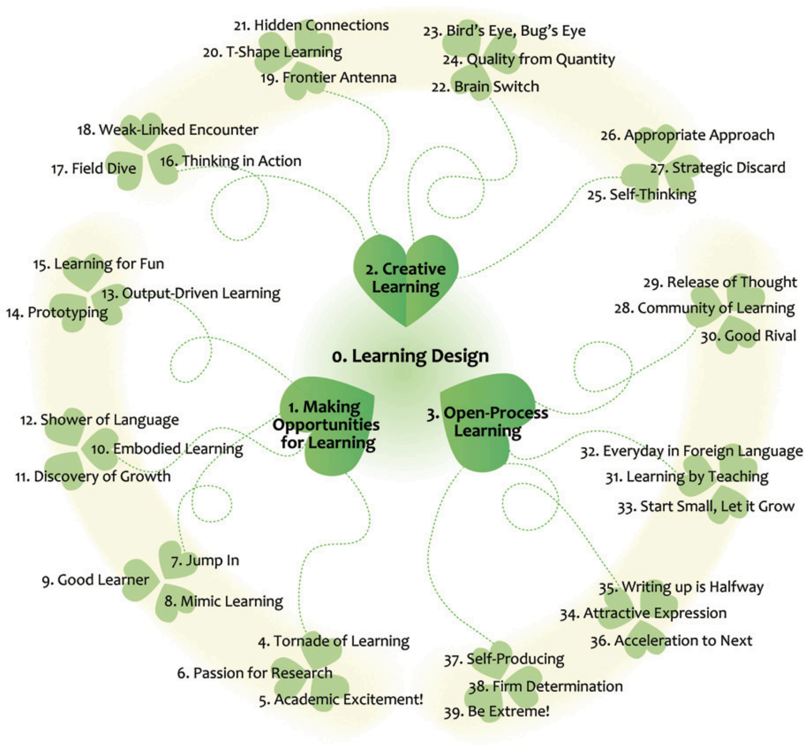


Figure 2: Overview of Learning Patterns

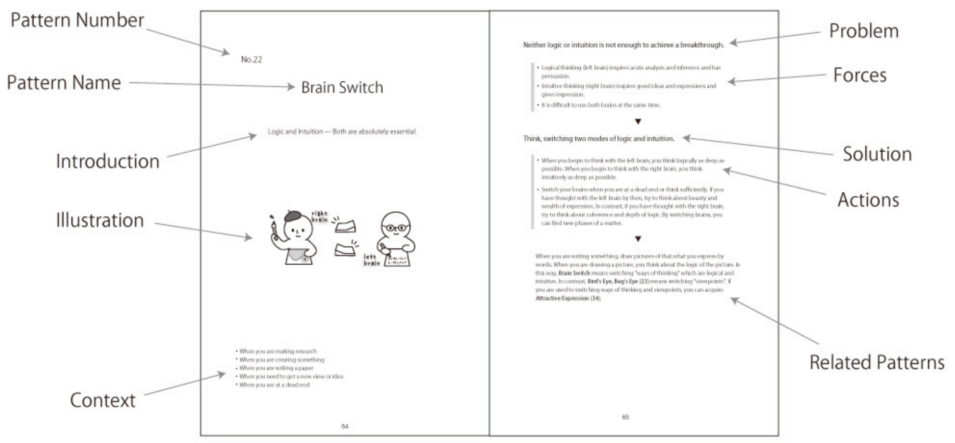
4. Pattern Format

Learning patterns are described in the format which consists of following items; "Pattern Number", "Pattern Name", "Introduction", "Illustration", "Context", "Problem", "Forces", "Solution", "Actions", "Related Patterns." Especially in the catalog of learning patterns, each pattern is printed in a double page spread (Figure 3), which is handed out for university students, as I mentioned above.

In the first half of pattern, which is printed at the left page in the catalog, the overview of the pattern is described. At first, Pattern Number is sequential number. Pattern Name is named as attractive and memorable phrase. Next, Introduction and Illustration is provided in order to help for the reader to imagine the meaning of the pattern lively. Then, there is a list of when the reader can use the pattern as Context. The reader can search his/her necessary pattern from his/her context with using the context navigation.

In the last half of pattern, which is printed at the right page in the catalog, the detail of the pattern is described. At first, Problem that is often occurred is described. Problem is emphasized in bold type. In succession to Problem, Forces are written as laws that are not able to or difficult to be changed. The difficulty to solve the problem comes from the existence of these forces, because your solution needs to meet all of them. After the Forces, the separator is placed. Next, Solution is written in bold type. Then, in the part of Actions, more concrete advice like examples or alternatives is introduced. After the Actions, the separator is placed again. At the last, Related Patterns are provided. Good learning is effectively achieved by combining some patterns. The reader can understand the meaning of the pattern deeper through reading the section of Related Patterns.

Note that it was a conscious choice not to show item names, such as “Problem” and “Solution”, in the format because of readability for students. Our format is midst between Alexander’s patterns and the design patterns.


No.22

Brain Switch

Logic and Intuition — Both are absolutely essential.

- When you are making research
- When you are creating something
- When you are writing a paper
- When you need to get a new view or idea
- When you are at a dead end

64

Neither logic or intuition is not enough to achieve a breakthrough.

- Logical thinking (left brain) inspires acute analysis and inference and has persuasion.
- Intuitive thinking (right brain) inspires good ideas and expressions and gives impression.
- It is difficult to use both brains at the same time.

▼

Think, switching two modes of logic and intuition.

- When you begin to think with the left brain, you think logically as deep as possible. When you begin to think with the right brain, you think intuitively as deep as possible.
- Switch your brains when you are at a dead end or think sufficiently. If you have thought with the left brain by then, try to think about beauty and wealth of expression. In contrast, if you have thought with the right brain, try to think about coherence and depth of logic. By switching brains, you can find new phases of a matter.

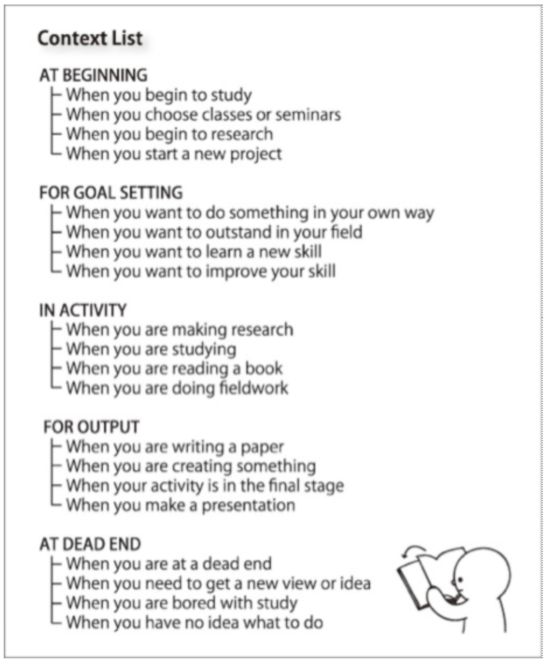
▼

When you are writing something, draw pictures of that what you express by words. When you are drawing a picture, you think about the logic of the picture. In this way, **Brain Switch** means switching "ways of thinking" which are logical and intuitive. In contrast, **Bird's Eye, Bug's Eye** (23) means switching "viewpoints". If you are used to switching ways of thinking and viewpoints, you can acquire **Attractive Expression** (34).

65

Figure 3: Pattern Format of Learning Patterns

In the catalog booklet, there is some navigation to find the patterns. One of the navigation is based on contexts of patterns. There are five categories of contexts: “at beginning”, “for goal setting”, “in activity”, “for output”, and “at dead end” (Figure 4). Each category consists of four contexts, which indicate to related patterns respectively. Therefore the reader can find patterns that are relevant to their situation.



Context List

AT BEGINNING
- When you begin to study
- When you choose classes or seminars
- When you begin to research
- When you start a new project

FOR GOAL SETTING
- When you want to do something in your own way
- When you want to outstand in your field
- When you want to learn a new skill
- When you want to improve your skill

IN ACTIVITY
- When you are making research
- When you are studying
- When you are reading a book
- When you are doing fieldwork

FOR OUTPUT
- When you are writing a paper
- When you are creating something
- When your activity is in the final stage
- When you make a presentation

AT DEAD END
- When you are at a dead end
- When you need to get a new view or idea
- When you are bored with study
- When you have no idea what to do

Figure 4: Context List of Learning Patterns

Another navigation in the catalog is provided in association with the curriculum of our university. Each course indicates to related patterns, therefore the student can find the patterns that are relevant to the classes they are taking.

5. An Example of Usage

Take, for example, a pattern of <u>*Acceleration to Next (36)*</u>. This pattern is supposed to be needed under the context like “When you are researching” or “When you are studying”. The problem frequently occurred is “It not seldom happens that people slack off their efforts subconsciously just before the goal”, and the solution is “Set next goal and pass through the current goal without slow down” (See more details in the next section of this paper).

The student who read this pattern may find new idea to design his / her learning activities, because pattern languages can become concept to comprehend the reality and amplify the ability of recognition. Thus, the method of pattern language provides the way to understand the existence of problem and the clue of the solution. Moreover, by virtue of the name of each pattern, it is getting easier to mention some aspect of learning. With using the example above, the teacher can advice the student with using the pattern name; “Don’t forget Acceleration to Next! ”. Otherwise, student can ask the teacher “Do you think I should increase Acceleration to Next?” Like this, pattern languages contribute to increase the vocabulary about learning among teachers and students. These are the reason why pattern language is called “language” of patterns.

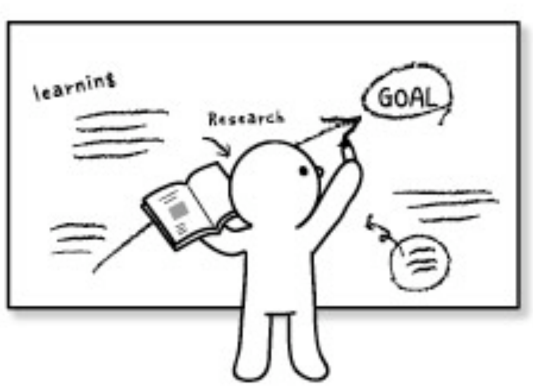
6. Four Learning Patterns

Here we take four patterns as examples: *Learning Design (0)*, *Brain Switch (22)*, Community of Learning *(28)*, and *Acceleration for Next (36).*

No.0

Learning Design

Design your learning.



➢ Always when you want to learn

It is not easy to learn "how to learn", while it is essential ability in complex and liquid society.

- Human is not able to learn everything because the time and memory is limited.
- There are several ways to study.
- People who learn effectively have a knack for good learning, which is independent on their fields or themes.

▼

Learn the 'knack' of learning from the experienced learners, and design your way of learning based on them.

- You can work on your activity with "learning patterns" which tells you the knack of effective learning.
- First, read roughly whole patterns to understand what "learning patterns" is like, especially the first half of each pattern; pattern name, introduction, illustration, and context. It is better to remember the pattern name and the illustration.
- Read the detail of patterns in which you are interested. In the last half of each pattern, there are description of "problem", difficulties why the problem is a hard to solve as "forces", "solution", and "actions" which are for solving the problem.
- You can find a learning pattern according to your situation with using the list of "context".
- Use "pattern name" of learning pattern as a common language, when you talk about learning with other students or teachers.

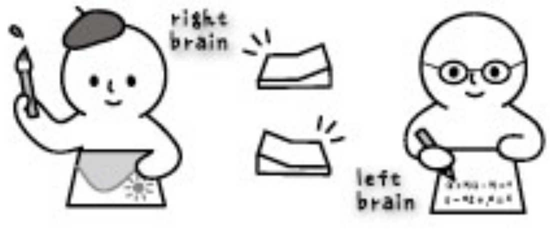
▼

Learning Design is important to do *Making Opportunities for Learning (1)*. For cultivating the opportunity, keep the tips of *Creative Learning (2)* in mind, and you can learn with excitement. *Open-Process Learning (3)* helps you recognize the significance of communities that you are in.

No.22

Brain Switch

Logic and Intuition —— Both are absolutely essential.



- When you are making research
- When you are creating something
- When you are writing a paper
- When you need to get a new view or idea
- When you are at a dead end

Thinking tends to be leaning to only logic or intuition, which each is not enough to achieve a breakthrough.

- Logical thinking (left brain) inspires acute analysis and inference, and has persuasion.
- Intuitive thinking (right brain) inspires good ideas and expressions, and gives impression.
- It is difficult to use both modes of thinking at the same time.

▼

Think, switching two modes of logic and intuition.

- When you begin to think with your "left brain", think logically as deep as possible. When you begin to think with your "right brain", think intuitively as deep as possible.
- Switch your brains when you are at a dead end or think sufficiently. If you have thought with "left brain" by then, try to think about beauty and wealth of expression. For example, when you are writing something, draw pictures of what you express by words. In contrast, if you have thought with "right brain", try to think about coherence and depth of logic. For example, when you are drawing a picture, think about the logic of the picture. By switching brains, you can find a new aspect of a matter.

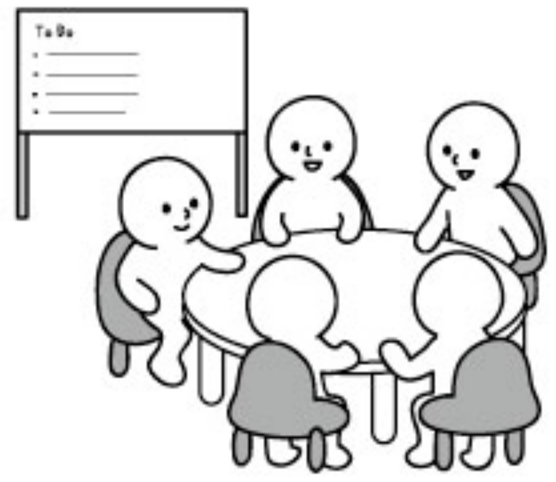
▼

Brain Switch means switching "ways of thinking" that are logical and intuitive. In contrast, *Bird's Eye, Bug's Eye (23)* means switching "viewpoints". If you are used to switching ways of thinking and viewpoints, you will provide *Attractive Expression (34)*.

No.28

Community of Learning

It is not necessary to study alone.



- When you begin to research
- When you begin to study
- When you are bored with study
- When you want to learn a new skill
- When you want to improve your skill

Individual's capacity is limited.

- The time we can spend is limited.
- Everyone has his/her own knowledge and viewpoints.
- By getting various viewpoints, human can deepen their understanding.
- Human can work harder with partners together than do alone.

▼

Find people with common objectives and build a community of learning to stimulate each other.

