Towards Mitigating Human Errors in IT Change Management Process

Venkateswara R Madduri[1], Manish Gupta[1], Pradipta De[1], and Vishal Anand[2]

[1] IBM Research India, Delhi
[2] IBM Integrated Technology Delivery, Bangalore, India

Abstract. IT service delivery is heavily dependent on skilled labor. This opens the scope for errors due to human mistakes. We propose a framework for minimizing errors due to human mistakes in Change Management process, focusing on change preparation and change execution. We developed a tool that brings better structure to the change plan, as well as, helps the change plan creator in developing a plan faster through use of knowledge-base and automatic guidance. At the change execution phase, we designed and implemented an architecture that intercepts and validates operator actions, thereby significantly reducing operator mistakes. The system can be tuned to vary the involvement of the operator. We have tested the system in a large IT delivery environment and report potential benefits.

1 Introduction

IT service delivery has seen unprecedented growth. To tackle the surge, IT service management has become a human labor intensive industry. The strong dependence on human workforce leads to several outages which can be traced back to human mistakes [5, 3, 2, 4]. Outage in service delivery stems from several factors, starting from hardware failures to simple misconfiguration of an application, leading to service downtime.

This paper proposes a framework for minimizing outages, that are triggered by human errors, in service delivery environment. We look closely at the *change management process*, as defined by ITIL [1]. A typical change management lifecycle involves: (i) a change request is raised and logged into a change request system, (ii) an expert team reviews the problem and draws up a change plan, (iii) designated change assignee executes the steps as documented in the change plan, (iv) changes are validated and the change ticket is closed. *Starting from change plan preparation to change execution, the process is heavily dependent on skills of the human workforce.*

We propose a solution targeted at two levels: change preparation and change execution. At change preparation, errors could be due to inadequately specified instructions, reuse of older change plans, and omission of instructions for rarely used stages, like pre-validation and backout. We introduce a column-based structure for change plan creation. An MS-Excel plugin based wizard, referred to as TexExpeditor, helps in building a structured change plan. Standardized structure, along with several functionalities to guide the change plan creation.

The proposed method to prevent errors during change execution is based on the idea of intercepting operator actions. The system, referred henceforth as Facade, checks the correct time window for the change execution, as well as, validates the server address on which the change must be acted. The interceptor is also capable of automating command execution by matching command outputs. Facade can also run in supervised mode where it waits for the user to indicate successful command execution.

In this paper, our key contributions are two-fold: we propose a standardization for change plan creation, and provide a tool to create standard change plans; and based on this standard template, we have designed and implemented a change execution engine, called Facade, which can be tuned to run at different levels of automation. In the rest of the paper, we present a broad overview of the tools (Section 2), followed by details of the implementation. We also present a set of results from our limited engagement with accounts (Section 4).

2 System Overview

We have developed a system that addresses the challenges of Change Management at two levels: change preparation and change execution. In this section, we describe the overall design of two complementary tools: TexExpeditor used for change plan creation, and Facade used for mitigating errors during execution of the plan.

2.1 Change Plan Creation

In most service delivery environments, Subject-Matter-Expert (SME) creates the change plan. A change plan is usually a set of instructions in unstructured text. Use of unstructured text leaves scope for misinterpretation during execution. Keeping the current state in mind, the key goal of the change plan creation phase is two-pronged: (i) to reduce ambiguity in the definition of execution instructions in a change plan, and (ii) reduce the effort, as well as, chance of mistakes in the creation of change plan.

We introduce a structure in the change plan to reduce ambiguity. The proposed change plan is a set of tasks, with each step in the task described as a record with fixed number of fields. The fields are <command, node to execute on, type of command, comments>. To aid the process of plan creation, we have implemented a MS-Excel based wizard to guide in plan creation (details in Section 3.1). Figure 1 shows the workflow of change plan creation and that of new change plan template generation. During the plan creation, a user can provide the keywords describing the change, which will pull out relevant plans for reference.

