# Facade : Collaborative Creation, Peer Review, and Execution of IT Tickets

IBM Research : Pradipta De, Manish Gupta, Venkateswara R. Madduri, Jai K. Singh

***Abstract*—Cost pressures on IT Service Delivery along with greater employee churn, and lack of skilled labor and training problems have resulted in lower customer satisfaction. As of today a typical system administrator, who is tasked with executing a server *incident* or *change* ticket follows a very ad hoc process of performing the task. This ad hoc approach often leads to poor quality of execution resulting in service level failures such as incorrect, incomplete, or not fully thought through work instructions, lack of a backout plan, task performed without authorization, instructions if provided were not followed, executing the task at a time when it is not supposed to be done, or executing the task for a wrong server, or skipping a command. Some key factors behind all these issues are: (1) diversity of tools that a user is expected to be familiar with, (2) lack of a standard in representing the work-instructions, and (3) the current practice of creating and executing work-instructions which are open to misinterpretation during execution. In this paper, we describe the design of a tool that alleviates the above issues. The tool has been implemented and used in 3 pilots involving large service delivery projects. We present usage experience of this tool from the pilots conducted to validate our design.**

**Keywords-Service Management; Service Planning and Execution; Service Delivery Platforms and Architectures; Change & Incident Management;**

## I. INTRODUCTION

ITIL [1] lays down the best practices for IT processes concerning incident, problem, and change management, yet it does not dwell on the *lifecycle* of how a system admin *creates*, *harvests*, and *implements* work-instructions (*techplan* in short) for a given change or incident/problem ticket. Despite the growth in the number of customer accounts being delivered by a large strategic outsourcing provider, the above lifecycle often evolves differently for each account and largely remains lacks a structured approach. The ad hoc approach may be described as follows: a change ticket is typically executed without a formal plan detailing the techplan or the commands that need to be executed. The system admin who is executing the techplan has what needs to be done in his/her mind or noted in one or more documents on his/her workstation. There are potentially multiple windows open on the workstation and the user sifts through these windows to execute the next command for a server. Typically commands are manually copy-pasted from the personal notes and documents on the workstation to a server client window like PuTTY [8]. The execution logs are typically not harvested for future analysis. As more and more customers are being onboarded, service providers are grappling with how to increase productivity and prevent human errors from occurring in executing the tickets while still having lower-skilled personnel execute tickets given the complexity of the managed environment. It has been observed for most complex large scale systems, errors are often due to operator errors [3]. Research efforts in mitigating such human errors in IT management has looked at different ways of understanding the causes of such operator errors [7], modeling configuration problems to alleviate chance of errors [5][6], as well as, proposed techniques to prevent or counter such errors [2].

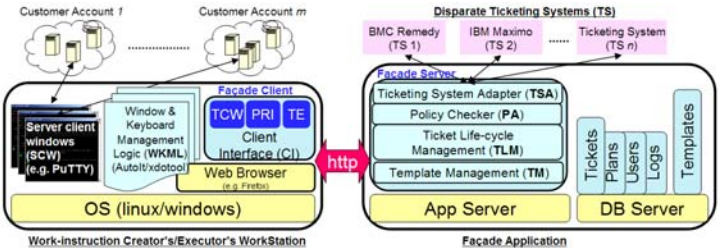


**Figure 1: Framework for Techplan Creation and Execution**

In this work, we propose a detailed design of a web-based framework where the different facets of the *lifecycle* of ticket execution namely (1) creation of a techplan, (2) an optional peer review of the plan, and (3) implementation of the plan are seamlessly unified. Since at present a bulk of the process is manually driven, therefore it is prone to human errors leading to unplanned downtime for the servers. The **main contributions** of this paper are: (1) Identification of the *minimal* data fields and *techplan format* necessary for a system admin to address the life-cycle of a ticket, (2) a web-based design that allows for *collaborative* technical plan creation and review, and (3) a design that uses existing server client software (like PuTTY [8] for windows, xTerm for linux) and yet *precludes common human errors* like implementation of the techplan at the wrong time, or on wrong server, or skipping of a command, (4) presentation of the *actual usage experience* in 3 pilots in a large IT service delivery provider.

## II. DETAILED DESIGN AND TOOL COMPONENTS

A service request for change or incident is represented as an electronic document called *ticket* in different ticketing systems (TSs), like BMC Remedy, IBM Maximo, Manage Now. Each TS has its own representation of the ticket making it hard for even an experienced user to learn the nuances and style of each one of them. After the ticket is created, a *techplan* (containing detailed work instructions/commands) is created by the *creator* and attached to the ticket. An *executor* potentially different from the *creator* may implement the ticket. The entire process of creation and execution is usually ad hoc. We next propose a framework, called Facade, to alleviate the current ad hoc approach in creation and execution of a techplan.

In Figure 1, we show Façade's client-server architecture. Ticketing System Adapters (*TSAs)* virtualize the disparate TSs and pull *minimal* information necessary for the creator and the implementer to perform their tasks. The creator uses a web browser, say Firefox, on his workstation to access the Techplan Creation Wizard (TCW) which is part of the Client Interface (CI) to create a techplan as well as create *templates* for

subsequent *sharing* and *reuse*. The CI also has a sub-component, called PRI, to submit techplan/templates for review by experts/peers for preventing any error. The executor uses the TE subcomponent of CI to execute the techplan. All the actions are automatically recorded for future verification and analysis. TLM provides the server-side logic for TCW, PRI, and TE.