- First of all, plan for making "community of learning". For example, you plan that what kind of workshop or research project you can do.
- Gather some members of your surroundings who are interested in your plan. Then, you launch your project and make some concept and rough schedule with them.
- Decide how you show your efforts of your learning. For example, you can make a paper, and publish it on the web site or have a conference. This makes you keep your motivation.
- On the basis of this plan, accept applications for your "community of learning", preferably on a large scale beyond your acquaintances. It may be that you can gather more members who have similar interest than those of your acquaintances. Moreover, with the member of people who you don't know, you can avoid loose meeting and keep focusing on.
- To keep the member's motivation, confirm what you have done with each other regularly. If is necessary, you should reset your goal.

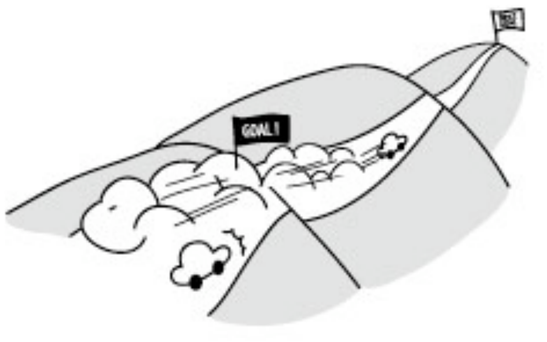
▼

There are no fixed rolls such as "teacher" or "student" in *Community of Learning*. So, all participants have opportunities for *Release of Thought (29)* and *Learning by Teaching (31)* as well as learning from others. One may find *Good Rival (30)* in the community. Your *Tornado of Learning (4)* based on your interests grows together with the other member's, and grows bigger and bigger. As a result, you will encounter *Academic Excitement! (5)* more and more.

No.36

Acceleration to Next

Just before the goal, people tend to press the brake pedal subconsciously.
Now is the time to set next goal and press down on the accelerator.



- When you are making research
- When you are writing a paper
- When you are creating something
- When your activity is in the final stage
- When you are at a dead end

It not seldom happens that people slack off their efforts subconsciously just before the goal.

- Just before achieving the goal, human tend to lose their motivation unconsciously.
- Finishing work is always tough.
- Human can work hard in active, just in the process of pursuing our goal.

▼

Set next goal and pass through the current goal without slow down.

- Think of the meaning of your activity, and imagine what you should do after achieving its goal.
- Set the next goal on the extension of your activity temporarily, and consider the immediate goal as a passing point. With an image of bigger goal, you can avoid losing the end work.

▼

Even if you work on your activity with the mind of *Passion for Research (6)* and *Firm Determination (38)*, it seems be tough to finish the work before its goal. When you are in the situation like this, it is important to put *Acceleration to Next*. Work on your activity with *Bird's eye, Bug's eye (23)* for taking the immediate goal as a part of next big goal. With "Bird's eye" for looking down a whole of your project, think of what you can do now, and take next goals. In this way, if you keep on progressing while you take your goals as passing points, you will be *Be Extreme! (39)*.

Appendix: Other Learning Patterns

Here, we shall show a summary of all 40 patterns together form a language for creative learning. We begin with the part of the language that defines learning design itself. This is the root and premise to use this pattern language;

0.	Core	*Learning Design*

Next, we shall go through the part of the language that gives fundamental attitudes for learning;

1.	Fundamental Attitudes	*Making Opportunities for Learning*
2.		*Creative Learning*
3.		*Open-Process Learning*

Now we start the part of the language that tells how you can achieve to learn more actively in detail. This part can be roughly divided into twelve groups of patterns, where each group consists of three patterns respectively.

4.	For Motivation and Fundamental Aspect	*Tornado of Learning*
5.		*Academic Excitement!*
6.		*Passion for Research*
7.	Key to Start	*Jump in*
8.		*Mimic Learning*
9.		*Good Learner*
10.	Acquire and Improve the Skill	*Embodied Learning*
11.		*Discovery of Growth*
12.		*Shower of Language*
13.	Make Your Learning More Interesting	*Output-Driven Learning*
14.		*Prototyping*
15.		*Learning for Fun*

16.	For Active Effort	*Thinking in Action*
17.		*Field Dive*
18.		*Weak-Linked Encounter*
19.	Scope of Learning	*Frontier Antenna*
20.		*T-Shape Learning*
21.		*Hidden Connections*
22.	For Innovative Thinking	*Brain Switch*
23.		*Bird's Eye, Bug's Eye*
24.		*Quality from Quantity*
25.	The Way of Going about Activity and Learning	*Self-Thinking*
26.		*Appropriate Approach*
27.		*Strategic Discard*
28.	Social Aspect of Learning	*Community of Learning*
29.		*Release of Thoughts*
30.		*Good Rival*
31.	Improve the Skill or Works	*Leaning by Teaching*
32.		*Everyday in Foreign Language*
33.		*Start Small, Let it Grow*
34.	Idea for the Final Phase of Activity	*Attractive Expression*
35.		*Writing up is Halfway*
36.		*Acceleration to Next*
37.	Strategy for the Medium and Long Term	*Self-Producing*
38.		*Firm Determination*
39.		*Be Extreme!*

The sequence presented here is not only one possible sequence, because “A pattern language has the structure of a network” (Alexander 1977). We can capture and trace the relation among the patterns in many ways. This is related to one of Alexander's significant findings that the design of a building and a town cannot be reduced to the structure of tree, but can be considered as semi-lattice.

Acknowledgment

We would like to do our best to thank you for other members of Learning Pattern Project at Keio SFC; Tsuyoshi Kato, Yuji Kobayashi, Kazeto Shimonishi, Natsumi Yotsumoto, Mariko Hanabusa, Mayu Iida, Mami Sakamoto, and Miyuko Naruse. Wonderful pattern language and this paper would not have happened without their collaboration. We want to thank to workshop participants who gave us great advices for previous paper at PLoP09. Furthermore, we are grateful to Yuji Yamano for shepherding in AsianPLoP2010. In the last, we are grateful to Christopher Alexander for inventing the idea of pattern languages and taking a great step to open collaboration.

References

Alexander, C., S. Ishikawa, and M. Silverstein. *A Pattern Language*. Oxford University Press, 1977.

Alexander, C., *The Timeless Way of Building*. Oxford University Press, 1979.

Alexander, C., H. Davis, J. Martinez, and D. Corner. *The Production of Houses*. Oxford University Press, 1985.

Anthony, D. L. G. "Patterns for classroom education" in J. M. Vlissides, J. O. Coplien, and N. L. Kerth, editors, *Pattern Languages of Programming 2*. AddisonWesley, 1996.

Beck, K. and W. Cunningham. "Using pattern languages for object-oriented programs", in *OOPSLA-87 workshop on the Specification and Design for Object-Oriented Programming*, 1987.

Bergin, J. "Fourteen pedagogical patterns". In *European Conference of Pattern Languages of Programs*, 2000.

Coplien, J. O. and N. B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice Hall, 2004.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

Iba, T., Miyake, T., Naruse, M., and Yotsumoto, N. Learning Patterns: A Pattern Language for Active Learners. In *16th Conference on Pattern Languages of Programs*, 2009.

King, I. F., *Christopher Alexander and Contemporary Architecture: a+u Architecture and Urbanism, August 1993 Special Issue*. a+u Publishing, 1993.

Kobayashi, Y., M. Yoshida, A. Sasaki, and T. Iba. Research patterns: A pattern language for academic research. In *15th Conference on Pattern Languages of Programs*, 2008.

Manns, M. L. and L. Rising. *Fearless Change: Patterns for Introducing New Ideas*. Addison-Wesley, 2005.

Naruse, M., Y. Takada, Y. Yumura, K. Wakamatsu, and T. Iba. Project patterns: A pattern language for promoting project. In *15th Conference on Pattern Languages of Programs*, 2008.

Metamorphosis – A Successful Organizational Change Management Pattern

Table of Contents

1

Abstract

Organization change management initiatives of various kinds including process changes, organization structure changes and policy changes are often met with a lot of apprehension and skepticism. To ensure the success of the change in an organization, a certain series of actions need to be in place and this usually tends to have a particular pattern. Successful change management initiative is an attempt through this pattern. This paper is an attempt to demonstrate a pattern that will help successfully implement, change management initiatives in an Organization

In our organizational context as in our personal lives, change is an essential part. Rather than meeting change head on and resisting it, we should rather embrace it and move along in the changed context. Establishing this pattern will help us individually and as an organization deal with change in a suitable manner.

Metamorphosis originates from biology and has a connotation of drastic transformation, like in the case of a larva turning to a pupa and then transforming itself into a butterfly, which has little or no resemblance to the previous phase. However for a butterfly to exist, it has to pass through the stages of Pupa and larva.

Metamorphosis is an organizational pattern; every Organization is greater than sum of its parts. We have focused on the "big picture" or the "forest" rather than granular areas viz."Individual Trees". As in nature, a large change cannot be attributed to few changes; it is often a result of large number of small changes.

Likewise, the actions proposed and smaller things achieved through this pattern, will not have a direct resemblance to the final outcome, we believe the outcome will be colorful and successful.

2

Introduction

After some natural caution and hesitation, a number of people get enthusiastic about a change in the organization (this could be various type of changes including organization structure, new process initiation, etc.), throw themselves into this new style of working and things begin to move on. The work force begins to smell success in the environment of change and they can now see the causes of many chronic problems.

The team begins with greater sense of self confidence, a belief that can actually nail some long standing issues. Most people they speak to, already know about the right things to do but they speak about how lack of time, ownership and lack of right direction is hindering them doing the right thing. The organization gives a mandate to some people to independently report the causes of issues.

However, some people in position begin to get uncomfortable with the situation; although they are too anxious to see the results. People who are used to evaluating and screening information before it is reported see this as publicizing embarrassing shortcomings; some they are aware of but haven't got the time to fix the issues themselves.

This eventually leads to a cascading situation where the changes are eventually even on the verge of being rolled back.

2.1. Problem

The move to the new building is announced, everyone wants to be in the new building. The management expects for such a simple task people have the common sense to make their way over to the new building allocate spaces, move furniture, hook phones and all these to happen over the weekend. What is the likelihood of this project being successful? None. The shift may happen but with lot of chaos.
Whether we are looking at an office move or a major organizational change; unless someone takes responsibility for masterminding the whole affair, either nothing will happen or there will be complete chaos.
Organizations go through turmoil, while institutionalizing processes, Tools, Policies, Frameworks etc. which meet with partial success, soured relationships and submissive compliance.
Success is never an accident: it is always the result of high intention, sincere effort, intelligent direction and skillful execution. It represents wise choice of many alternatives.
"There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order to things." – Niccolo Machiavelli

3

Solution

Newton's First Law of Motion states **"A body continues to maintain its state of rest or of uniform motion unless acted upon by an external unbalanced force"**. Arguably, no other law holds good more for a change management initiative. As human beings, our tendency to be comfortable in continuing with "things as they are" is similar to that of a body described by Newton's first law. For a change management initiative, it is critical that an external unbalanced force is provided to ensure that the initiative kicks off and is carried out successfully. Many analogies can be provided for this and one such analogy would be that of a satellite. To position a satellite in a geo-synchronous orbit, continuous rocket propellers are used till the satellite reaches its orbit. A change management initiative requires similar continuous "external unbalanced forces" to make it successful.

3.1. Change Management

"The only thing constant in life is change itself". A successful change management initiative requires various conscious steps to be carried out to ensure its success. We believe that a change management initiative is akin to a machine that has a motor driving various gears – these gears along with the motor working together ensure that the initiative chugs forward. The figure below indicates our depiction of the change management initiative as a pattern:

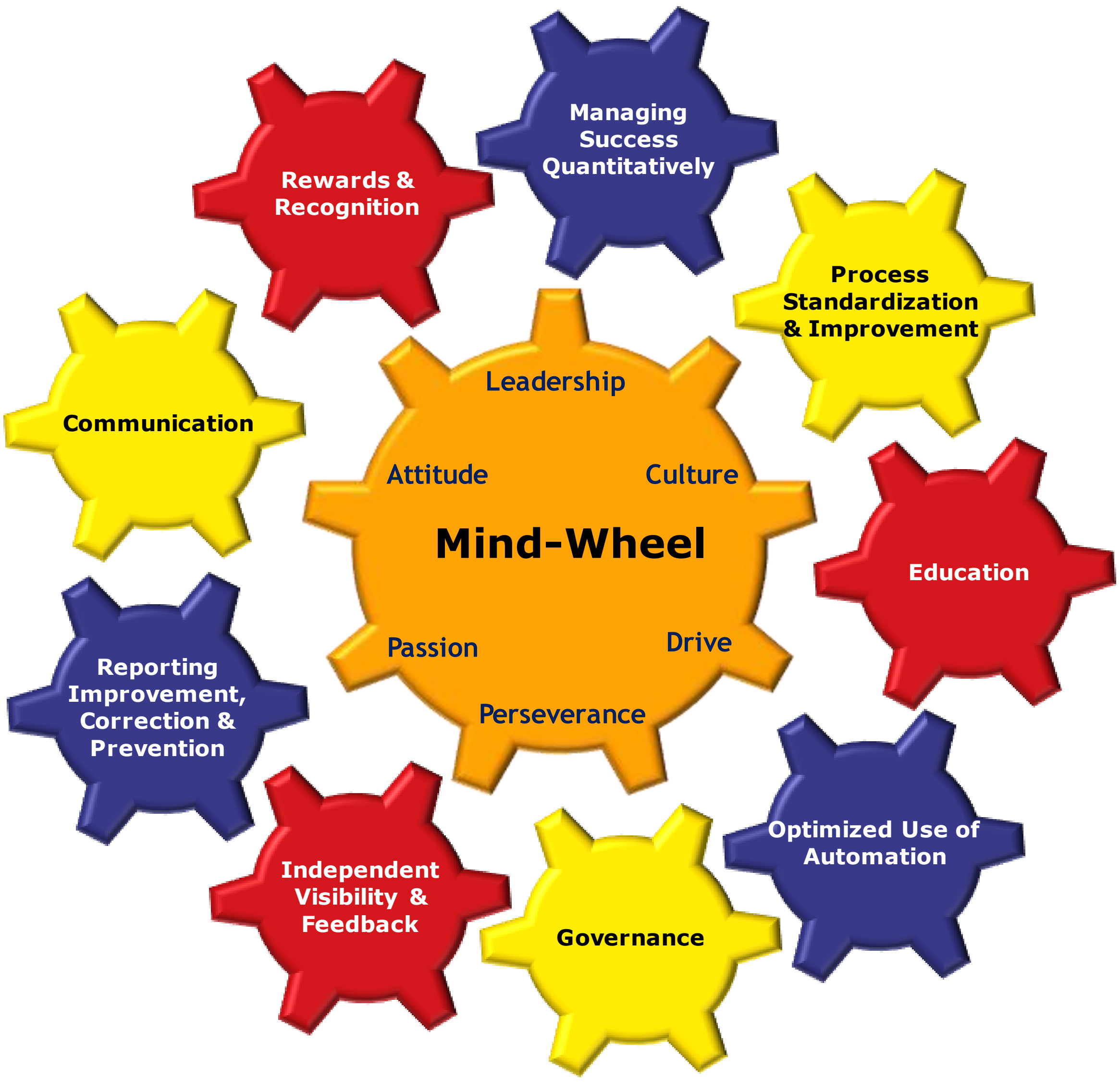

Managing Success Quantitatively
Rewards & Recognition
Process Standardization & Improvement
Communication
Leadership
Attitude
Culture
Mind-Wheel
Passion
Drive
Perseverance
Education
Reporting Improvement, Correction & Prevention
Optimized Use of Automation
Independent Visibility & Feedback
Governance

Fig. 3.1A

The central motor in the change management engine is what we refer to as the Mind-Wheel. This is perhaps the most critical component of the engine and without this wheel; the engine itself would cease to function. Along with the Mind-Wheel, there are nine other gear wheels that contribute to the success or failure of a change management initiative depending on how these are applied. All these peripheral wheels are important for a change initiative to be successful and should be applied in the right direction with required force.

