RI08007 17 April 2008 Computer Science

IBM Research Report

Enabling Selective Automation of Human Decision-Making Using Rules as Preferences in a Service-industry Application

Biplav Srivastava and SivaKiran YellamRaju
IBM India Research Laboratory
4, Block - C, Institutional Area
Vasant Kunj, New Delhi - 110 070, India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted is accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T.J. Watson Research Center, Publications, P.O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

Enabling Selective Automation of Human Decision-Making Using Rules as Preferences in a Service-industry Application

Biplav Srivastava and **SivaKiran YellamRaju**
IBM India Research Laboratory
New Delhi and Bangalore, India.
Email: {sbiplav,syellamr}@in.ibm.com

Abstract

Rules, in their various forms, has been a prominent AI tool in automating the processing of real-world applications. In this paper, we explore how rules can be effectively used as preferences to expedite processing on behalf of humans in the labor-intensive service industry. The new challenges we handle stem from human behavior and audit/ quality considerations that impose design considerations for the rules and the rule system. We consider the domain of Information Technology (IT) change management which seeks to control and reduce the risk of any alteration made to an IT infrastructure in its hardware, software or attached network. We apply rules to selectively automate the decision-making to proceed with a proposed change (called approval) while respecting the considerations of multiple (human) role players. We piloted our system, called Approval Accelerator, in a live environment with about 150 servers for a month and obtained up to 93% reduction in effort and 44% reduction in end-to-end duration. We identify hither-to unknown issues about the impact of the rules on the application process and how rules can be better managed (e.g., storage, elicitation) to address them.

Introduction

Rules has been a prominent AI tool in automating the processing of real-world applications. The success of a rule-based solution depends on the nature of problem that is sought to be automated, how the rules are elicited and how the rule system is integrated into the overall application. Previous work here has almost completely focused in areas where the role and number of humans involved in the activity has been limited (e.g., manufacturing).

In this paper, we explore how rules can be effectively used to expedite processing in a human dominated service-industry application. Due to the presence of humans, new factors become important:

- *F1:* Humans make many decisions mechanically but are uncomfortable to any change that takes away their role in reviewing the decision later.
- *F2:* Human decision-makers may realize patterns in their decisions but have trouble expressing them as factual statements in some rule language.
- *F3:* Even when rules are adopted, for audit and quality reasons, the application should be usable indistinguishably from the case where no rules were present.

The factors need a careful analysis of how the process for using the application changes due to the adoption of rules and what design considerations it imposes for the rules and the rule system. An example of the latter is that even when someone has provided a rule and the rule is applicable, humans should still be provided reasonable opportunity to rebut the rule-based system's decisions (for whatever reason).

We consider the service domain of Information Technology (IT) Change Management(OGC 2006)[1] which seeks to control and reduce the risk of any alteration made to an IT infrastructure in its hardware, software or attached network. A rapidly increasing segment of the IT industry today deals with offering services to businesses to manage their IT infrastructure. It is common to see a large organization having its managed IT systems, e.g., servers or data centers, co-located with its operations but being managed from a remote location. Customers often demand that change management be followed for all remotely delivered services. The services span a variety of processes like performing patches, user account management, storage and backup, etc.

Approval is a key step (sub-process) in change management by which all stakeholders of the change give concurrence to the delivery team to perform the requested change. In spirit, approval should balance the need to keep *relevant* stakeholders abreast of a *particular* change that is going to affect them while let the change be performed at the earliest to meet the client's request. In practice, as we find, the delivery teams are conservative by taking approvals from more stakeholders than necessary and this not only over-burdens the approvers but also slows the delivery of service. In extreme cases, it can take more than 99% time in a change to get approval when the change can itself be performed in less than 1% of the total change time. For a delivery center that handles 10000s of changes per day, the opportunity to reduce delivery time and improve efficiency is immense.