### A. *Ticket Details fetched from Ticketing Systems*

Ticketing Systems are overloaded with many attributes, which may be useful in different contexts. With help from subject matter experts (SMEs), we identified that 7 key attributes are necessary for techplan creation, review, and execution. These are *id*, *description* (for describing the ticket), *start* and *end time* of ticket, *assignee* (designated person for the ticket), *submitter* (one who submitted the ticket in the TS), *status* of the ticket in TS. Thus Facade is able to provide a uniform view of a ticket pulled from diverse TSs, without cluttering the user's view with information not relevant to him for the task.

### B. *Collaborative Techplan Creation and Peer Review*

A techplan consists of a *high-level plan (HP)* and *Task Specifications (TSpec)*. HP consists of high-level *tasks*, where each high-level task (HLT) is elaborated further in *TSpec* in terms of either sub-tasks or Unix commands/script. Mentioned along with each HLT are the server(s) to execute the task on (we allow same task to be executed in parallel on multiple servers). TCW is used to collaboratively and asynchronously create the techplan. Evolution of techplan as it changes is maintained via versioning. Another feature is that a plan creator can create templates (of whole plans or of tasks) by introducing variables for command or task parameters. We have analyzed tickets from a large telecom account to discover that repeatability of tasks across plans is quite common [4]. A techplan can be submitted for peer review to multiple reviewers who are alerted by email. Reviewers provide rating, risk, and modification to the techplan (using TCW) that helps improve the quality of the plan, and speeds up the approval process.

### C. *Techplan Execution and Audit Management*

The TE component provides a GUI button for each High-Level Task (HLT) to launch a PuTTY window for each of the servers specified for the HLT. PA (Policy Checker) allows the button press to result in a PuTTY window opening only when the ticket is approved and only within the ticket time window. For each command within a HLT, a GUI button is provided to *push* the text of the command simultaneously to (but *only* to) the servers specified corresponding to the HLT. This prevents typos, as well as ensures *execution of a command on the correct server*. AutoIt/xdotool [10] is used to provide WKML (see Figure 1) for pushing the command text to PuTTY windows. The executor monitors the result of the current command execution to decide upon executing the next command or aborting. If a command is skipped the TE provides with necessary prompts for the user. If Facade Client runs on a Linux workstation then xTerm is launched for SCW. TLM component is responsible for recording all the actions of the creator, reviewer, and the executor vis-à-vis Facade along with the PuTTY execution logs. The logs provide valuable information in the event of a failed plan for performing forensics on the causes of failure.

## III. USAGE EXPERIENCE RESULTS FROM PILOTS

We hosted the Façade web-application from India. IT delivery centers across the world, India (#Tickets: 170, #Users: 14), Argentina (#Tickets: 50, #Users: 4), and Brazil (#Tickets: 12, #Users: 2), participated in usage tests. Figure 2 presents the survey on the usage and benefits of Facade. Based on the promising result, deployment in one delivery center is in progress.

| Benefits of Facade for implementing incident/change tickets | Approval Rating % ♣ | | |
|---|---|---|---|
| | India | Argentina | Brazil |
| **Avoid unauthorized changes**: Prevents changes to be executed outside the corresponding window schedule. | 100 | 100 | -* |
| **Traceability/Auditing**: Provides automatic logging for unix sessions; easy to retrieve these logs; user time spent on each task can be calculated | 100 | 100 | 100 |
| **Collaboration**: Easy-to-use peer review capability to improve quality of techplan | 85 | 100 | - |
| **Errors reduction**: Details and commands of each task are pre-loaded on the tool thus reducing human errors; open PuTTY to the correct servers | 100 | 100 | - |
| **Templatization**: Ability to create templates ahead of time results in a better plan for execution; Reuse of plans for repetitive deployments | 85 | 100 | 100 |
| **Multiple Servers**: send a command in one shot to multiple server windows | 85 | - | 100 |

♣ percentage of users who agreed with the benefit mentioned in the 1st column
* When a '–' is specified then no response was received from the corresponding pilot team

Figure 2: Usage Experience from 3 Pilots

## IV. CONCLUSION & FUTURE WORK

The observed benefits of a structured framework are paving the way for use of tools, like Façade, in IT delivery operations worldwide. Some of the key further requirements expressed are: extension to Windows RDP and use of PowerShell, enhance techplan formats to include programming constructs, automation in executing the next command via evaluation of output of the previous command.

## REFERENCES


[1] IT Infrastructure Library. ITIL service support, version 2.3. In Office of Government Commerce, 2000.

[2] Fabio Oliveira, et al. "Barricade: Defending systems against operator mistakes". In Proc. of Eurosys, 2010.

[3] 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.

[4] Venkat Madduri, et al., "Towards Mitigating Human Errors in IT Change Management Process". Proceedings of the ICSOC, 2010

[5] Alexander Keller, et al., "A Configuration Complexity Model and Its Application to a Change Management System ". IEEE Transactions on Network and Service Management, 2007.

[6] A. B. Brown, "Oops! Coping with Human Error in IT Systems," Queue., vol. 2, no. 8, Dec. 2004.

[7] D.Patterson et al., Recovery Oriented Computing (ROC): Motivation, Definition, Techniques and Case Studies, Computer Science Technical Report UCB//CSD-02-1175, U.C. Berkeley, 2002.

[8] PuTTY, http://www.putty.org/

[9] Larisa Shwartz, et al., "Quality of IT service delivery - Analysis and framework for human error prevention", SOCA 2010.

[10] AutoIt: http://www.autoitscript.com/site/autoit/ for windows; xdotool : http://www.semicomplete.com/projects/xdotool/ for linux.