2.2 Change Plan Execution

At present, a change plan is written in an unstructured manner leading to misinterpretation of the instructions, and errors in execution. Other sources of error

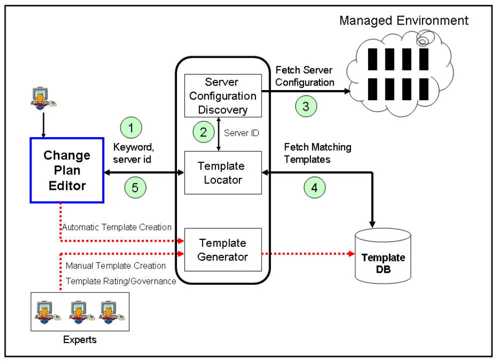


Fig. 1. Steps in Change Plan Creation using TexExpeditor tool

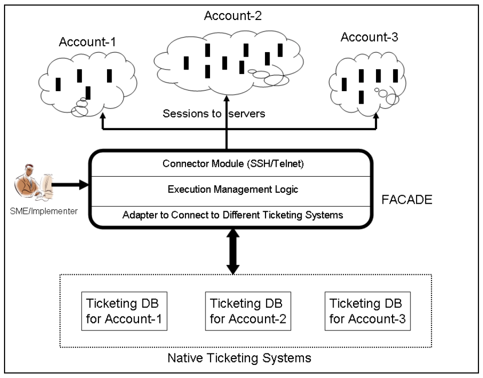

Account-2
Account-1
Account-3
Sessions to servers
Connector Module (SSH/Telnet)
Execution Management Logic
Adapter to Connect to Different Ticketing Systems
FACADE
SME/Implementer
Ticketing DB for Account-1
Ticketing DB for Account-2
Ticketing DB for Account-3
Native Ticketing Systems

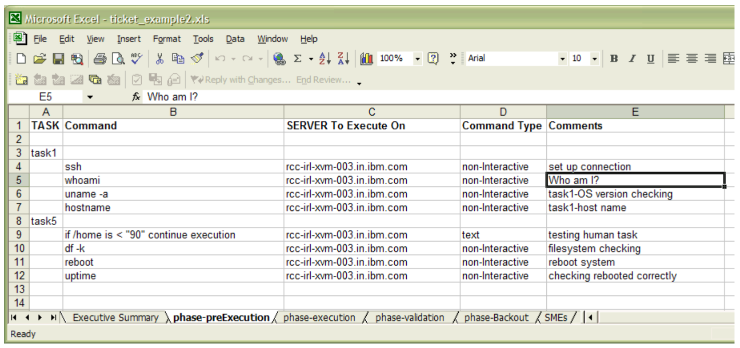
Fig. 2. Overall Architecture of Facade

in this manual mode of change execution are, (i) the user mistakenly logs into the incorrect server and performs changes, (ii) being in a different time zone from the target servers, the implementer may start execution at wrong change window,

Facade execution engine is a gateway before any change action reaches the target server, as shown in Figure 2. The ticketing system holds the change requests and change plans for target servers. Facade architecture has three important modules which makes it flexible to incorporate new ticketing systems and new connection types. The adapters to the ticketing system perform the task of fetching relevant tickets for display to the user. The connector module opens sessions to the target hosts specified in the change plan. Since the connections are opened automatically by reading in the hostname from the change plan directly, this eliminates possibility of wrong server login. Before executing any change command on the target host, the change time window is checked with the server time to ensure that a change is restricted within its change window. In order to make the transformation from the current state-of-the-art to a fully automated change plan execution, we have implemented an extensible framework. The flexibility provided in the semi-automatic mode is that the user analyzes the output of a command and decides on the progress of the change, while in the automatic mode the command is pushed to the endhost, output analyzed and the next command is executed.

3 Design Choices and Implementation

The design choices and implementation of the change preparation tool (TexExpeditor) and change execution engine (Facade) are guided by several business and practical constraints. In this section, we will present the requirements that shaped our design choices, followed by the implementation of the tool.



Who am I?

	A	B	C	D	E
1	TASK	Command	SERVER To Execute On	Command Type	Comments
2					
3	task1				
4		ssh	rcc-irl-xvm-003.in.ibm.com	non-Interactive	set up connection
5		whoami	rcc-irl-xvm-003.in.ibm.com	non-Interactive	Who am I?
6		uname -a	rcc-irl-xvm-003.in.ibm.com	non-Interactive	task1-OS version checking
7		hostname	rcc-irl-xvm-003.in.ibm.com	non-Interactive	task1-host name
8	task5				
9		if /home is < "90" continue execution	rcc-irl-xvm-003.in.ibm.com	text	testing human task
10		df -k	rcc-irl-xvm-003.in.ibm.com	non-Interactive	filesystem checking
11		reboot	rcc-irl-xvm-003.in.ibm.com	non-Interactive	reboot system
12		uptime	rcc-irl-xvm-003.in.ibm.com	non-Interactive	checking rebooted correctly
13					
14					