We successfully transformed the key approval sub-process of CM using rule-based automation, implemented it across a broad range of IT services and piloted the tool,

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] See also http://en.wikipedia.org/wiki/Change_Management_(ITIL), http://www.itil-itsm-world.com/

called Approval Accelerator (AA), in a live delivery environment. AA is a cross-process framework for approval acceleration that uses rule-based preferences and data integration to remove the drudge work for an approver[2]. The rules serve as an explicit representation of an approvers' frequently-made, routine decisions and captures his approval preferences. Note that the approver may be aware of his preferences or it can be learnt from his historical performance. We anticipate approvers to be most comfortable providing preferences for low-risk decisions but they can still rebut system decisions based on those preferences if they so choose. Consequently, AA focuses the approvers' attention on high-risk and skill-based decisions. We piloted AA in a live delivery environment with about 150 servers for a month and applied to a majority of its changes. AA lead to up to 93% reduction in effort and 44% reduction in end-to-end duration for the changes. The encouraging results are being applied to more IBM delivery centers and helping in guiding new features in IBM's system management tools that support rule-based automation (Tivoli).

Our contributions are:

- We explore the issues of using rules in human-dominated applications. The key issues are how can the application prove that the right rules were used by the application in the right situation, when should the human approver's attention be sought in approval, escalation or revocation, and how can one still devise rules that are effective.
- We identified the hither-to unknown situation that if one does not limit the minimum time duration between when an approval is requested and the change is scheduled, a change for which a rule is applicable can be triggered and the change performed without giving the approver sufficient time to rebut system's decision on her behalf.
- We show how rules as preferences was successfully used for selectively automating approval in IT change management through a case study in a live environment.

In the remainder, we first give a background of rule processing and IT change management. We analyze the problem of seeking (and giving) approvals for changes in delivery centers and illustrate how AA seeks to address them. We then discuss our solution to using policy-based selective automation of human approvals and present the AA tool. Next we explain the pilot study in a live delivery environment and conclude.

Background

Rules as an Approach for Automation

A rule is a declarative specification of behavior that is desired from an agent. Rules have been used since the early days of AI to automate decision making (as expert systems[3]). They are also popularly called business rules today[4] or policies. They are an active field of research in IT system management(Kephart 2005; Verma 2001). There are even off-the-shelf tools like Tivoli[5] which support industry-strength rule-based system management.

There are many rule formats of which the simplest are If-Then-Else and Event-Condition-Action (ECA) rules. Here, a detailed list of events and/or conditions is recorded along with the appropriate prescribed actions. During runtime, a rule engine will verify the rule triggers and take the stipulated action. We use the ABLE rules system(Bigus *et al.* 2002).

Illustration of an IT Process

Table 1 shows the near-reality patch process managed by a remote service delivery team. Such a process description, or *process model*, consists of the activities involved at each step, the roles involved with the activities, the data items needed to process a step, the data generated and the tools used. The last column identifies if each process step is common or specific to a particular aspect of the IT environment.

The Patch Analysis Report or PAR is a document to issue security advisories that may affect systems that are owned or managed by the IT vendor. It is issued by the Security Team (ST) in Step 1 and sent to platform teams (i.e., Windows, AIX, etc.) of different accounts to be processed by their System Specialists (SS). The specialists (also called Change Builder, Change Implementer in ITIL(OGC 2006)), in Step 2, check the relevance of the PAR to the servers that are present in their account. The information needed to make this decision is the PAR notice and the server description. In the 3rd Step, a Change Management Request (CMR) is created in a tool like Remedy, ManageNow or CPMA by the Service Management (SM) representative (also called Change Manager in ITIL). The change is now sent to different groups for internal checking and scheduling approval (Step 4). The approval consists of determining a mutually acceptable change schedule and these teams determining if they have pre-requisite, co-requisite or post-requisite patches that need to be applied along with the proposed patch changes. In Step 5, the system specialist knows the final list of patches that will be applied. She downloads the patch from the patch site and tests it. In Step 6, the patch is uploaded to the server(s) where the patch will be applied. In Step 7, the change is built by the platform team and in Step 8, customer's approval is solicited by the SM representative. In Step 9, the change is built and in Step 10, it is proposed to be closed. Now, the SM team takes over in Step 11 and closes the change after verifying with the customer. Finally, in Step 12, the PAR is closed.