Embracing change is a mindset, like in agile methodology, change if anticipated, expected and embraced; only then it can be successfully implemented.

Sections below explain each one of these wheels starting with the motor.

3.2. Mind-Wheel

"I think therefore I am". The Mind-Wheel is the core of the engine and holds the key to success or failure. There are various key elements that contribute to this wheel. The core team that is formed to lead a change initiative forms the essence of the Mind-Wheel.

While the other wheels manage various functions, the Mind-Wheel or the motor is the soul that keeps everything together. In the event that one of the wheels slows down or stops performing its function, the motor is the one that gets it started again. During the course of implementing the change, the motor helps overcome the hurdles that come in the way of the change being successfully implemented. The motor provides the huge energy that is required to overcome the tendency of the organization or organizational entity that is impacted by the change to remain in a state of inertia. While a change initiative is underway, there are various opportunities for things to go astray and in such an event; the motor provides the situational leadership to overcome the current situation and to move on.

Resistance to change is the balancing force that creates the inertia to change. The core team provides the first level of “external unbalanced force” to move out of that inertia. However, without the larger team being influenced, there is a chance that the resistance is greater than the unbalanced force trying to overcome it. Hence, the core team that forms the motor should be able to influence the larger group that is impacted by the change and therefore is a part of the change initiative.

That being said, resistance to change is a good thing and a necessary part of change management. The absence of this resistance could often mean “submissive compliance” which results in less desirable results and defeats the purpose of change. Doing something “because it should be done” versus “because I want to do it” often determines the amount of success from the change initiative.

Leadership, Culture, Perseverance, Attitude, Passion and Drive forms the oil that makes the efficient functioning of the central motor – The Mind-Wheel

“A leader is a person who people opt to follow to a place where they would not go by themselves” – Joel Barker. In the change initiative, the leader is a person of utmost importance.

3.3. Managing Success Quantitatively

“To be or not to be – that is the question”. The strategic objectives of a change initiative should be clearly articulated and communicated. Special emphasis should be placed on how these connect to the bigger picture at an organization level. The impact on the organization in terms of a positive benefit should be clearly brought out.

The strategic objectives should be driven top down and the very fact that it is part of the senior management goals sends a strong message about the importance of the initiative itself. These should be carried forward through the line function downwards. This mode will ensure that everybody in the line function will breathe these objectives and will make it a success.

The objectives should have clear measurement criteria that are monitored at regular intervals to know the progress. Necessary corrective action needs to be taken if they fall short of the expectations. Measurable incentives can be tied to these objectives, to make them attractive and to provide strategic importance.

The management should carry and "live the message" about the initiative day in and day out. This will ensure that importance is visible and the criticality felt. Apart from formal reports, corridor chats, or even any casual remarks one might make in a meeting, should have questions revolving around these. This will keep the message alive.

The right team must be put in place to take the initiative forward. Right team will consists of the following:

- Capable team, possessing necessary skills
- Team that has right authority and power
- From the line function, where this change is affected

While choosing the right team, it's important to choose members that are passionate about the initiative at hand. This brings positive thrust and will act as additional gear during heavy / tough loads when needed.

The following criteria to be used while framing the goals and objectives:

- Objectives to be realistic
- Objectives to be time bound
- Objectives to be measurable

The key to note here is that the ultimate target and hence any applicable objectives should not change midway through, unless the situation really demands.

A step by step approach is to be used towards goal achievement. Each step in this cycle will have an upper and lower control limit, within which the output of the step is expected. This way, it will be easy to predict whether the next step can be achieved or not, and if not action plan can be formulated to achieve this.

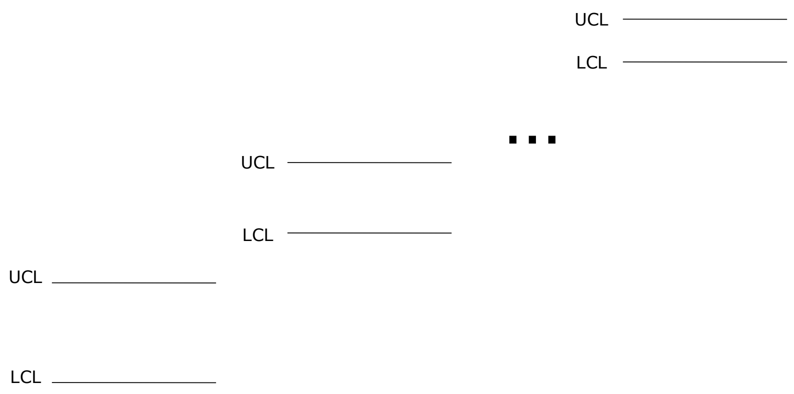


Fig. 3.3A

Moral authority plays a vital role in terms of achieving the objectives during execution. The moral support can add boost if things are falling short and to get that extra bit done.

In addition to line function goal setting, other informal ways like networking through channels can provide added value, as depicted below. The use of boundary spanners across various organizational social networks is critical to the permeation of the message across the organization. This way any hurdles that might come across can be overcome.

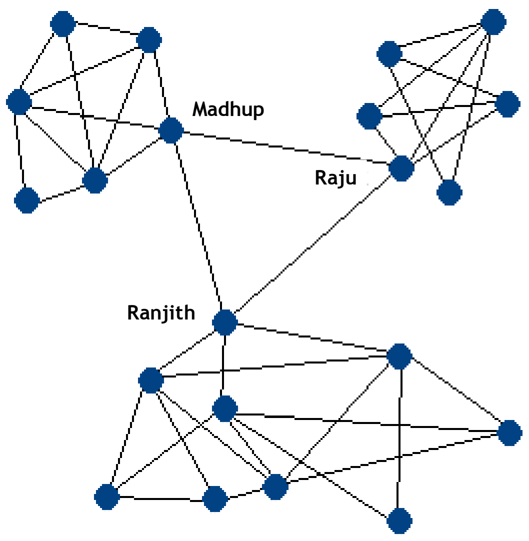


Fig. 3.3B

It's important for the core team (change management team) to know the informal team networks involved and who are the real boundary spanners that connect different teams. While the objectives set will get things done, in parallel, the informal networking should be used to emphasize the message. This will re-enforce the need, and have the required buy-in to get things done.

Another important aspect one should remember is related to the responsibilities. The boundary of responsibilities needs to be very clearly understood. If activities are falling outside these boundaries, then right help to be sought at the right time, rather than getting involved in it and then realizing later that enough help was not provided by the concerned party.

3.4. Process Standardization & Improvement

"Citius, Altius, Fortius – Swifter, Higher, Stronger". The way in which the processes are carried out needs standardization, meaning procedures executed in a uniform manner. By doing this the following benefits are achieved.

- Brings in consistency – uniform working across entities
- Gives predictability – so that one can know what's coming
- Sets basis for further improvement
- Makes the process individual independent rather than person dependent

Once the given process is standardized, then it has to be improved and can be taken up to the next level. Once higher level is achieved, it should be standardized and then this becomes the base for the next improvement. The following picture articulates this perfectly.

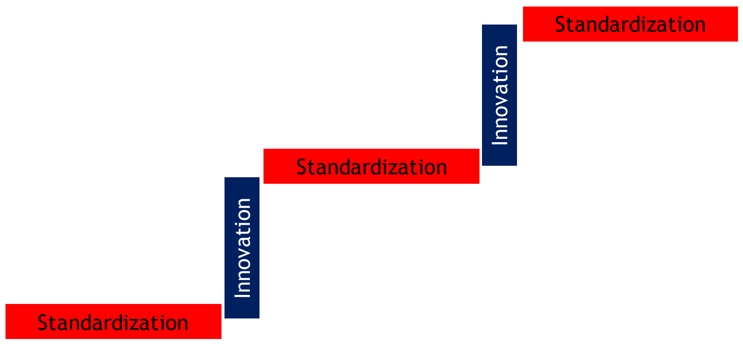


Fig. 3.4A

Constant improvement is needed, because

- we will lose ground, as someone else or the competition will reach high before us
- someone else will catch up at where we are, and hence it's important to go higher level
- it acts as encouragement and inspiration for the team, bringing in challenge
- by nature we want to be best at what we do
- there is also another angle, customer wants to know how we are improving day by day

3.5. Education

"None of us are as smart as all of us" – One cannot over-emphasize the importance of "collective learning". A change management initiative needs to be ably supported through education to help people answer the critical question "What's in it for me?" This is intended to cover both the negative and positive aspects of the change initiative and it is important that the larger team realizes the impact of the change, be it positive or negative. Preparing people for change is as critical as the change itself and unpreparedness or under-preparedness can often lead to the collapse of the change initiative. To educate a person implies setting an expectation in terms of the individual's contribution to the initiative as well as what the individual is to expect from the core team. There are various dimensions of education as depicted in the figure below:

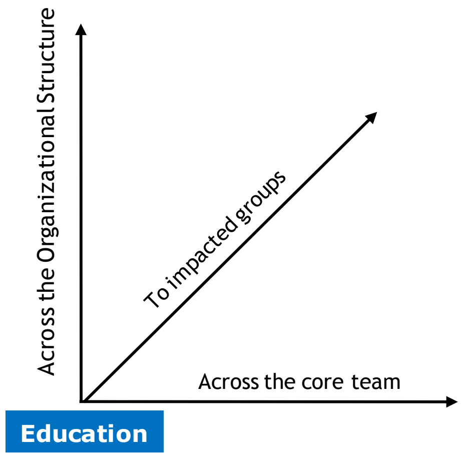


Fig. 3.5A

An important aspect of Education is that before educating others, it is important that one educate oneself. This is an essential step to developing the required conviction and belief in the message that is to be passed on.

The core team must profile the people and provide relevant education based on the dimensions given in Fig 3.5A.

3.6. Optimized use of Automation

"Let the machines analyze – let humans apply judgment and take decisions". The goal of automation is to ensure maximum productivity. To achieve this, it is necessary to reduce the impact of manual errors and also ensure repeatability and reproducibility of the gauge itself. Errors in the gauge should not impact decision making. For improved speed of decision making, one should use automation optimally so that the analysis speed is increased. An important caveat of using automation is that we should not use automation for automation sake – the end result should be the "achievement of the goal" and should not be confused with "use of automation" as the goal itself.

3.7. Governance

"In God we trust, rest bring data". Governance provides the "scaffolding" for a successful change initiative. Just as the scaffolding needs to be in place prior to the actual task being taken up, the Governance mechanism should be set in place ahead of the change initiative being kick-started. The Governing body is analogous to the brain which requires the sensory perceptions from nerve endings to be fed back to it before it can react with an adequate and appropriate response. The sensory perceptions are received from the other wheels in the form of information, issues, communication, success stories, data, etc. through the change actors who are part of the core team. Occasionally, the perceptions can be obtained through feedback from other channels as well. Governance should be a formal process that is "cast in stone" and should be treated that way – in essence, what is defined as the governance process during the change initiative should be resistant to change!

3.8. Independent Visibility & Feedback

"He who told me that I wasn't doing the right thing was my well-wisher". While the governance will ensure that monitoring and control happens through the central vertebrae, there should be host of other mechanisms that need to be used to get the visibility.

Some of the mechanisms that can be deployed are:

- Outside representative assigned for monitoring the workings and reporting independently to the leader
- Informal chats, corridor, water fountain conversations
- Feedback from the team where the change is being done, formal or informal
- Body language of the team during discussions
- Informal networking, through boundary spanning

These inputs should be used to control directly or indirectly the things that need attention. The feedback loop ensures that the process self corrects itself in order to run smoothly.

At times it is noticed that the vital feedback actually comes through the independent channels. This feedback in conjunction with the governance will ensure proper monitoring and bring in required control.

3.9. Reporting, Improvement, Correction & Prevention

"An apple a day keeps the doctor away". Prevention of an issue from occurring helps ensure the fastest possible way to drive through a change initiative. However, prevention is not always possible given the context that the change initiative often involves getting into previously uncharted waters.

Given this, Reporting is a mechanism to help monitor the overall change initiative. This does not imply that the reporting is only done for consumption of senior management but it essentially has to work hand in hand with communication to ensure that the right details are reported to the right audience. For example, the measurement of a parameter against control limits should be reported to the group that has concerned itself with the monitoring of this parameter. Reporting provides a platform for recognition of problems and hence kick-off of corrective action.

Correction involves fixing the issue and also includes setting in place a feedback mechanism so as to prevent the same issue from occurring in the future. This cycle leads to improvement and resulting success for the initiative.

3.10. Communication

"What surprised me was not the surprise itself, but the fact that it was a surprise". This should be the basic tenet with regards to communication in a change management initiative. The quality, quantity, timing and audience of the communication are all to be given equal importance. The importance of communication too cannot be over-emphasized and there should be a belief that there is no such thing as over-communication.

At the onset of a change initiative, communication helps in education and setting objectives. Once the change initiative is in progress, communication helps governance, feedback and reporting. In essence, communication works in tandem with the other wheels to act as a vehicle facilitating the change by supporting the other wheels. The various dimensions of communication are indicated in the figure below:

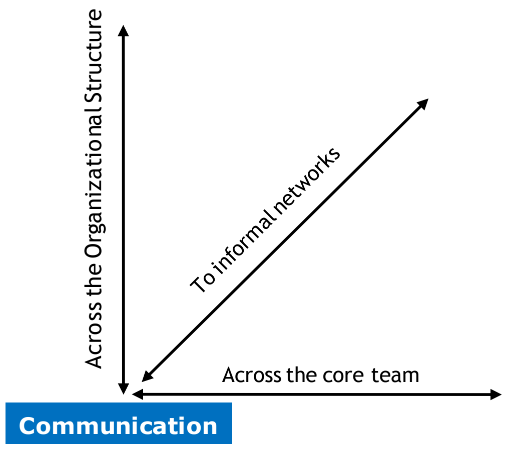


Fig. 3.10A

The core team must profile the people and send the relevant communication to the teams as depicted in Fig 3.10A.