Fig. 3. A typical change plan along with the wizard for creating a plan

3.1 TexExpeditor: Design and Implementation

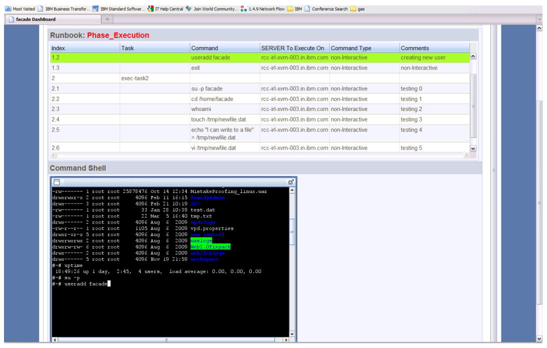
Several factors lead to making mistakes during change plan creation. *Cut-Paste error* is the classic case of reusing old plans to create new ones, and forgetting to change a parameter in the new plan. *Omission error* happens when an important field, like patch version, is omitted during plan creation. Another common practice is to *omit backout plan* for simple changes, assuming there will be no complication during change.

TexExpeditor tool forces a structure while building the change plan, as well guides the user with recommendations. It is built on top of MS-Excel, and opens a wizard to guide the plan creation. Six mandatory worksheets are defined: *Executive Summary, Pre-execution Phase, Execution Phase, Validation Phase, Backout Plan, SME list.* Any violation of the structure alerts the user. For example, while filling out a sheet, if incorrect type is entered in a column, e.g. host address does not match IP format, then an error is raised. The wizard is shown in Figure 3. We also create a change plan store which can be searched based on keyword. Search for a relevant plan is based on keywords. The task store is a 3-way dictionary with keywords describing *Actions, Software and Platform.* Sample keywords for Action are Install, Upgrade, Update, Modify, Patch, etc; for Software is DB2, Oracle, Websphere, etc,; and for platform are Linux, Windows, AIX, etc.

3.2 Facade: Change Plan Execution Engine

Several observations guide the Facade design. Complex changes often require human intervention, and therefore, Facade must be tuned to switch from automatic to supervised mode. Multiple changes could be implemented simultaneously. Once a change has started, Facade must ensure that no other person by mistake accesses the same ticket; this requires maintaining session state of the ticket. An alerting mechanism is necessary to intimate relevant stakeholders on exception scenarios.

Facade is a web application with a web-based Graphical User Interface (GUI), as shown in Figure 4. The web UI allows several features, viz. Role-based access



Runbook: Phase_Execution

Index	Task	Command	SERVER To Execute On	Command Type	Comments
1.2		useradd facade	rcc-irl-xvm-003.in.ibm.com	non-Interactive	creating new user
1.3		exit	rcc-irl-xvm-003.in.ibm.com	non-Interactive	non-Interactive
2	exec-task2				
2.1		su -p facade	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 0
2.2		cd /home/facade	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 1
2.3		whoami	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 2
2.4		touch /tmp/newfile.dat	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 3
2.5		echo "I can write to a file" > /tmp/newfile.dat	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 4
2.6		vi /tmp/newfile.dat	rcc-irl-xvm-003.in.ibm.com	non-Interactive	testing 5

Command Shell

```
-rw------- 1 root root 25878476 Oct 14 12:34 MistakeProofing_linux.war
drwxrwxr-x 2 root root     4096 Feb 11 16:15 RemedyAdmin
drwx------ 3 root root     4096 Feb 21 10:19 SCC
-rw------- 1 root root       33 Jan 28 10:39 test.dat
-rw------- 1 root root       22 Mar  5 16:40 tmp.txt
drwx------ 2 root root     4096 Aug  6  2009 updilogs
-rw-r--r-- 1 root root     1105 Aug  6  2009 vpd.properties
drwxr-xr-x 5 root root     4096 Aug  6  2009 was_install
drwxrwxrwx 2 root root     4096 Aug  6  2009 waslogs
drwxrw-rw- 6 root root     4096 Aug  6  2009 Web2.0fixpack
drwx------ 2 root root     4096 Aug  6  2009 web2feplogs
drwx------ 5 root root     4096 Nov 19 21:59 workspace
#-# uptime
 18:49:26 up 1 day,  2:45,  4 users,  load average: 0.00, 0.00, 0.00
#-# su -p
#-# useradd facade
```