Now suppose we want to know what steps in this process can be cost-effectively streamlined through rule-based automation and upto what level. There are frameworks like Autoseek(Srivastava 2007) to help select steps which are good automation candidates based on their importance in the process, the types of their information processing, and the effort needed to elicit, write and validate the rules.

[2]Implementing data integration methods on a wide scale across many accounts with differing IT infrastructures is generally harder, costlier and more disruptive than incorporating a rule-based preference system integrated with CM tools. Hence, the focus of the AA prototype and pilot was mostly on rules.

[3]http://en.wikipedia.org/wiki/Expert_system

[4]http://en.wikipedia.org/wiki/Business_rules_engine

[5]http://www.ibm.com/software/tivoli/

Step No. and Name	**People Role**	**Action Type**	**Data Needed**	**Date Generated**	**Tool(s)/ Database(s) Used**	**Specific To?**
1. PAR Notification	ST	C	N/A	Email notification	Lotus Notes	Operating system
2. Patch Relevance	SS	I	Server Information, PAR No.	Applicable information		Operating system
3. Create CMR	SM SS	S	Server Information, PAR No.	CMR	ManageNow, CPMA, Remedy	Customer specific
4. Check for Patch Dependence & Schedule Approval	SM SS	S or C	PAR No.	Pre-requisites, co-requisites, post-requisites	Patch site (Microsoft, IBM ftp site)	Operating system
5. Download Patch and Test	SS	S or C	PAR No.	Patch download	Patch site (Microsoft, IBM ftp site)	Operating system
6. Upload patch package	SS	S	PAR No.	Patch package		Operating system
7. Build Change	SS	C	Technical details of patch		ManageNow/ CPMA	Change and Customer
8. Get Customer Approval	SM	S or C			ManageNow/ CPMA	Customer
9. Implement change	SS	S	Patch package		Terminal Services	Customer
10. Propose to Close	SS	S	Implementation successful?	Server Hotfix update	ManageNow/ CPMA	Customer
11. Close Change	SM	I	Verification with customer		ManageNow/ CPMA	
12. Close PAR	ST	I	CMR		Patch Notification System	

Table 1: A patch process at an example service delivery center.

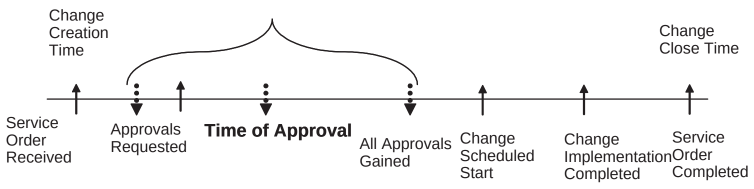

Figure 1: *Stages of a change and the time line.*

Problem and Challenges

Approval in Change Management

In a typical CM process (see Figure 1), a change request (also called a service order) is received by a Service Coordinator and assigned to a Change Implementer. Approval is a key step (or sub-process) of CM by which all stakeholders to a change give concurrence to the change. No change can go ahead without all approvals in place. The Change Implementer will initiate requests for approval specifying the reason for change, technical details and when it is planned, and wait till all the approvals are obtained. In practice, the approvers are divided into a small primary group whose approval must be taken even in an emergency, and a larger secondary group whose approval can be delayed during emergency but must be still taken after a change implementation. Regardless of how the approvals were obtained, all approvals have to be documented for audit reasons. After getting approvals, the change is started at scheduled time, implemented and completed.

We were looking at the problem of how to improve the time and effort taken to perform changes in delivery centers. It was anecdotally known that acquiring approval was problematic in many processes and accounts. We set out to systematically understand the problem through profiling of sample processes and their data analysis, and interviewing a wide set of practioners. We had access to many types of processes (e.g., software installation, account-id creation, hardware installation, patching) in different delivery accounts at IBM. We also could interview Service Coordinators and Change Implementers from a wide cross-section of accounts with differing characteristics - e.g., internal v/s commercial, different regions, using different tools.

Profiling

The aim of profiling was to know the accurate time taken by humans to perform the different steps of a process and also estimate periods when the request was waiting between the steps. The information would not only let us understand the bottlenecks in a process but also help us evaluate how good any solution that claims to remove bottlenecks is (or does it merely transfer them to some other step?). We built a portal-based profiling tool that showed only relevant process steps to each role player. The practioner had to click a button to indicate that the corresponding step is initiated or completed. The time stamps of the clicks for all changes is stored in a central database and later queried appropriately for further analysis.