3.11. Rewards & Recognition

"Reward me not for what I am but for what I have achieved and how I have achieved it". Rewards & Recognition and Reprimand are two sides of the same coin. During the change process, it is essential that people within the organization who help facilitate the change and act as "change agents" be rewarded appropriately and in a timely manner. There are various ways in which to use Rewards & Recognition. While these can be used at times to "recognize" the efforts of people, these can be used to send a powerful message in terms of setting an example for the "water cooler crowd". The same is true of using a reprimand. There are innovative ways of applying these concepts and sometimes reward for some can be used as a reprimand as illustrated by the following anecdote.

Six Sigma folklore has it that when GE decided to use Six Sigma as a driver within the organization, Jack Welch requested his leadership team to pool in their best people for the initiative. It didn't take him long to realize that not the "best people" were driving the initiative. To counter this, he proposed an incentive where he recommended that the annual bonus be paid only to the best people and since the best people are in the six sigma initiative, only they would be eligible for the bonus!

4

Consequences

In the example where the change to a new building is announced, if the management used the Metamorphosis pattern the move would have been without any hiccups.

A core team could have been formed and a goal of successful movement could have been given to them. A seat allocation plan could have been created and people would have been allocated seats, cabins, prime locations based on designations or other pre-defined criteria. People could have been educated on staggered move and a process could have been established to be followed. Some rewards for the organizing team could have been planned by the management for a successful move within specified time. A complete report of number of seats, number of network connections, phones, etc could have been published if the pattern was used.

“The satisfaction of a thing well done is to have done it!” A change management initiative is successful if the objective of the initiative sees the light of day.

5

Known Uses

This Metamorphosis pattern can be used for any change initiative, small or large. Depending on the initiative the emphasis of each of the wheels will vary. Following are indicative areas where this can be used:

- **Process Changes**: Changes or introduction of quality processes and standards
- **Behavioral Change**: Building a high performance team
- **Organizational Change**: Implementation of a strategic vision or goal – e.g. Becoming a Billion Dollar Company

Please refer Section 6 for an example

6

A Scenario - Offshoring

6.1. Context

An Organization called MR2 wants to experiment with the offshore model. MR2 is an organization that designs and develops applications in the area of Learning and education. MR2 works with various universities within the country.

6.2. Problem Statement

A new policy of the government which encourages distance learning had enabled universities from other countries to provide learning and training solutions. One day while CEO was having a meeting with the board members, they pointed out on the declining sales figures of the organization. The sales head was almost giving up as none of the normal sales methods were proving to be yielding the results that their competitors were demonstrating. The competition was nowhere near MR2 during the last 2 years and they seem to be doing very well now; in fact they were happy due to the change in the government policy.

Competition analysis was done, they found that the competition was able to build the solutions faster; they were agile with changes in the products. Another startling thing that the competitive analysis showed was that significant portion of the revenue was coming from outside the country. MR2's management took note, they decided to make changes to the way MR2 thinks, manages itself and operates.

MR2 had 50 year old legacy of methods of doing things which were not yielding results in the changed market scenario. They analyzed culture being the root cause of it. Culture which was built over time and they were proud of it once was holding the organization back.

6.3. Forces

The forces identified were:

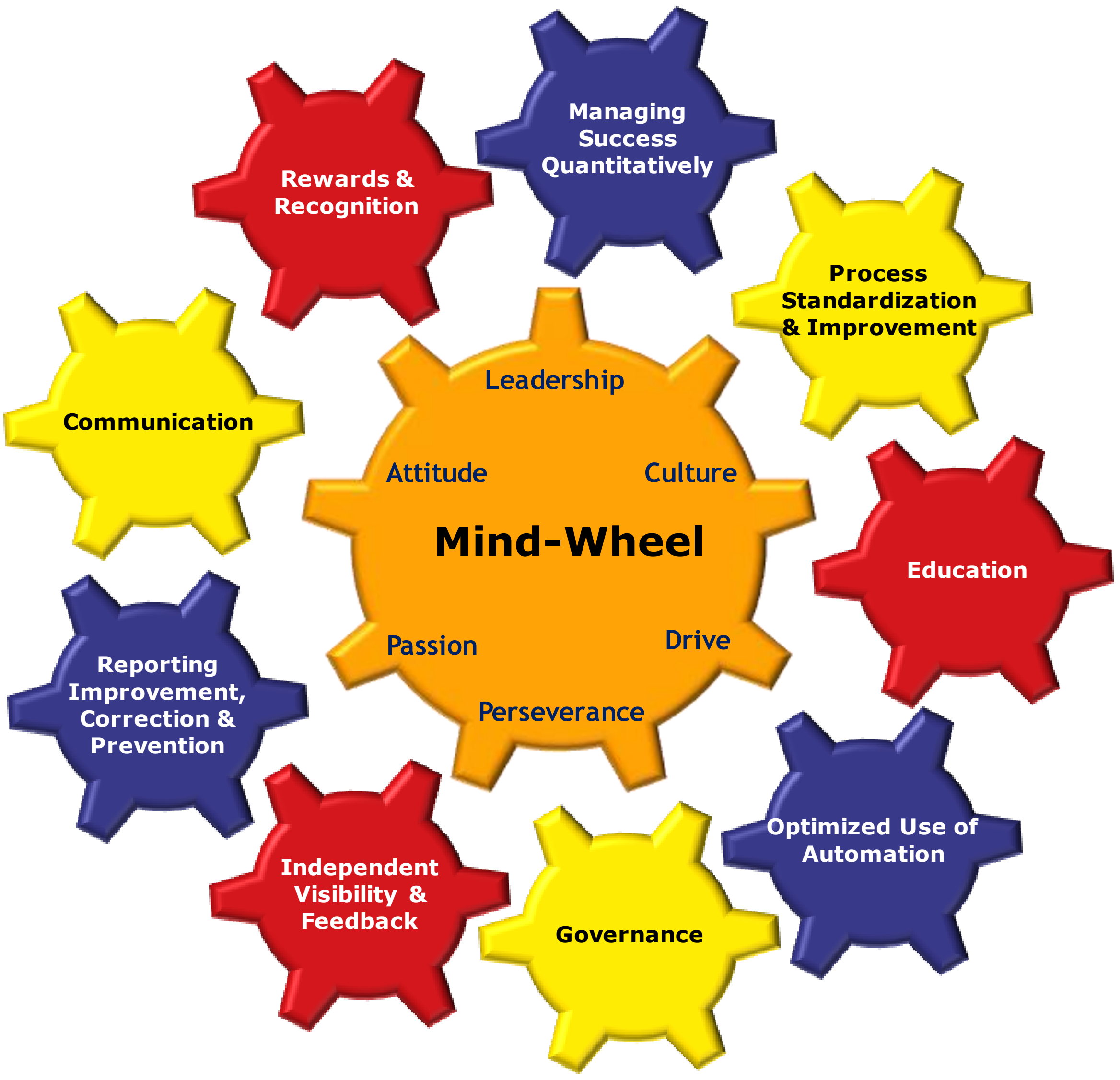
- Business trend
    - Globalization of market; competition started using offshore as a value partner
- Changed Government policy

    - Government policy of opening of the domestic market to international players thus encouraging distance learning program.
    - Government was providing tax holiday for organizations dealing in Foreign exchange
- Declining sales
    - Sales were declining as competition was providing solutions at competitive price
    - Competition was charging premium pricing for value added services
- Reputation at stake
    - Customers were getting attracted to competition
    - It was difficult to attract new and fresh talent
- Impacted morale of workforce
    - It was no longer ‘cool’ to say I work for MR2
    - Declining sales and profitability was adding to the misery
    - Work force perceiving greener pastures outside the organization

6.4. Solution

Management decided to take a few steps; they needed to embark on the offshore model. But it was not as simple as it seemed. They were faced with resistance to bring in this change due to change in work timings, working with folks who speak different language, need of traveling, collaboration issues, etc.

They managed the solution by using the gears as depicted in the pattern in the following manner**:**


Managing Success Quantitatively
Rewards & Recognition
Process Standardization & Improvement
Communication
Leadership
Attitude
Culture
Mind-Wheel
Education
Passion
Drive
Perseverance
Reporting Improvement, Correction & Prevention
Optimized Use of Automation
Independent Visibility & Feedback
Governance

6.4.1. Example of approach followed:

The central gear serves as the engine to get the process rolling. Although there is no one sequence that is mandated through this pattern, an organization needs to discover its own sequence, size of the gears, and intensity with which they need to be applied. Every problem is different and so will be the application of the pattern. The pattern suggests that all gears need to be necessarily applied for successful change, only the size and intensity might vary.

The approach that was followed by MR2 is depicted in the table below; they used the various gears and outlined actions to bring about the needed change.

Name of Gear	Weight/Size Low - High(1, 2, 3, 4, 5)	Solution/Action
MindWheel	5	* A core team was identified along with a Mentor who had worked in multiple countries * The Mentor explained the importance of change management within the organization * A vision was articulated to bring in the change. * A three month plan was prepared * Key people were identified to work on the plan (now these people were the best people they had in the Organization not the ones who were available) * The criteria to select the task force was people with - High energy, positive mindset, perseverance, respected for their area of work, who can think differently * Three pilots projects were chosen for offshoring * Part two of the vision was to sustain the changed way of working without slipping back
Education	3	* A week long workshop was planned with the identified task force led by the mentor * Senior management was involved * When one of the Senior Directors talked about not attending the complete program the mentor explained the importance of driving from the front. CEO requested the Director to spend full time during the workshop *(part of change management)* * Mentor trained people on cultural aspects * Team Communication skills were refined * Culture appreciation trainings were conducted * Core Team was educated about offshore model and its pitfalls * Identified skeptics were educated on the concept of offshoring
Communication	4	* Advantages of offshore were communicated to everyone through a newsletter * Advantages like working with offshore, time difference, cost arbitrage, higher productivity were emphasized * Out of the three pilots, two pilots were managed well, the third pilot ran into issues. Learning from both the experiences were documented and communicated * First they started with monthly meeting and then changed to Newsletter every month and meeting every two months. The management went around doing road show within the organization about benefits of offhosring and showcasing people who were being the role models at that time * The work force was asked to speak freely about what could be hindering the progress, people were being rewarded and news communicated * Case studies were published on how offshoring is increasing the market share demonstrating the value add with numbers

Name of Gear	Weight/Size Low - High(1, 2, 3, 4, 5)	Solution/Action
Independent visibility and Feedback	3	* Management appointed a small group to review and report the progress * The management wanted to make sure that all actions that were being suggested by the core team were getting implemented completely * There was some early visibility given by this group on the third pilot not going well. This helped in mitigating the actions. Though the third pilot was delayed, it finally got accomplished due to the early visibility provided
Reporting, Improvement, Correction and Prevention	3	* Weekly reports were published as per the quantitative goals set for the offshore partner, some of these included the following: - Progress on the scheduled plan, percentage completion - Percentage of bandwidth released for MR2 managers - indication on cost savings - Service level agreements tracking * The reporting format/data was fine tuned with the pilot phase concluding. * Corrective actions were put in place, especially related to the third pilot learning, where things didn't go that well. * By the end of the third pilot, the management and the core team had a handle on the steady state method of operation
Governance	4	* Weekly meetings were conducted to review and monitor progress of the three pilots, involving Project Managers from MR2 and offshore partner * Monthly steering reviews were held with MR2 management and offshore partner management * All numbers (wins and losses) were reviewed every fortnight * Actions were taken against underperformance and non-performance
Process Standardization and improvement	4	* The working process at MR2 and at the offshore partner was analyzed. A common understanding was arrived at by looking at the best of the two process models, with minimal investment required by MR2. * At the end of the pilot phase improvements were listed and the processes were fine tuned, taking the learning into account.
Managing success quantitatively	4	* Two small teams were sent to two identified locations to study feasibility of offshoring * Due diligence was done and offshore partner was chosen * Three pilots were chosen for the three month pilot phase * The partner was given quantitative goals to demonstrate their competence, which were monitored through the pilot phase.
Rewards and recognition	5	* The 2 successful pilots were rewarded. * During the steady state phase, Bonus was declared for the first 5 units to move work offshore and also for the first 5 business wins outside the country * There was a press release issued quoting the names of the managers that successfully piloted the offshore initiative.

Name of Gear	Weight/Size Low - High(1, 2, 3, 4, 5)	Solution/Action
Optimized use of automation	1	* Collaborative tools were put in place to encourage distributed development, leveraging different time zones. * Various Dashboards were put in place for reporting metrics, so that all data is available at one place on a real time basis; which is core to taking management decisions

7

References

7.1. QUOTES

The only thing constant in life is change itself - François de la Rochefoucauld

I think therefore I am - René Descartes

To be or not to be – that is the question – William Shakespeare

Citius, Altius, Fortius – Swifter, Higher, Stronger – Olympics Motto

None of us are as smart as all of us – Japanese Proverb

Let the machines analyze – let humans apply judgment and take decisions: One of Us

In God we trust, rest bring data - W. Edwards Deming

He who told me that I wasn't doing the right thing was my well-wisher – One of Us

An apple a day keeps the doctor away – Idiom

What surprised me was not the surprise itself, but the fact that it was a surprise – One of Us

Reward me not for what I am but for what I have achieved and how I have achieved it – One of Us

The satisfaction of a thing well done is to have done it! – One of Us

Some goals are better not met – One of Us

I know not fear when I dive into the deepest realms of my imagination – One of Us

8

Acknowledgements

We acknowledge all individuals who have triggered our thoughts and have been examples of resistance to change. They have essentially quite literally set the wheels in motion.

MindTree Limited

Contact Information

Email: madhup_jain@mindtree.com

Raju_dani@mindtree.com

Ranjith_kutty@mindtree.com

Phase 1, West Campus,

Global Village, RVCE Post

Behind RV College of Engineering,

Mysore Road, Bangalore

INDIA – 560059

Analyzing the HCI Design Pattern Variety

Christian Kruschitz
Department of Informatics-Systems
University of Klagenfurt
Klagenfurt, Austria
chris@isys.uni-klu.ac.at

Martin Hitz
Department of Informatics-Systems
University of Klagenfurt
Klagenfurt, Austria
hitz@isys.uni-klu.ac.at

ABSTRACT

Human-Computer Interaction (HCI) design patterns are an often used tool for developing user interfaces. They render the communication among stakeholders more efficient and allow for a faster design of user interfaces. However, today there exists a vast amount of patterns written by many different authors, published on Web repositories, in scientific papers, and books. This causes the form or structure of the patterns to vary according to the authors' preferences. This paper presents the results of a survey that analyses the structure and relationships of HCI design patterns from 21 different design pattern resources.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Theory and methods; H.5.2 [**User Interfaces**]: Style Guides

General Terms

Design Pattern, Formalization, Human-Computer Interaction, Pattern Structure, Usability, Survey, Standardization

1. INTRODUCTION

We investigated a selection of 21 HCI pattern languages and collections that were published through Web repositories, scientific papers and books between 1996 and 2007. Our main goal of this survey was to analyze the similarities in the pattern form in order to specify a unified HCI design pattern format that should be used as the basis of XPLML (eXtended Pattern Language Markup Language). An unification of the pattern structure should be achieved to exploit the full reuse potential of patterns, which does not constrain pattern authors in their work but supports pattern users by easing the process of understanding and instantiating patterns to specific design problems[12].