Fig. 4. Facade Graphical User Interface for executing a change plan

control, Multiple ticket execution, Failed execution handling. Different users, like Subject-Matter-Expert(SME) and an implementer, get different views of the ticket. Separate tabs are opened for each ticket, so that a user implementing multiple changes can navigate across tickets. When exception occurs during a change, Facade allows user to abort a ticket, and triggers the backout plan. It also alerts the list of SMEs mentioned along with the ticket, and updates are sent to the Ticketing system.

Three key building blocks in Facade framework are interface to the native ticketing systems, the session management and interceptor modules, and end-host connection manager. Tickets are accessed from ticketing system using web-service interfaces. For example, BMC Remedy ARS [6] exposes web service APIs, that is used to implement access to the ticketing database.

Once a change begins, Facade maintains a session for the ticket to guide the user with the progress. Facade maintains the current command being executed on the endhost, and proceeds to the next command when previous command is successfully completed. In supervised mode, user indicates successful completion; in automatic mode, it is detected by Facade. There are two types of commands in a change plan, non-interactive, and interactive/text based. For the text based commands, Facade opens a putty session to the designated endhost and allows the user to execute the change directly on the target host. However, the chance of an error in terms of execution on a wrong node is precluded by opening the putty session to defined host.

The connection management to the target host is the other important functionality in Facade. Since all target hosts are now accessed through Facade, this allows the ability to control access to endhosts. If due to incorrect scheduling of change, multiple tickets try to access the same endhost, Facade can easily intercept and prevent. For connecting to the endhost, Facade uses the same authentication method, like ssh or telnet, and asks the user for the credentials.

4 Results and Observations

This section presents preliminary results collected from a test engagement. The results are indicative of the benefits of the tools. In order to understand the benefit of TexExpeditor, we studied over 300 change requests over a period of 6 months raised for database operations team at a large telecom account. Approximately 60% of the changes were repeating, where some of the major keyword classes were *dba, patch, update, database, upgrade, migration*, with changing tablespace in DB being the most common activity.

During change execution, Facade was useful for simple changes, but more supervision was required for complex changes. Facade was able to prevent execution of about 50% of the changes which were attempted. We prevented execution of tickets whose change time window has not been reached, and those which were pending approval. The web based display of the execution result is difficult to read for the user. However, in our upcoming version, we are integrating a VT100 terminal emulator to maintain the standard look-and-feel for the user.

5 Conclusion

In the current human labor intensive IT delivery model, scope for errors due to human mistakes cannot be precluded. We designed a framework that minimizes the chance of human mistakes in Change Management process. We target two key stages to restrict the errors: change preparation and change execution. TexExpeditor tool introduces a structure in change plan creation, and guides the user during change plan creation, thereby reducing the chances of making common mistakes. Facade execution engine acts like a validation system before a user can start executing change on the target host. It intercepts operator actions, and allows execution only after validating the correct execution parameters, like correct time window, correct target host. Facade can be tuned to run in a semi-supervised mode, as well as, execute changes automatically on the endhost. We envision that commonly occurring changes will benefit significantly from the automated execution framework of Facade, while complex changes will involve human intervention.

References

1. It infrastructure library. itil service support, version 2.3. In *Office of Government Commerce*, 2000.
2. Theophilus Benson, Sambit Sahu, Aditya Akella, , and Anees Shaikh. A first look at problems in the cloud. In *Proceedings of the 2nd Workshop on Hot Topics in Cloud*, 2010.
3. Jim Gray. Why do computers stop and what can be done about it. In *Proc of SRDS*, 1986.
4. Fabio Oliveira, Andrew Tjang, Ricardo Bianchini, Richard P. Martin, and Thu D. Nguyen. Barricade: Defending systems against operator mistakes. In *Proc. of Eurosys*, 2010.
5. D. Oppenheimer, A. Ganapathi, and D. Patterson. Why do internet services fail, and what can be done about it. In *Proc. of Usenix Symposium on Internet Technologies and Systems*, 2003.
6. BMC Remedy IT Service Management Suite. http://www.bmc.com/products/offering/bmc-remedy-it-service-management-suite.html.