We took the following approach to understand the approval problem and AA solution using profiling.

1. Validate an account based on changes raised and types of processes covered in a period.
2. Profile the selected existing processes in the account
3. Insert AA (discussed later) into the account for approval in the selected processes
4. Compare (2) and (3) and quantify the results

Observations

While the benefit of profiling is that we get accurate measurements, the activity is time consuming. We tried to increase the coverage of our analysis by orders of more processes and accounts by conducting interviews of delivery practitioners. We found that the two complementary forms of analysis were consistent in highlighting approval as a bottleneck and the possible reasons.

We found that:

- Gaining approval from internal teams could take anywhere from 10% - 80% time of the whole time to serve a change request. In pathological cases, more than 99% of service time could be wasted on trying to get approvals.

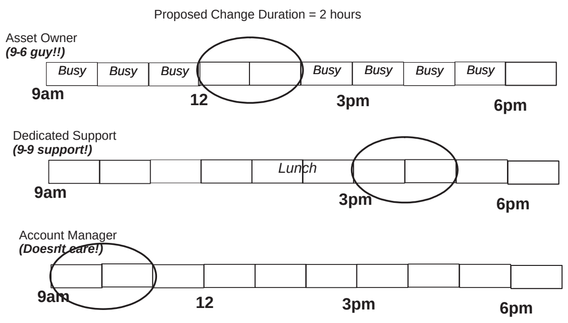

Figure 2: *The schedules of approvers in the example change and their next free times (circled). The change can be done at 6pm.*

- Many approval requests were of little concern to the approvers. On the other hand, approvers would have to make efforts to find changes with major impacts.
- The main reasons for delayed approval were:
  - Data fields in the Change Record would be incorrect or missing. Example: change details, approvers, category, etc.
  - There would be conflict in the proposed schedule or resource used for change
  - An approver's technical or business concerns about the change was not addressed or documented in the Change Record

Illustration of AA Usage

Before we present the solution, we illustrate how AA can accelerate approval using policies and the intricacies involved. There are many aspects that needs to be taken while approving a change. As a simplified example, we now focus on a single aspect of approval – the time at which the change should be done. Suppose the change is needed as per the patch process presented earlier and in a given account, there are three approvers: the administrative owner/ user of the server (asset), the dedicated on-site support personnel and the account manager (see Figure 2). Let the owner be a person using the server from 9am-noon and 2pm-6pm. Let the on-site support window be from 9am-9pm but a lunch break is permitted from 1-3pm. The account manager does not care when a change is done as long as all steps are followed.

In this simplistic scenario, in the current practice, the Change Manager will seek approval from each of the three approvers with a proposed time, and even if one of them refuse, he has to find a new time slot and then seek approvals from *all* of the approvers again. The unsuccessful approval cycles depend on the order in which approvers are approached as is clear in Figure 2.

However, it is usually the case that all approvers are not equally sensitive about the time of change. They may be able to express them as rules, as illustrated in Figure 3, or we can detect such patterns from their history of giving approvals. It is important to note that past history is not a predictor of what an approver may do on a particular change, but only a good indicator. So, the approver should retain the flexibility to revoke any decision made with a subscribed rule for a particular change.

P1: *I give pre-approval for any security-related mandatory change that is after 6pm*
If (changeType == "IGA-Mandatory" and changeTime $\geq$ 1800)
result = "yes"
P2: *If there is a low-risk suggested change not requiring a reboot, please make the change without bothering me. But please document it.*
If (changeType == "IGA-LowRisk" and needsReboot == "no")
result = "yes"
P3: *If a change is initiated by me, please involve me at the time of the change to ensure that it is done as I want it*
If (changeType == "UserInitiated")
result = "no"

Figure 3: Choice of example policies for the asset owner.

Suppose the asset owner subscribes to rule P1. AA would record such a preference and when the Change Manager will seek approvals, the system will use the rule in proposing the likely time of change. The asset owner will still be notified that a change is being made but now, the notification will not be a request for approval but an information. It will also include a provision to revoke the decision (here, i.e., the time of change) resulting from the subscribed rule.