Four books, eight scientific papers and nine online resources were examined. Three out of four books deal with pattern languages [5, 24, 10] and one is a pattern collection [21]. Two papers describe a pattern language [9, 15, 27] and five papers portray pattern collections [6, 7, 18, 20, 25]. Finally, nine pattern collections were found online [11, 13, 16, 17, 23, 22, 28, 29, 30], the most important resource for design pattern users, because of the easy access and high availability.

The survey is divided into seven subcategories which are described in more detail. These categories are:

- Publication Year/Publication Medium
- Arrangement of Design Patterns
- Device Categories for HCI Design Patterns
- HCI Design Pattern Domains
- HCI Design Pattern Categorization
- HCI Design Pattern Structure
- Pattern Relationships

From the vast amount of design patterns available, we have chosen the above-mentioned repositories and pattern sources because we believe that they reflect the current state of efforts in the HCI design pattern community well.

2. DEFINITION

We are talking from pattern languages, design patterns and pattern collections. To make it clear what we mean with this terms we provide a short definition of each of them.

2.1 HCI Design Pattern

An HCI design pattern describes a recurring user interface design problem together with a proven solution. An HCI design pattern, in the following referred to as "pattern" or "design pattern", has a well defined form, which is dependent on the individual author's preferences. A pattern form should be used consistently across a pattern language or pattern collection. This makes it easier for pattern users to understand the problem, context, and solution of a pattern throughout a pattern collection/language. The pattern itself, when it is a part of a collection or a pattern language, may have references to other patterns.

2.2 Pattern Catalogue/Collection

Patterns stored in so-called DESIGN PATTERN CATALOGUES or COLLECTIONS are categorized to support faster navigation within the repository. However, they show almost no relationships among each other and thus do not form a fully interconnected system. The catalogue/collection contains several patterns that stand alone and have no connections to predecessor or successor patterns. Furthermore, such a collection usually does not completely cover a specific application domain.

2.3 Pattern Language

In contrast to a pattern catalogue / collection, a "pattern language" is a complete collection of patterns for a given family of design problems in a given domain. A pattern language describes problems by means of high-level design patterns, which are solved by low-level design patterns. The design patterns are connected through relationships, so that they constitute a network.

In a pattern language, the "words" are the patterns, while the connections between patterns represent the "rules of grammar" which are situated in the pattern itself. When words and rules of grammar are combined, a "sentence" is generated. Sentences can be built in many different forms when the rules are followed. So there is not only one path through a pattern language, it offers several possibilities to solve a design problem. A good example is "The Design of Sites" by van Duyne et al., a pattern language that allows designers to articulate an infinite variety of Web designs [24].

3. SURVEY RESULTS

3.1 Publication Year / Publication Medium

The cornerstone of design patterns as a tool of knowledge was laid back in the late 1970s when the mathematician and architect Christopher Alexander published several books [1, 3, 2] in which he proposed the concept of design patterns and pattern languages. Ward Cunningham and Kent Beck have adopted this principle to object-oriented programming (OOP) and user interface (UI) implementation in 1987 [4, 19]. They presented five patterns for designing window-based user interfaces in Smalltalk.

In Human-Computer Interaction, the start of the design pattern era was when Coram et al. [7] published the first design patterns of a pattern language for user-centered interface design (see Fig. 1). The objective of this design pattern language was to provide high-level patterns with which user interface designers could build graphical user interfaces which are pleasurable and productive to use.

In the following years several other pattern collections and pattern languages were published. The publishing activity in scientific papers recently slowed down because of the publication of four seminal books in the HCI design pattern community. The first in 2001 [5], two books in 2003 [10, 24] and one in 2005 [21]. At the same time, pattern writers have focused on developing repositories on online platforms. Due to the hypermedia characteristics and the 24/7 availability of data on the Internet, it is much easier to reference to other patterns and disseminate patterns across the HCI community. A further benefit of online resources

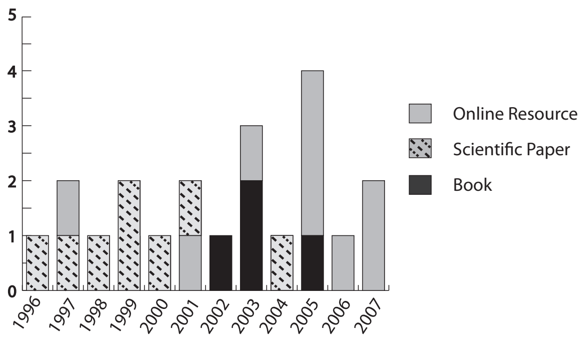


Figure 1: Year and Type of HCI Design Pattern Publications

is that users can contribute in writing and improving HCI design patterns using Web 2.0 features, for example, *van Welie's Online Repository* [29] and the *Yahoo!Design Pattern Library* [30] allow users to post comments on a design pattern, while it is not allowed to make changes directly to specific design patterns.

3.2 Arrangement of Design Patterns

Design patterns can be arranged in pattern collections or languages (cf. Section 2). The latter connect design patterns to an interconnected network whereas collections do not.

Five pattern languages were published in books consisting of 10 to 90 interconnected design patterns, and three were published in scientific papers. Due to the spatial limitations of scientific papers, these pattern languages are composed of 8 to 9 design patterns.

Beside pattern languages, 15 design pattern collections were found. One was published in a book, five were published in scientific articles and 9 were published through online repositories. The book consists of 94 design patterns, the scientific papers of 5 to 45 patterns and the online resources of 18 to 130 patterns.

A reason for the predominance of books as a publication media for complete pattern languages – besides the volume aspect – may be the scientific incentive for the intensive and time consuming research necessary to discover and describe a pattern language. A disadvantage of casting patterns in books, however, is the updating process. It is difficult to add to or improve already published pattern languages, whereas the nature of design patterns mandates to update them on a regular basis since interaction techniques change over time due to the invention of new hardware and novel interaction methods. Therefore, several books come with companion websites where pattern language updates are published regularly (e.g., [21], [24]).

Many design pattern collections can be found online, because of the easy updating and dissemination of patterns to a large audience. Some of these resources are, however, not maintained very well - last updates have occurred years ago - perhaps a typical fate of short term academic projects. Beside these not maintained online collections there are a

Domain	Book	Scientific Paper	Online Resource
Web User Interface Design	2	2	7
User Interface Design (Desktop Applications)	2	5	1
Interactive Exhibits	-	1	-
Software Design (UI Related Programming)	-	1	-
Hypermedia Applications	-	1	-
Ubiquitous Computing	-	1	-

Table 1: Domain and Publication Medium of HCI Design Pattern Languages / Collections

few which are updated regularly. These are the *Little Spring Design – Mobile UI Design Resources* [13], *UI Patterns – User Interface Design Pattern Library* [23], *Welie.com – Patterns in Interaction Design* [29] and *Yahoo!Design Pattern Library* [30]

3.3 Device Categories for HCI Design Patterns

Several design patterns are applicable for specific hardware devices. Our research showed that most design patterns have been discovered for desktop and handheld/smartphone applications. Handheld/smartphone applications applications differ from desktop applications in the fact that the display and input devices available are smaller and cannot be used as easy as those from desktop applications. Therefore, different interaction methods are used.

In most cases it was not explicitly mentioned which design pattern applies to which platform. Normally, design patterns should be written in a platform independent manner, but hardware limitations as mentioned before force specific interaction methods. When no specific platform is mentioned to which a design pattern applies, the forces, the context, the problem, and the solution element of the design pattern constrain the platform where the pattern can be implemented. To make an HCI design pattern more effective, the author should mention the hardware platform in the head of the pattern.

3.4 HCI Design Pattern Domains

Each domain has its specific forces. Thus, forces should be resolved in a domain specific way. Therefore, HCI design patterns are written for a specific domain they can be applied to. Our survey showed that the predominant domains are web design and interface design for desktop applications. Web design patterns were almost exclusively published online, while interface design patterns for desktop applications mostly appear in scientific papers (see Table 1).

3.5 HCI Design Pattern Categorization

Patterns are grouped according to the basic idea they address. As shown in Table 1, prominent domains are Web user interface design and interface design for desktop applications. Table 2 and Table 3 are showing which informal categorizations design pattern authors have used to subdivide the design patterns in the respective domain. These categories are the most popular but should not be considered as a formal standard categorization in the field of HCI design patterns. Research approaches in categorizing HCI design patterns are mentioned in the literature [8, 14, 26].

These categorization schemes differ in a few elements. The basis of the Web user interface design categorization (Table 3) is the interface design for desktop application categorization (Table 2) without the elements *Visibility*, *Data Representation* and *Natural Mapping* but enriched by *Site Genres*, *E-Commerce*, *Optimization* and *Accessibility.* Due to the nature of the Web design domain, these supplements are necessary to complete the solution space.

3.6 HCI Design Pattern Structure

The content elements of a design pattern can vary due to the preferences of a pattern author. Several authors are using the Alexandrian form to describe discovered patterns in a structured way. However, the Alexandrian form is not taken as the preferred structure in all design patterns and therefore many different pattern forms and content elements exists in the HCI design pattern domain.

We have analyzed the design patterns according to their structure and their content elements. Table 4 shows all discovered content elements and which author is using which content elements. The pattern structure was divided into 4 parts, namely the HEAD, the BODY, ADDITIONAL INFORMATION and REFERENCES. To understand each of the content elements described in Table 4 we have brievely described each of them below.

The minimal set of common content elements throughout all analyzed design patterns can be used as a basis of a "standard" HCI design pattern form. These are:

- HEAD
  - Pattern Name
  - Sensitizing Image
  - Short Problem / Content Description (Summary)
- BODY
  - Context
  - Forces
  - Solution
  - Example / Pattern Instance Gallery
- REFERENCES
  - Related Patterns

Category	Description
Visibility	How to design something so that the user knows immediately how to use it just by looking at it
Feedback	Describes how feedback should be generated if a task is being correctly or incompletely completed.
Natural Mapping	Creates a clear relationship between what the user wants to do and the mechanism for doing it.
Content Organization	Information architecture and application structure.
Navigation	How to get around efficiently in the application.
Layout	Shows how to layout application screens for a satisfying result.
Data Representation	Techniques how to represent large data sets.
Getting Input from User	Provides appropriate input methods.
Search	Describes how search methods can be incorporated into the application.
Accessibility	Techniques and methods to adopt the website for people with disabilities.

Table 2: Categorization of Design Patterns in the Interface Design for the Desktop Application Domain

Category	Description
Site Genres	Describes various site genres, e.g., News Site, Personal Homepage, Shopping Site, Information Site, etc.
E-Commerce	Shows methods which can be used to enrich websites with e-commerce functions.
Optimization	Technical advices to speed up the website.

Table 3: Additional Categories of Design Patterns in the Web User Interface Domain

These elements can be considered as a mandatory set of content elements of a well-defined HCI design pattern. There is enough information to understand the problem, context, and solution of the addressed design pattern. Beside these basic elements, authors should have the possibility to add their own elements to enrich the patternŠs content with useful information for easier implementation of the pattern. The following resources are using the minimal set of content elements together with others:

- Ian Graham, “A Pattern Language of Web Usability”, [10],
- Douglas van Duyne, “The Design of Sites”, [24],
- Carol Stimmel, “Hold Me, Thrill Me, Kiss Me, Kill Me”, [20], and
- “User Interface Design Patterns”, [22].

Below we give a description of each of the content elements mentioned in Table 4.

3.6.1 HEAD

The HEAD or introduction paragraph of the pattern gives the pattern user a short overview of the problem which the pattern addresses. A short context description and an image which shows a successful solution to the addressed problem are placed at the top of the patter structure. The image (or sensitizing image) ensures that the pattern is remembered more easily. The description of the problem and context must be as short as possible while it must give as much information as possible to get a rough idea of the patterns problem space. Along with this information some metadata is also placed in the head of the pattern. Below there is a short description of each of the content elements which were found in the head of the analyzed HCI design patterns.

Pattern Number
The number uniquely identifies a pattern within a pattern language/collection. It is useful for referencing. An alphanumerical code is used when it should encode the categorization of the pattern as well.

Pattern Name
The pattern name is the reader’s first “contact” to the pattern proper and - beside the sensitizing image - the most important cue to remember the pattern. Therefore, it should be chosen wisely to give a significant hint to the content of the pattern. A unique name should be used which is easy to remember and to use for unambiguous communication within design documents, meetings and other situations.

Alternative Pattern Name
Also known as “AKA”. Indicates the alternative names of a pattern.

Rating/Ranking
This item indicates how the author or the pattern users are rating the pattern. It should help pattern users to decide if they can use it without worries or should rather consider another pattern.

Sensitizing Image
"A picture is worth more than a thousand words". Beside the name, the sensitizing image creates the user's first impression of the design pattern. A good screenshot or - even better - sketch can help to grasp the idea behind the pattern's solution much faster. If the solution has a dynamic character, a short animation may be appropriate to demonstrate it to the user.

Short Problem/Content/Context Description
This content element should give a short overview of the problem, content and/or the context of the pattern. For a more detailed description of the *problem* and/or *context*, the context and the detailed problem description content element in the BODY part of the pattern should be used.

Author Name
The author name designates the contact person and writer of the pattern.

Pattern Classification/Group
Pattern in languages/collections are grouped according to a common underlying idea. (see Pattern Number)

Creation Date
Shows when the pattern was first created.

Last Revision Date
Together with Creation Date the Last Revision Date content element is very useful to show when the pattern was revised. Ideally all revision dates are published so the user knows how often and when a pattern was updated since patterns tend to change over time due to the invention of new hardware and interaction methods.

Hardware
Shows on which device the solution of a design pattern can be implemented.

Level
Indicates if it is a high-, medium- or low-level pattern. High-level patterns describe problems and solutions in a very abstract way. These patterns lead to more detailed patterns such as medium- and low-level patterns. Low-level patterns are the most detailed, describing e.g. the function of a certain interaction widget such as an action button.

3.6.2 BODY

After a user has decided to use a pattern, the BODY section offers the user detailed information about the problem, forces, context, solution, as well as many more additional information for a better understanding of the design idea addressed. The information provided in the BODY section extends the information of the HEAD section and is more detailed. The BODY of a pattern consists of the following (unordered) content elements.

Context
Beside the problem and the solution content element, the context is essential for the understanding of a design pattern. This element makes a pattern distinct from a style guide or guideline document. It shows designers the preconditions in which situation the problem and its solution occur, and thus defines the applicability of a particular pattern.

Detailed Problem Description
This element describes the problem the pattern solves. A detailed analysis of the problem and background information is provided to clearly understand the design problem.

Forces
This element discusses the forces and constraints relevant to the pattern and how they conflict and/or interact with each other. Forces help the user to better understand the problem and the connection to the context.

Solution
The Solution addresses the reuse of recurring design practices and how to resolve the forces discussed in the Forces element. It is written in a way that the designer has an idea how to resolve those forces in an efficient way. But it is not like a guideline where you only need to follow the instructions step-by-step to get a solution. Therefore the designer's creativity is necessary to produce a good solution.

Rationale
The Rationale element describes why the pattern works. It describes how and why the current pattern resolves its forces. It goes deep into the mechanisms which are used to get the forces into harmony. In other words, it is a proof of concept.

Diagram
Sometimes diagrams or sketches are used to summarize the solution of a pattern. It is not a working instance of a pattern, but it rather gives the user another view of the solution and it supports the design decision of a UI engineer. When the solution has a time dimension, a storyboard may be a better tool to demonstrate the solution.

Resulting Context
After resolving the forces of the current pattern, it builds a new context for other patterns. This content element discusses the resulting context and which patterns may be applied next.

Examples
Links and screenshots or working instances of the design pattern's solution are presented in this content element. It is good practice to show many different solutions, to give the UI designer a better understanding of how to implement the pattern's solution.

Known Uses
It gives information where to find good, already implemented solutions on different platforms.

Counter Examples
Shows bad design in the context of the current design pattern's problem/solution approach. Usually a link and short description or screenshot to the faulty design is provided.

3.6.3 Additional Information

These elements do not fit in the two sections above, but enrich the pattern with more information. The elements shown below are used by a few authors. Actually, there is no order of the content elements.

RELATED IDEAS / LITERATURE
This element comprises references to basic literature and/or ideas regarding the interaction mechanisms the design pattern describes.

REFERENCES TO IMPLEMENTATIONS
Authors provide direct links to good solutions. Because of the nature of a link, it is used in online resources. It supports the element EXAMPLES in the BODY part of the design pattern.

CODE EXAMPLES
Whereas code examples are very often used in software engineering patterns, in HCI they are not so popular because of the many different possibilities to implement an interaction mechanism. Sometimes there are code examples provided for better understanding of the pattern's solution. This content element is provided only in online resources.

ACCESSIBILITY
It shows how to extend the pattern solution in such a way that the application can be accessed by people with disabilities.

3.7 Pattern Relationships

References are essential when working with pattern languages. They must be incorporated into the design pattern structure. Authors must take care to link the patterns in the right way and order so that they can build a network. In pattern collections references to other patterns are not so essential because there are many pattern which are not connected to another pattern. References are also used to indicate which other patterns can be applied after having implemented a certain pattern.

During our survey it was interesting to find out that many authors as inter-pattern relationships used association and aggregation. Specialization was used by two authors.

A new relationship is also pointed out. It is named the "anti-association" connection. This connection is similar to the association but it references to an anti-pattern. An anti-pattern describes, in contrast to a design pattern, a problem statement with a bad solution. But only one author used an anti-pattern to show the user how not to solve a problem [20]. This is an interesting concept since it can show user interface designers common pitfalls to learn from.

REFERENCES IN TEXT
This relationship indicates if references to other design patterns are made within the PROBLEM, FORCES, SOLUTION, etc. content elements. The benefit of referencing design patterns in such a way is that when a problem occurs that the design pattern does not solve, the user can be guided to more appropriate patterns without having to search the design pattern for the RELATED PATTERNS content element.

RELATED PATTERNS
Unlike REFERENCES IN TEXT, no references were made in the design pattern itself. The RELATED PATTERNS content element encapsulates all references to other patterns. Usually, it includes references to lower-level patterns.

SPECIALIZATION
Specialization of a design pattern means to add more attributes to it. The `SEARCH` pattern, for example, only provides a basic search mechanism. To extend the concept of this pattern a more specialized pattern is generated. It inherits the attributes from `SEARCH` and adds new ones to fulfill the purpose of advanced searching. This leads to the specialized pattern `ADVANCED SEARCH` (see Fig. 2).

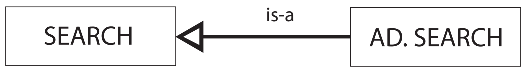


Figure 2: Design Pattern Specialization

AGGREGATION
When a pattern consists of more than one sub-pattern, an aggregation relationship is used to connect them. As an example, consider the `SHOPPING CART` pattern in Fig. 3. This pattern consists of many sub-patterns, like `LIST BUILDER`, `WIZARD`, etc. After applying these patterns in a way suggested by `SHOPPING CARD`, the problem and forces of `SHOPPING CART` are solved.

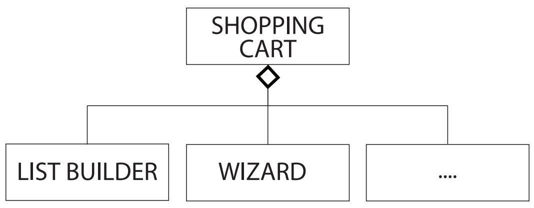


Figure 3: Design Pattern Aggregation

ASSOCIATION
An association (see Fig. 4) between design patterns is a unspecific connection between them. When referencing patterns, the words "related to" and "similar to" are often used to indicate an association with another design pattern.

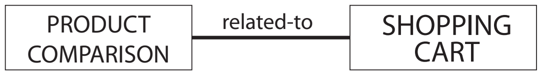


Figure 4: Design Pattern Association

ANTI-ASSOCIATION
Anti-Association is similar to Association. It is a connection to an anti-pattern. It shows how a pattern should not be implemented. This is a good way to demonstrate common pitfalls and bad design solutions.

4. CONCLUSION

This survey shows in which way HCI design pattern authors are writing design patterns. Most HCI design patterns are published through websites or Web portals. Basically a good idea, but it is also necessary to maintain the site to provide up-to-date information, since over time, a design pattern may change because of the invention of new interaction mechanisms. Therefore, patterns published online should be updated regularly. Only a few portals are maintaining their pattern collections, one reason being possibly the lack of tool support.

Structure and organization of patterns vary due to their authors' preferences. Thus, there is no consensus on how patterns should be formulated and categorized in order to provide appropriate information to produce good interface design. Many authors are using the Alexandrian form, possibly because it was the first form used to encapsulate design knowledge. So pattern authors have transferred the Alexandrian pattern structure to software engineering and then to the HCI domain. It seems necessary, however, to identify out the most important elements for HCI design patterns to better support the work of HCI designers and pattern authors, respectively. This survey has analyzed the most frequently used content elements. But to propose a unified pattern form for the HCI domain, workshops and discussions in the HCI community are necessary to develop an generally acceptable basic HCI design pattern structure.

Finally, during our research on freely accessible HCI design patterns it was interesting to find out that most patterns are written for Web design issues. In the last years Web design was one of the hottest issues in UI design, where UI designers have adapted many principles of ordinary desktop interface design to the special needs of websites.

5. FUTURE WORK

With the results of this survey and a study of categorization schemes in usability literature we will be able to propose a classification scheme or a taxonomy for HCI design patterns. It should help pattern authors to identify overlapping design patterns and define patterns according to their problem group or scope. For pattern users a taxonomy is useful when searching for solutions to a specific design problem and alternative solutions can be found more easily.

6. ACKNOWLEDGMENTS

We would like to thank Prof. Dinesha from the Indian Institute of Technology Bombay, for his suggestions and discussions on this topic which helped us to improve the quality of this paper.

7. REFERENCES

[1] C. Alexander. *The Oregon Experiments.* Oxford University Press, 1975.

[2] C. Alexander. *The Timeless Way of Building.* Oxford University Press, 1979.

[3] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language*, volume 2. Oxford University Press, New York, 1977.

[4] K. Beck and W. Cunningham. Using Pattern Languages for Object-Oriented Programs. In *OOPSLA 87 workshop on the Specification and Design for Object-Oriented Programming*, 1987.

[5] J. Borchers. *A Pattern Aproach to Interactive Design.* Software Design Patterns. Wiley, 2001.

[6] E. S. Chung, J. I. Hong, J. Lin, M. K. Prabaker, J. A. Landay, and A. L. Liu. Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing. In *DIS '04: Proceedings of the 5th conference on Designing interactive systems*, pages 233–242, New York, NY, USA, 2004. ACM.

[7] T. Coram and J. Lee. Experiences - A Pattern Language for User Interface Design. 1996. Available at: `http://www.maplefish.com/todd/papers/Experiences.html`.

[8] S. Fincher and P. Windsor. Why patterns are not enough:some suggestions concerning an organising principle for patterns of UI design. In *CHI '2000 Workshop on Pattern Languages for Interaction Design: Building Momentum*, 2000. `http://www.cs.kent.ac.uk/people/staff/saf/patterns/chi00.pdf`.

[9] A. Garrido, G. Rossi, and D. Schwabe. Pattern Systems for Hypermedia. In *Pattern Languages of Programming 1997*, 1997.

[10] I. Graham. *A Pattern Language of Web Usability.* Addison-Wesley, 2003.

[11] Hypermedia Design Patterns Repository. Online. Available at: `http://www.designpattern.lu.unisi.ch/index.htm`, Accessed on December 27, 2009.

[12] C. Kruschitz. XPLML: a HCI pattern formalizing and unifying approach. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4117–4122, New York, NY, USA, 2009. ACM.

[13] Little Spring Design - Mobile UI Design Resources. Online. Available at: `http://patterns.littlespringsdesign.com/index.php/Main_Page` , Accessed on December 27, 2009.

[14] M. J. Mahemoff and L. J. Johnston. Pattern Languages of Usability: An Investigation of Alternative Approaches. In J. Tanaka, editor, *APCHI 98 Proceedings*, pages 25–31. IEEE Computer Society, Los Alamitos, CA, 1998.

[15] M. J. Mahemoff and L. J. Johnston. The Planet Pattern Language for Software Internationalisation. In *Pattern Languages of Programs 1999 Proceedings*, Monticello, IL, 1999.

[16] PatternCube - Design Pattern Portal. Online. Available at:`www.patterncube.com`, Accessed on May, 2008.

[17] Patterns for Personal Web Sites. Online. Available at:`http://www.rdrop.com/~half/Creations/Writings/Web.patterns/index.html`, Accessed on December 27, 2009.

[18] K. Perzel and D. Kane. Usability Patterns for Applications on the World Wide Web. In *Pattern Languages of Program Design 1999 Proceedings*, 1999.

[19] R. Smith. Panel on Design Methodology. In *OOPSLA '87: Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum)*, pages 91–95, New York, NY, USA, 1987. ACM.

[20] C. L. Stimmel. Hold Me, Thrill Me, Kiss Me, Kill Me: Patterns for Developing Effective Concept Prototypes.

In *Pattern Languages of Program Design 1999 Proceedings*, Monticello, IL, 1999.

[21] J. Tidwell. *Designing Interfaces.* OReilly, 2005.

[22] UI Patterns - User Interface Design Pattern Library. Online. Available at: `http://ui-patterns.com/`, Accessed on December 27, 2009.

[23] User Interface Design Patterns. Online. Available at: `http://www.cs.helsinki.fi/u/salaakso/patterns/` , Accessed on December 27, 2009.

[24] D. K. van Duyne, J. A. Landay, and J. I. Hong. *The Design of Sites : Patterns, Principles, and Processes for carfting a Customer-Centered Web Experience.* Addison-Wesley, 2003. Website `http://www.thedesignofsites.com/`.

[25] M. van Welie and H. Traettenberg. Interaction Patterns in User Interfaces. In *7th. Pattern Languages of Programs Conference*, 2000.

[26] M. van Welie and G. C. van der Veer. Pattern Languages in Interaction Design: Structure and Organization. In *Human Computer Interaction - INTERACT 2003*, pages 527–534. IOS Press, 2003.

[27] W. C. Wake. Patterns for Interactive Applications. In *Pattern Languages of Programm Design 1998 Proceedings*, 1998. Available at: `http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P44.pdf`.

[28] Web Patterns - A UC Berkeley Resource for Building User Interfaces. Online. Available at: `http://groups.ischool.berkeley.edu/ui_designpatterns/webpatterns2/webpatterns/home.php` , Accessed on May, 2008.

[29] Welie.com - Patterns in Interaction Design. Online. Available at: `http://www.welie.com`, Accessed on December 27, 2009.

[30] Yahoo! Design Pattern Library. Online. Available at: `http://developer.yahoo.com/ypatterns/`, Accessed on December 27, 2009.

	BOOKS				SCIENTIFIC PAPERS								ONLINE RESOURCES								
HEAD	[10]	[24]	[5]	[21]	[25]	[18]	[7]	[27]	[9]	[20]	[15]	[6]	[29]	[30]	[11]	[16]	[23]	[17]	[13]	[22]	[28]
Pattern Number	✓	✓	✓	✓	-	-	-	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-
Pattern Name	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Alternative Pattern Name	✓	-	-	-	✓	-	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-	✓
Rating/Ranking	✓	-	-	-	✓	-	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-	✓
Sensitizing Image	✓	✓	✓	✓	-	-	✓	-	-	✓	-	-	-	✓	-	-	-	-	-	✓	✓
Problem-Context Summary	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	✓
Author Name	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	✓
Pattern Classification	-	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-
Creation Date	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-
Last Revision	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	✓	-
Hardware	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-
Pattern Level	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BODY																					
Context	✓	✓	-	✓	✓	-	-	✓	-	✓	-	✓	✓	-	-	✓	✓	-	✓	✓	✓
Problem Description	✓	✓	✓	-	-	-	✓	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
Forces	✓	✓	✓	-	-	✓	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-	-	-	-
Solution	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓
Rationale	-	-	-	✓	✓	✓	✓	-	-	✓	✓	-	✓	✓	-	✓	-	✓	✓	✓	-
Diagram	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Resulting Context	✓	✓	✓	-	-	✓	-	-	-	✓	✓	-	-	-	✓	-	-	-	-	-	-
Examples	-	-	✓	✓	-	✓	-	✓	-	-	✓	-	✓	✓	✓	✓	✓	-	✓	✓	✓
Known Uses	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	✓	-	-	-	-	-	-
Counter Examples	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
ADD. INFO.																					
Related Ideas / Literature	-	-	-	-	-	-	-	✓	-	-	✓	✓	✓	-	-	-	-	-	-	-	-
Ref. to Implement.	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-
Code Examples	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	✓	-	✓
Accessibility	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-
REFEREN.																					
Ref. in Text	✓	✓	-	?	-	-	-	-	-	✓	✓	-	✓	-	-	-	-	-	✓	✓	-
Related Patterns	-	-	✓	-	✓	✓	✓	✓	✓	-	-	✓	-	✓	✓	✓	✓	✓	-	-	-
TYPES OF REF.																					
Specialization	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-
Aggregation	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-	-
Association	✓	✓	-	✓	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	-	-	✓	✓	✓	-
Anti-association	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-

Table 4: HCI Design Pattern Content Elements

Research Organization Servicelization Patterns

Yuriko Sawatani
IBM Research-Tokyo
1623-14, Shimotsuruma, Yamato,
Kanagawa, Japan
Tel: +81-462-5157

yuriko@jp.ibm.com

ABSTRACT

Innovation in services has become a topic of interest to researchers due to the worldwide shift to "services" economics. This comes from the growth of services economies and the shift to services businesses by manufacturing industries, including IT-related industries. Innovations, in particular, service innovations are difficult to articulate their structures and mechanisms, due to intangibility coming from services characteristics, and a lack of languages to describe them. In the previous paper [1], we discuss the following two items for service innovation cases: 1. How can we capture the characteristics of innovations as patterns? 2. What are the categories of patterns for innovations? In this paper, I would like to focus on organizational and process aspect of research and service activities, which create service innovations.