As a result of AA, here rules become a concrete representation of an approver's scheduling constraints where none existed before. Although incomplete, the constraints allow the Change Manager to request for approvals with more relevant information. In the example, a simple schedule comparator can detect that 6pm is the first common time that is free for all approvers and should be proposed.

The benefit of AA will be felt if unsuccessful approval requests and subsequent unsuccessful cycles are reduced. To do so, we should get more primary approvers to subscribe to meaningful rules so that there is little or no revocation of the decision later.

Solution Approach

IT services are delivered using a combination of process, skilled people and technology, and these are also the parameters which can be varied to address any delivery problems. We wanted to tackle approval delays using rule technology in a manner that it can be applied to existing processes and available personnel.

In AA, we take a two-pronged approach as shown in Figure 4. We pre-fill most of CR fields with all relevant and available process details before the time approvers need them *via data integration from multiple sources*. We allow

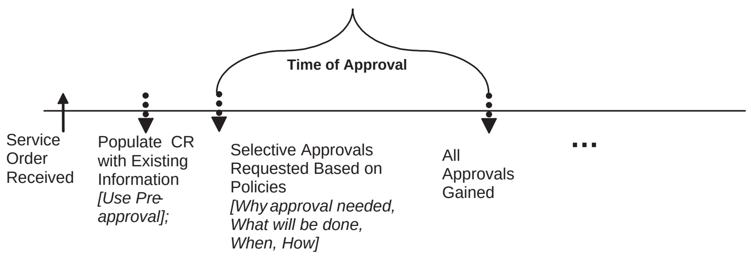
Time of Approval
Service Order Received
Populate CR with Existing Information
[Use Pre-approval];
Selective Approvals Requested Based on Policies
[Why approval needed, What will be done, When, How]
All Approvals Gained
...

Figure 4: *Modified stages of change with AA.*

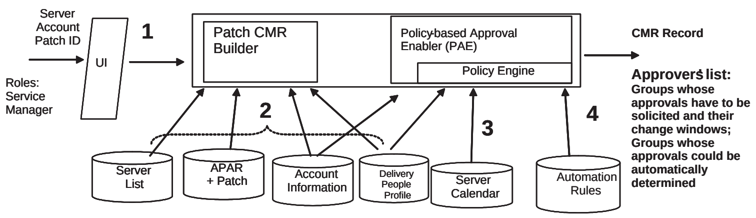

Figure 5: *AA architecture; data flow is marked.*

approvers to subscribe to rules that reflect their concerns about the change and use the rules to process the approval. Note that implementing data integration methods on a wide scale across many accounts with differing IT infrastructures is generally harder, costlier and more disruptive than incorporating a rule-based preference system integrated with CM tools. Hence, the focus of the AA prototype and pilot was mostly on effectiveness of rules as approver preferences even though data integration is the preferred mechanism to achieve it.

Data Integration

For pre-filling a CR related to patches, in AA, we need to integrate data about Servers (e.g., server names, IP addresses), Account Information (e.g., SLA), Delivery Personnel (e.g., contact details) and the process information (e.g., Patch number and details for patches). There are two novel features in the AA system. To enable rule-based automation, we *introduced rules as a data source in the system*. Moreover, to simplify change scheduling which is a major source of conflict, we introduced the notion of having a calendar for each IT resource and included its schedule as a data source.

Now, given a change request (see Figure 5), the data from the repositories are retrieved and aggregated, consistency of schedules is checked and rules evaluated to determine the final CR. With this, only approvers whose decisions could not be determined for the raised change are bothered, and for the rest, the system informs about processed decisions.

Specification and Usage of rules

With rules, we want to capture any patterns or preferences, whether desirable or undesirable, that the approvers can give about the changes. The rules can encode resource conflicts, technical or business concerns about a change like:

- the approver's (or his applications') relationship with the proposed change
- preference about when to do change in his environment
- preferred mode of communication about the change

We earlier noted that humans should be allowed to rebut any decision taken on their behalf using rules. This raises the question about when can an automated decision be activated. An approver's decision for a change is automatically determinable if:

1. She has subscribed to a rule for that type of change, AND
2. A rule engine determines that that rule applies for the new change being proposed, AND
3. The approver has not refuted the proposed change until the scheduled start of change

```
01. Process Change Request
02.    ServerList ⟵ List of Servers in the request
03.    ApproverList ⟵ List of Distinct Technical Owners (Approvers)
04. Schedule the Change Based on resource calendars & ApproverList
05. For each Approver 'A' in ApproverList
06.       If A.subscribedRules is NOT null then
             ;; Approver has subscribed to a rule
07.          Process the change for A.subscribedrules
08.          If result is yes for some rule 'P' then
09.             Result ⟵ "YES(P)"
10.             Generate Email and Send but approver's response
                is not required to make the change
11.          Else
12.             Result ⟵ N/A (No pre approval)
13.             Send email asking for approval
14.       Else
             ;; Attempt to learn by proposing a default rule
15.          Process the change using DefaultRules
16.          If result is yes for some rule 'P' then
17.             Result ⟵ "Default-YES(P)"
18.             Send email asking for approval mentioning the application of
                the default rule as FYI.
19.             (Analyze the user's response over a period of time and
                make suggestions
20.          Else
             ;; The situation without AA
21.             Result ⟵ N/A (No pre approval)
22.             Send email asking for approval
```

Figure 6: The pseudo-code of AA.

Figure 6 shows how rules can be used to automate approvals during CM. There are two main cases: an approver has either subscribed to a set of rules or not. In the former case, the current change record is evaluated to see if any of the subscribed rule matches. If yes, the system already has a pre-approval from this approval and an appropriate email ***informing*** the decision taken is sent as a notification. Otherwise, none of the subscribed rules match, and the approver has to be conventionally requested for his/her approval. In the latter case, we can attempt to learn rules. We define a default set of rules unknown to the approvers and evaluate a change based on them. The result is a suggestion and sent to the approver in the initial approval request e.g., as a scheduled start time. If the approver accepts the suggestion, we have been able to learn one instance of the user's preference. These instances can be aggregated via different learning schemes to derive candidate rules. Note that we are merely trying to intelligently ***request*** for approval but have to ***wait*** until the approver gives her decision. The current process, which is what happens if no default rules are used, ***requests*** for approvals blindly and ***waits***.

Pilot Evaluation and Experience

Our objective was to evaluate the impact of using the AA approach in a live environment and understand the likely issues. We present the pilot results of AA in an account, the estimation of the likely savings for delivery centers and the overall learnings.

Piloting AA in an Account and Measuring the Impact

We looked at an account with 150 servers+ for a month. The AA system supported IT process covering 95% of the changes in that account – patching, hardware and software changes and user management. There were an average of 3 approvers for any change in these processes of which 1 was the primary approver. For the servers in the account, there were about 51 unique primary approvers. Our focus with rules was the primary group.

We wanted to evaluate two types of metrics: efficiency, whereby one measures the effort (e.g., time, cost) spent in giving approvals, and effectiveness, whereby one measures the performance (e.g., end-to-end duration) as seen by a customer. Efficiency is affected by the approver who has to process (gets impacted) the most to give the approval. Usually, he is in the primary approver group. Effectiveness is affected by the approver who is tardiest in giving approvals. In the pilot, though we can directly impact efficiency of primary approvers, we may not be able to control effectiveness since it could be caused from secondary approvers as well.

We used the profiling tool discussed earlier to collect information about the time taken to approve a change by approvers and the end-to-end time of the change, both before and after the AA system. We had a list of 25 rules that the primary approvers could subscribe to. The rules were acquired from our interviews of approvers and delivery personnel in the pilot account and others as well. We explained the rules available to all the primary approvers but only 10% of them subscribed to some rule. We used 4 default rules to make suggestions when no user role was available (either applicable or subscribed).