Categories and Subject Descriptors

K.6 [Management of Computing and Information Systems], K.6.1 [Project and People Management]

General Terms

Management, Human Factors

Keywords

Servicelization, Service Innovation, Organization, Process

1. INTRODUCTION

Patterns and pattern languages are widely adapted not only technical area, but also social, organizational and management area. Fearless Change: Patterns for Introducing New Ideas" [2] provides patterns to introduce a new idea into an organization. The objective of those patterns is to build a community to discuss interesting ideas spreading the formal business organization. It includes the following pattern categories, Roles (Champion Skeptic, Connector, Corporate Angel, Early Adopter, Early Majority, Dedicated Champion, Evangelist, Innovator, Local Sponsor, Mentor, Respected Techie), Keeping the Idea Visible (e-Forum, Group Identity, In Your Space, Plant the Seeds, Stay in Touch, Treasure, Token), Dealing with Skeptics (Adopt a Skeptic, Champion Skeptic, Fear Less), etc. The structure of these patterns is Name, Context, Problem, Solution, Related patterns. James O. Coplien's "A Development Process Generative Pattern Language" [3] initially takes the structure of Name, Problem, Context, Forces, Solution, Resulting context, and Design Rationale. His book, "Organizational Patterns of Agile Software Development" [4], just follows the same structure of Alexander's pattern language. He developed four pattern languages, such as "Project Management Pattern Language", "Piecemeal Growth Pattern Language", "Organizational Style Pattern Language", and "People and Code Pattern Language", which have intersections of patterns each other.

Patterns and pattern language are widely adapted to software engineering areas, from an architecture level to a programming code level. Adding to the technical design areas, the pattern approach is used in organizations, processes and management areas. In this paper, I look into service delivery process which research organization is involved with to create service innovation. In service delivery projects, some level of involvement of the service receivers is necessary and inevitable. The involvement of the service receivers does not happen only at the beginning of the service delivery, but throughout of the service delivery process. The presence of service receivers creates strong functional interdependencies in a service delivery organization. This functional interdependence between service delivery and service research affects the research lifecycle and its management [5].

In the next section, I describe the following three key patterns of service delivery process.

2. Research Organization Servicelization Patterns

In service delivery, most of information is in intermediary project artifacts, which are hard to transfer, so I focus on the processes of the service delivery. To understand service research activities, I looked into ODIS projects as case studies. First I modeled traditional research activities as three steps: 1. Knowledge proposition, 2. Knowledge creation, 3. Knowledge repository, and the service activities are modeled as the following two steps: 1. Value proposition, and 2. Value co-creation. Knowledge Proposition (KP) is a planning step, when the initial idea of a research plan is developed. The following step Knowledge Creation (KC) is an execution step, which creates knowledge. After the knowledge is created, then it is described in papers and saved in repositories, such as research journals.

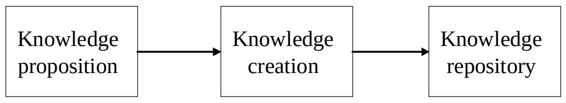


Figure 1. Research activities

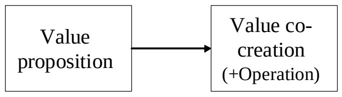


Figure 2. Service activities

For service activities, Value Proposition (VP) is the step in which a proposal is presented to the receivers of the services. When the proposal has been accepted, then proposed value will be co-created by the service providers and the service receivers. The Value Co-creation (VC) step may include operations in which the service receivers use the created value in a value co-creation step.

Due to the interdependencies of the service delivery functions and the dynamic changes of the service project inputs from the service receivers, service research activities are not limited to traditional research activities, but include service activities in the research coverage. Based on the observed paths of the service research activities, ODIS, the following three patterns were found by focusing on the value co-creation step. An evaluation phase was excluded since it was the same in all cases.

- Closed pattern
- Interactive pattern
- Open pattern

1.1 Examples

The inputs and outputs of service research activities are considered as information processing for service systems. The service systems include people, such as service receivers and service providers. The outputs of the service systems are knowledge and knowledge embedded service systems, which are IT-based systems into which the created knowledge is embedded. The inputs of these service systems are mainly information from the service receivers, including end-users, who use the output of the service systems, such as knowledge embedded service systems. The inputs are based on intensity of the communications with the service receivers.

Using these two types of outputs (knowledge base and knowledge embedded service system), and inputs (high intensity and low intensity), we developed a conceptual framework of service systems, as showed in Table 1. Typical service projects are described in each quadrant of each pattern. Projects in the High intensity x Knowledge base quadrant are professional services. The open pattern of service research activities is mapped to this quadrant.

People involved process enhancement, such as CRM, and supported tools, such as Computer Aided Design (CAD), are typical projects in the High intensity x Knowledge embedded service system. These focus on the front stage of a service system, such as service receivers, referring to the theatre model of services by James Teboul. The interactive pattern of service research activities is mapped to this quadrant. Optimization projects using standardized processes, such as SCM, are example of projects in the Low intensity x Knowledge embedded service system quadrant, which are mainly in the back stage of a service system. A part of the back stage service activities needs to be integrated with the front stage of the service system.

TABLE 1

SERVICE SYSTEMS FRAMEWORK: SERVICE PROJECTS CATEGORY

		Intensity of service receivers	
		High	Low
Produced value	Knowledge base	Professional services (Open pattern)	NA
	Knowledge embedded service system	IT supported front stage services (Interactive pattern)	IT supported back stage services (Closed pattern)

2.1 Closed Research Organization Servicelization Patterns

The traditional research activities for product innovation follow this pattern, so this may not have a problem to execute for research organization.

Example:

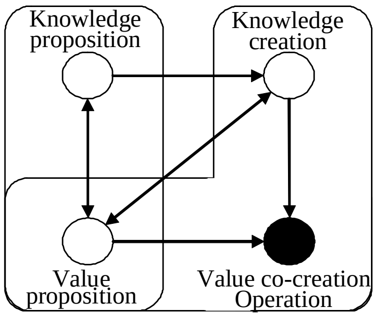


Figure 3 Closed Research Organization Servicelization Patterns

Context: The closed pattern in Figure 3 includes the patterns that end with the step of value co-creation (VC). This pattern is specifically used to solve the predefined problems in service systems. It tends to create knowledge without understanding the current service system. Output examples for this pattern are optimization of IT systems, which are IT-supported systems with enhanced logic created by automating standardized processes.

Problem: Requirements from service receivers are well defined. Research organization provides appreciate technology for the defined requirements from service receivers. If there is no communication gap between service receivers and research, then this shows the similar process with the traditional research activities for product innovation.

Solution: Not so much difference than research activities for product innovation. When there is communication gap between service receivers and research organization, then it needs to have a translator role to have effective communication, such as consultants, science communicators, etc…

2.2 Interactive Research Organization Servicelization Patterns

In this pattern, knowledge are created after the value co-creation activities, which service receivers and providers (in this case, researchers) work together.

Example:

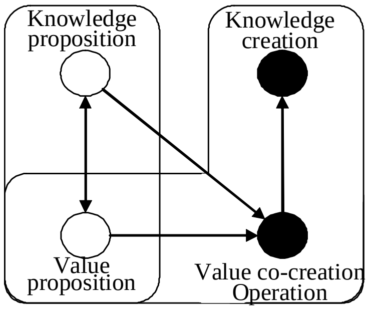


Figure 4 Interactive Research Organization Servicelization Patterns

Context: The interactive pattern in Figure 4 is for creating differentiated values for the current service system, so the researchers need to understand the current service system and do the activities for value co-creation. Output examples of this pattern are Customer Relationship Management (CRM) and Business Process Management (BPM), which are IT-supported systems to improve human-related processes by applying technologies, such as text analytics.

Actors are service receivers, and researchers. Service receivers are persons in service organization, who have request to improve the current situation. Researchers have research area and related technologies which might help a request of service receivers. The responsibility of service receivers is to describe the current issues and requests to solve the issues. The responsibility of researchers is to solve the issues and create service innovation to meet service receivers' requests.

Problem: Service receivers might not be able to express their request clearly. In addition, researchers do not understand their request or issues clearly due to a lack of local knowledge of service environment.

Researchers tend to stick to the current discipline area, and do not explore the issues from service receivers' point of views even though the issues might be solved by the technologies which research organization have or the extension of them.

Even if the key technology could be provided by researchers, but to realize requirements from service receivers, enabling technologies would be required to complete the solution. It would be necessary for researchers to keep interests to solve the entire solution.

Solution: Researchers need to learn local knowledge from service receivers by hearing and data analysis to understand issues and requirements of service receivers in the service system. Researchers work with service receivers to identify where research technologies could contribute.

Researchers need to recognize that research activities for service innovation could not be executed separately from service receivers.

2.3 Open Research Organization Servicelization Patterns

In this pattern, the initial step starts from understanding the issues of the service system which service receivers might not recognize.

Example:

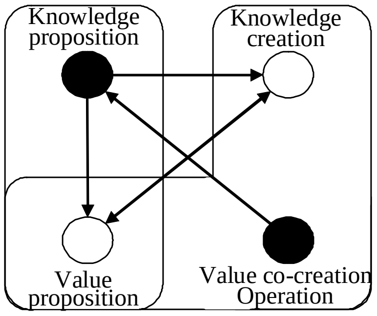


Figure 5 Open Research Organization Servicelization Patterns

Context: The open pattern in Figure 5 starts to understand the current operations by analyzing the service system, and the staring point of these service research activities is the VC. Output examples of this pattern are R & D management services, and innovation management services which mainly produce knowledge for service receivers.

Actors are service receivers, and researchers. Service receivers are persons in service organization, who have issues in the current service environment. Researchers have research area and related technologies which might help a request of service receivers, but mainly need to formalize the current service environment issues as research questions. The responsibility of service receivers is to describe the current issues. The responsibility of researchers is to formalize and analyze the issues, create service innovation with service receivers (value co-creation).

Problem: Service receivers might not be able to come up with service innovation to solve the current issues. Researchers do not understand their issues clearly due to a lack of local knowledge of service environment.

Researchers tend to stick to the current discipline area, and do not explore the issues from service receivers' point of views even though the issues might create a new research area to explore.

In addition, service receivers do not know what service researchers could help and do not expect that their current problems could be solved.

Solution: Researchers need to learn local knowledge from service receivers by hearing and data analysis to understand issues and requirements of service receivers in the service system. Researchers work with service receivers to identify where research technologies could contribute.

Researchers need to recognize that research activities for service innovation could not be executed separately from service receiver. Researchers need to understand issues and requirements of service receivers in the service system of service receivers, where a new research area will be created, by learning local knowledge in the service environment.

Management of service research needs to support for researchers to explore are service research area.

Researchers and service receivers need to create a longer relationship to build trust for the future research activities.

3. REFERENCES

[1] Yuriko Sawatani, et. al, "Innovation Patterns", SSC, 2007

[2] Mary Lynn Manns, Linda Rising, “Fearless Change: Patterns for Introducing New Ideas”, http://www.cs.unca.edu/~manns/intropatterns.html

[3] James O. Coplien , ”Process patterns” http://users.rcn.com/jcoplien/Patterns/Process/process.html

[4] James O. Coplien, ”Organizational Patterns of Agile Software Development”, Prentice Hall, 2004

[5] Yuriko Sawatani, Kiyoshi Niwa, “Service Systems Framework Focusing on Value Creation: Case Study”, IJWET, printing

Adaptable Load Balancing

Sung Kim, Youngsu Son, Gaeyoung Lee
Home Solution Group
Samsung Electronics

ABSTRACT
The proposed load balancing system includes multiple counts of servers for processing network traffics and a client for transmitting connection request signals to those network traffic processing servers.
First, the client sends request signals to all the servers. After receiving those request signals, based on the server resource availability, they calculate the wait time before it sends a response signal back to the client. The client makes the connection to the first server that transmits the response signal and ignores all servers response. Delay time is calculated from system resource availability and predefined maximum tolerance response time for the service. This system combines the use of both local load balancing and global load balancing. By controlling the delay time, it decides which one to put more focus than the other one.
By using this system, the client request can be efficiently distributed. This system is cost efficient because it does not need load balancer and it has flexible architecture.
.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Contructs and Features – *patterns*.

D.2.11 [**Software Engineering**]: Software Architectures – *patterns*.

General Terms
Algorithms, Design

Keywords
Load Balancing, RTT Control, Delay time, Maximum response time, Flexible architecture

1. INTRODUCTION
As requests from the client increases, the servers have to process more transactions.
As the transaction increases, the number of servers in the system has to increase in order to handle those transactions. To efficiently control those increased servers, load balancing method is used. Normally, Load Balancer is needed in order to perform load balancing. However, in this method, it does not require one.

Because Load Balancer is the center gateway of all the transaction, as the transaction count increases, it gets overloaded. Complex load balancing algorithm also adds additional stress to the Load Balancer. In here, for some reason, if Load Balancer breaks down, it will cause the entire service to stop. However, our proposed system does not get this issue since it does not require a load balancer [1].

In the proposed system, the combination of the local load balancing and the global load balancing system is used to adjust the system dynamically depending on the situation/environment.

2. BACKGROUND
Local Load Balancing - Local load Balancing System distributes the clients request to multiple server. With distribution of the load, availability of systems is enhanced. This cause the smart use of servers resource, thus it produce efficient system. Load balancing servers are placed in same location and set up. When the client makes requests to the servers, the client will connect to the server with the lowest load [2].

Global Load Balancing – Global load Balancing System are used when the servers are located in different networks. It is used to located the server with fastest network response. So this considers efficiency of network. The global load balancing system uses redirection method [3]. Load balancing servers are deployed to other place on network. When client request to server, client will connect server that is fastest response.

3. EXAMPLE
For example, let’s assume that we are developing an office automation system for buildings located closely together in a downtown. There are various types of devices in the system and they are connected to a wired or wireless network. In addition, it is required to keep software in each device up-to-date. The server will provide the latest software via client-server model. In Polling method, each client requests data from the server without considering any other clients. So, it would cause server overload. On the other hand, in Push method, a client that is turned off or malfunctions at the time of upgrade wouldn’t be updated.

Let’s take a specific example with the figure below. The server has to update various types(green, yellow, red) of devices that are placed in different location. Some office would have all types of devices but some would not. In this situation, it is possible for the server to manage devices in a way that groups them by device type or location.

4. CONTEXT
Environments like websites which services various type of contents, such as video feeds, html/image rendering, puts stress to both network and server resources. These require 2 types of load balancing. The global load balancing focuses on network related

issues. The local load balancing focuses on server resource related issues [4].

In Local Load Balancing, a load balancer is located at the server side and distributes clients' requests only based on the resource availability of the servers. It does not take into consideration of the network status or service type (whether it is video streaming service or html data transmission service)

Global load balancing balances the load by taking into consideration of requests Round Trip Time(RTT).

The local load balancing is able to balance the server overload, but it is difficult to consider external conditions, e.g. network bandwidth, systematic vaccine update, and so.

The global load balancing is apt for providing clients with fast response.

5. PROBLEM

Due to the workload of server and the status of network changes constantly, these changes should be considered and reflected in real time. So to balance the load, we should consider multiple factors, e.g. Round Trip, Load of Sever, Weight of Load, load item count, maximum response time, Delay time, RTT

6. FORECES

The following items should be regarded as forces:

- Providing load balancing without the load balancer. Thus prevents the service stop when the load balancer breaks down. Being able to adjust which one to focus more, between local load balancing and global load balancing, depending on the situation.
- Ignoring any server that does not respond, from the load balancing list.

7. SOLUTION

7.1 Structure

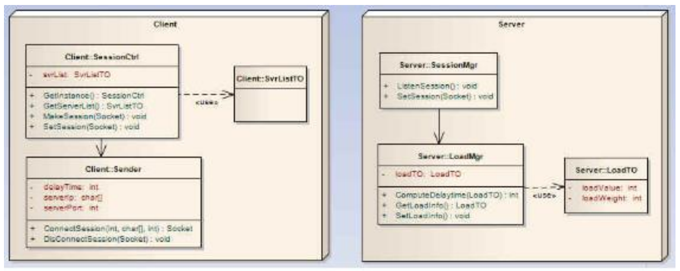


Figure 1. Half-Push/Half-Polling Structure

multiple count of servers for processing network traffics and a client for transmitting connection request signals to those network traffic processing servers. The proposed method includes multiple counts of servers for processing network traffics and a client for transmitting connection request signals to those network traffic processing servers. The proposed load balancing method structure is shown in Figure 1.

SessionCtrl Component controls the session between server and client. Sender Component performs connection and disconnection.

- GetInstance : Start load balancing.
- GetServerList : Get loading balancing target server list..
- MakeSession : Make session to server.
- SetSession : Set session that connects to a server transmitting the first received response signal.
- ConnectSession : Sends request signal to server.
- DisconnectSession : Disconnects all others except to the first session.

SessionMgr Component transmits connection request signals and registers client session. LoadMgr Component computes delay time according to the algorithm.

- ListenSession : Receives client request signal and responses to the client after delay time
- Set(Register)Session : Registers a client session.
- ComputeDelayTime : Compute the delay time from the server load.
- GetLoadInfo : Get values of server load items.
- SeLoadInfo : Set weight of load items.

7.2 Algorithm

An algorithm exists that controls the server response time. Clients transmit connection request signals to the servers. The servers receive clients connection request signals and compute the delay time (DT) from the server load. During the computed Delay Time (DT), the servers will delay to send response signal to the client. In result, if the server load is big, the response time will be delayed that much.

Variables for load balancing algorithm are shown in table 1.

Variable	Value
RTT	Round Trip Time
LT	Load Type- - CPU, Memory … of server.
LW	Weight of load
LC	Count of load item
MT	Maximum response time
DT	Delay time
DRTT	RTT considered load.

Table 1. Variables for load balancing algorithm

RTT is the required time for network communication to travel from the client to the server and back. LT is the server load type – CPU usage, memory usage, etc. LW is the priorities amongst the load types. LC is count of LT. MT is the maximum response time that takes client to get the response back from the server after sending request signal.Load balancing algorithm is expressed in following equations.

DT (Delay Time)

$$DT = \frac{\sum_{i=1}^{LC} LTi \times LWi}{LC} \times MT$$
$$(i = 1,2,3\ldots, LC)$$

DRTT (RTT considered Server Load)

$$DRTT = RTT + DT$$

(Round Trip Time(RTT) gets added to the delay as well)

The type of data being transmitted/processed affects the load of server resource, Load Type(LT).
Depending on the data being video steaming or html data, the server resources would have different load, one resource (such as CPU) getting priority from other resources. An algorithm can be used to apply different weight to these resources.

Also server load count (LC) can be increased or decreased depending on the type of data.. For example, Load Types are memory usage, CPU usage, disk usage, etc. If server is influenced by CPU usage the most, The CPU usage will have the biggest weight. As result, the increase of CPU usage will influence DT, in turn, DT will be increased. In other words, DT is sensitive to the CPU usage change.

MT decides how big the delay time can get. If MT is increases, DRTT will be influenced that much If MT decreases, DRTT will be influenced that much less. The other hand, if MT decreases, the influence of RTT will be bigger than DT because the weight of RTT is more than weight of server load. In result, the balance of focus between the local load balancing and the global load balancing is controlled by MT.

7.3 Dynamics

Sequence of load balancing method is shown in Figure 2.

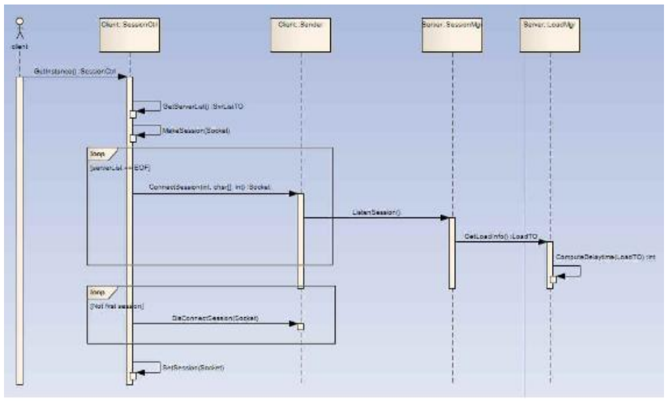


Figure 2. Sequence of load balancing method

Client has server list. It also sends the request signal to server. First, the client sends request signals to all the servers. After receiving those request signals, based on the server resource availability, they calculate the wait time before it sends a response signal back to the client. The client makes the connection to the first server that transmits the response signal and ignores all servers response.

When client sends request signal and receives response signal from server, DT is influenced by RTT automatically. Thus, RTT is important. Therefore client is connected to server that has the best one when considering the low load or the fast RTT.

If a server breaks down, the server will not be able to respond. Those broken-down servers will be ignored from load balancing.

8. Experiments

In our experiment environment, the system was composed of two servers and one client. Network is LAN environment. The experiment compared the round robin method with the proposed method. Different request types affect server resource usage differently as shown in below table.

Request Type	CPU Usage	Memory Usage
HTTP Transaction	1%	1%
DB Transaction (A Type)	3%	1%
DB Transaction (B Type)	1%	3%

Table 2. Assumption values of load

As shown in the table 2, A Type of DB Transaction influence more on CPU usage than memory usage. B Type of DB transaction influences more on memory usage than CPU usage.

#Experiment 1 – Were tested with Server1 and Sever2 having same Value of variables (CPU, Memory, RTT).

Name of variable	Value of variable
RTT	Server1 : 300ms, Server 2 : 300ms
LW	CPU : 50, Memory : 50
LC	2 (CPU, Memory)
MT	3000ms

Table 3. Value of variable for LW experiment

Result of the experiment is shown in Figure 3.

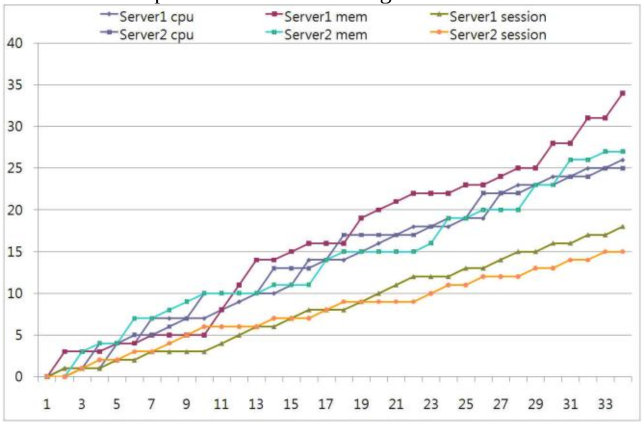


Figure 3. result of LW Experiment

In Result, the number of sessions in two servers came out to be different. However resource usage between server 1 and server2 is similar. As Figure3 indicates, the experiment #1, caused the server's load balancing to act like the local load balancing.

# Experiment 2 – Servers were setup with different network environments.

Name of variable	Value of variable
RTT	Server1 : 100ms, Server 2 : 300ms
LW	CPU : 50, Memory : 50
LC	2 (CPU, Memory)
MT	3000ms

Table 4. Value of variable for RTT experiment

Result of the experiment is the same as Figure 4.

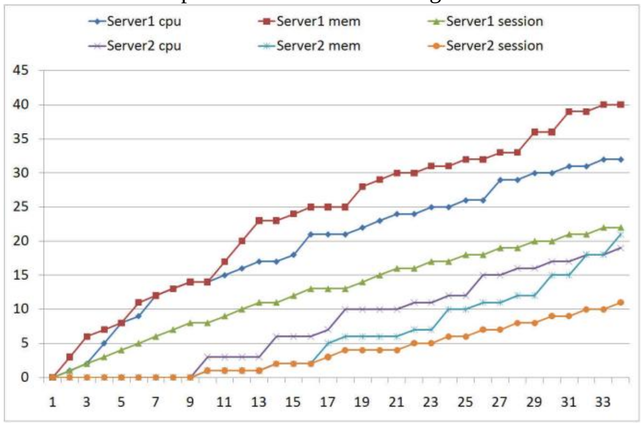


Figure 4. Result of RTT experiment

As Equation #1 suggest, the experiment #2 with low MT value caused DT to be small. Since DRTT is the sum of DT and RTT, low DT insinuates DT having less influence to the DRTT than the RTT value influencing DRTT. As Figure 4 indicates, the experiment #2, caused the server experiment #2, cause to act like the global load balancing.

# Experiment 3 – Added more value on CPU than memory.

Name of variable	Value of variable
LW	CPU : 70, Memory : 30
LC	2 (CPU, Memory)
MT	3000ms

Table 5. Value of variable for LW experiment

Result of the experiment is shown in Figure 5.

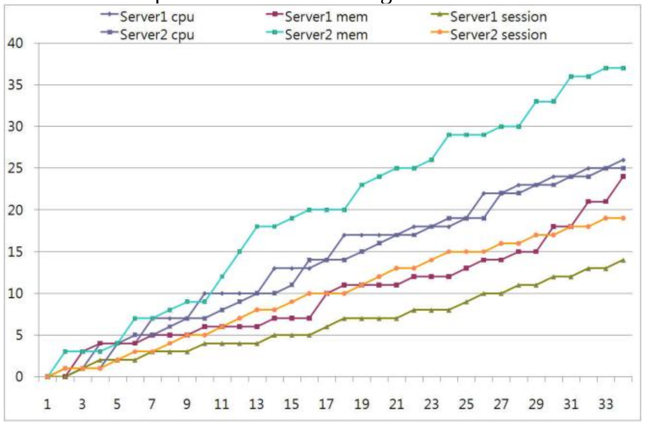


Figure 5. Result of LW experiment

As seen in Figure #5, the experiment result shows the servers having different session counts and memory usage from each other. However, CPU usage seems to be similar on those servers.

#Experiment 4 – Increased the values of MT.

Name of variable	Value of variable
RTT	Server1 : 100ms, Server 2 : 300ms
LW	CPU : 50, Memory : 50
LC	2 (CPU, Memory)
MT	10000ms

Table 7. Value of variable for RTT and MT experiment

Result of the experiment is shown in Figure 7.

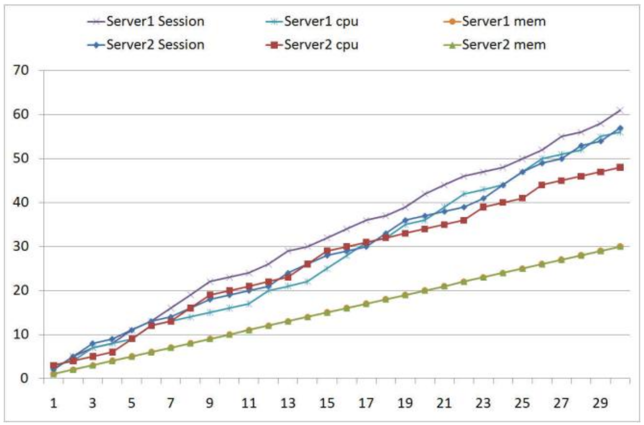


Figure 6. Result of RTT and MT experiment

As Equation #1 suggest, the experiment #4 with high MT value caused DT to be high. Since DRTT is the sum of DT and RTT, high DT value insinuates DT having bigger influence to the DRTT than the RTT value influencing DRTT.
As Figure 6 indicates, the experiment #4, caused the server's load balancing to act like the local load balancing due to having less influencing RTT value to DRTT.

9. Side Effects

As the number of the servers increases, the difference in the time that each servers receive the request signal sent by the client increase. This time difference affects the load balancing.

10. RESULTING CONTEXT

The proposed method can perform load balancing without the load balancer. Thus, it automatically ignores any server that does not respond.
The Value of variable gets set depending on the importance of each resource in server.
The load balancing is performed by considering these resources with high values. From observing the resource usage and RTT, the proposed method uses the load balancing according to the environment.

ACKNOWLEDGMENTS

We thank our shepherd Norihiro Yosida for his valuable comments that contributed to improve this paper. The EVA (Pattern Evangelist Group), James Chang provided useful improvements and corrections.

REFERENCES

[1] Valeria Cardellini, Michele Colajanni, Philip S. YU, “Dynamic Load Balancing On Web-Server System”, IEEE Internet Computing, June 1999.

[2] Network Load Balancing Technical Overview White Paper

[3] Valeria Cardellini and Michele Colajanni and Philip S. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-server Systems", IEEE 1999.

[4] CDN Conference, NetTrend 2001, 2001.

[5] Msurice Castro, Michael Dwyer and Michael Rumsewicz, “Load balancing and control for distributed World Wide Web servers”, IEEE, August 1999

[6] Rajkumar Byyya, “high performance cluster computing”, Prentice Hall PTR, 1999.

[7] R.j. Schemers, “lbmnamed: A Load Balancing Name Server in perl”, Proc, 9th Systems Administration Conf, Uscnix Assoc, Berkely, Calif, Sept 1995.

[8] A. Bestavros et al, “Distributed Packet Rewriting and its Application to Scalable Web Server Architectures”, IEEE Computer Soc. Press, Los Alamitors, Calif., 1998

[9] Carla Sadtler, John Chambers, Ariane Schuldhaus, “Load Balancing for eNetwork Communications Server” International Technical Support Organization