We found from the pilot that there was:

- Efficiency - effort in giving approval
  - There was 80% savings across all processes measured on a time basis
  - When savings are measured based on the salary of individual approver groups, the saving show as $\geq$ 93% across all processes.
- Effectiveness: end-to-end duration
  - The savings was $\succ$ 44% for patching
  - The savings was $\leq$ 5% for other processes

During the pilot period, patching was the dominant change in the account. No approver who had subscribed to a rule refuted its decision during the pilot. Surprisingly, we also found that while default rules were used in suggestions to primary approvers in 85% of the changes, there was no refutation of it. While we do expect rules to be refuted by approvers and more so when they have not subscribed to them, the pilot experience does indicate that there are good rules which broadly capture approvers' preferences.

The results were very encouraging. Recall that the efficiency is impacted by rules while the effectiveness was not in control of the primary approvers. We also did a user survey of the approvers to gauge their experience. Results from surveys show that they:

- want help to reduce time spent on approvals
- can provide patterns of desirable changes (rules) and AA captured many of them
- did not feel that mere integration of data will improve their approval productivity

Estimating impact of AA in a delivery center

With the pilot, we had results from a controlled experiment. We also estimated the scope of savings possible for a large delivery center. In doing so, we had to consider that the typical number of approvers for a change can vary anywhere from 3 to 10 of which the primary approvers can be from 1-4. Finding patterns/ obtaining rules for primary approvers is time-consuming and difficult but if obtained, the rules can be very effective in reducing the efforts of rest of the approvers. Finding rules for secondary approvers is easier but lesser effective in increasing savings since they usually try to accommodate decisions of primary approvers.

We estimated that for large delivery centers, where there is a large diversity of accounts and IT processes, obtaining up to 50% savings in effort (measured on time basis) and up to 20% savings in end-to-end duration from preferences captured as rules is feasible. We also note that while data integration emerged as an issue in our problem analysis, it is not factored in our estimates. Hence, the possible savings could be even higher if a standardized approach towards data integration across accounts is taken.

Learnings from AA

The implications of using rules to supplement human decision-making are:

1. The rules subscribed by each approver at *any given time* should be known.
2. The life cycle of each rule available in the system has to be tracked and managed until it is obsoleted so that provenece of rules is known for audit.
3. The minimum duration between when the approval is raised and when the change is scheduled has to be prominently highlighted.

The last implications is very important if rules have to be used for automating human decision making and is largely unknown. Recall from previous section that the third condition for determining approver's concurrence is that there is no rebuttal during this period. But if the period is arbitrarily short, the approvers would not be able to rebut. How do we put a minimum limit and highlight it?

The answer is Service Level Agreements (SLAs). SLAs are contractual documents recording the performance criteria promised by a IT services delivery organization and its customer, and then closely monitored by all. With AA, this duration has to be brought into SLAs for all processes wherever the customer is among the primary approvers.

Conclusion

We looked at how rules can be effectively used to expedite processing in a human dominated service-industry application. The key principles in AA are: allows humans the opportunity to rebut decisions resulting from rules that they may have subscribed, exercise default rules to learn implicit rules that the humans can later ratify, and incorporate rules as a data source that is processed while data is being integrated in the change system. We did a controlled pilot and got encouraging results that we estimate can be generalized.

The learnings are being applied to more IBM delivery centers and helping in guiding new features in IBM's system management tools that support rule-based automation.

We believe the AA approach can serve as a guide for others trying to simplify and automate human decision making using preferences in other service-industry applications as well.

References

Bigus, J.; Schlosnagle, D.; J. Pilgrim, J.; Mills, W.; and Diao, Y. 2002. Able: A toolkit for building multiagent autonomic systems. In *IBM Sys. Jour., Vol. 41, No. 3*.

Kephart, J. O. 2005. Research challenges of autonomic computing. In *ICSE '05: Proc. 27th Intl. Conf. Software Engg.*, 15–22. New York, NY, USA: ACM Press.

OGC. 2006. IT infrastructure library (ITIL). In *http://www.itil.co.uk/*.

Srivastava, B. 2007. Autoseek: A method to identify candidate automation steps in IT change management. In *Proc. Intelligent Management (IM-2007), Munich, Germany*.

Verma, D. C. 2001. Policy-based networking: Architecture and algorithms. In *New Riders Pub., ISBN: 1-57870-226-7*.