RI 06008 1 September 2006 Computer Science

IBM Research Report

AutoSeek: A Method to Identify Candidate Automation Steps in a Service Delivery Process

Biplav Srivastava
IBM Research Division
IBM India Research Lab
Block I, I.I.T. Campus, Hauz Khas
New Delhi - 110016. India.

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

AutoSeek: A Method to Identify Candidate Automation Steps in a Service Delivery Process

Biplav Srivastava
IBM India Research Laboratory
Block 1, IIT Delhi, Hauz Khas,
New Delhi 110016, India.
Email: sbiplav@in.ibm.com

Abstract

A variety of processes for managing IT systems are being remotely serviced today. There is a growing realization that these services, which were hither-to labor-centric, should adopt more automation to improve quality and reduce cost. However, it is not clear how this can be done. We present a method to systematically analyze a process to find steps which can be automated in a cost-effective manner with the appropriate level of automation. In doing so, we balance the need for efficiency from automation while considering the cost of implementation and maintenance to perform it. The method, called AutoSeek, has been applied to different types of delivery processes and has been found effective as a broad framework towards systematically making process improvements. As a concrete example, we will also discuss how acceleration opportunities for approvals during change management can be understood with AutoSeek and undertaken.

1 Introduction

Information Technology (IT) infrastructure services are increasingly being delivered from remote locations. 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. IT Change Management[1][1] seeks to control and reduce the risk of any alteration made to an IT infrastructure in its hardware, software or attached network. 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.

There is a growing realization that these services, which were hither-to labor-centric, should adopt more automation to improve quality and reduce cost. In order to optimize the delivery of services, any process owner would want to consider automating some of the process steps. However, the key issue in doing so is that one needs to strike a balance between the money gained through savings by automation with the effort spent to ensure that the potential business risk of the automated step (due to errors, unexpected behavior) is low. Today, it is not clear that given a process, what steps in it should be automated and how? To our knowledge, no method currently exists to address this problem.

We present a method which can be applied to any service delivery process to identify what steps should be automated and up to what level. The solution characterizes the complexity of activity at each of the process step and then maps it to an increasing level of automation choices. As one chooses more complex automation, one needs to provide more control information and needs to spend more effort in validating that the automated steps behaved correctly. Finally, the two conflicting factors - of savings due to automation and the effort needed to implement the automated step and validate step execution - are compared to decide if the process step should be automated.

The advantage of using AutoSeek in a service delivery environment are many. It helps in process improvement so that:

- Time reduces due to better data availability and automation
- Cost reduces due to fewer resources
- More focus is put on high-risk changes; today, all

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

changes get equal (low or high) attention

- Processes are better documented

We illustrate the solution in the context of a near-real[2] patching process. AutoSeek has been applied to many types of delivery processes and has been found effective as a broad framework towards systematically making process improvements. As a concrete example, we will also discuss how acceleration opportunities for approvals during change management can be understood with AutoSeek and undertaken.

Outline: We begin by first giving the example scenario of applying operating systems patches, or *patching*, at a service delivery center. Then, we describe relevant background material from Autonomic Computing[2], specifically policies, that forms the basis for building a methodology for automation. Next, we present the AutoSeek methodology and illustrate it in the context of the patching example. Then we apply AutoSeek to look specifically at how getting approval for patching related changes can be improved. We end with a discussion of the advantages of the proposed approach, an implementation and related work.

2 Example

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 process can be modeled in many ways. For example, one can use modeling tools like WebSphere Business Integrator (WBI) Modeler, drawing tools like Visio, or a simple table representation. We choose the latter format here. We now focus on the type of activity processing happening at each step.

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[3]), 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 can be cost-effectively streamlined through automation in this process and with upto what level. The extreme decisions are to not consider automation at all or to try to automate all steps. AutoSeek is a framework to help answer these questions.

3 Background: Towards Developing a Methodology

We give background of different issues that one needs to consider before automating parts of a process. We descibe how the activities of a step can be characterized based on the nature of information processing happening at it and the kind of automation choices available via policies. Then we propose a modest approach for automation by mapping the types of activities to the suggested type of automation policies that can be used for them without significant effort on writing or validating them. For these candidate steps for automation, we further argue only those steps be considered for which the expected benefits balanced out the cost required to modify the step, e.g., automate access to the input data needed at the step, write and verify policies.

3.1 Characterizing the Activities at a Process Step

A service delivery process consists of steps describing activities. Figure 1 shows what happens in a

[2]Due to confidentiality reasons, we only expose sufficient information from real cases that is necessary for discussion.

[3]Information Technology Infrastructure Library, see http://www.itil.co.uk/.

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.

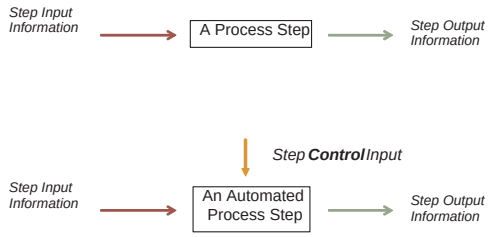


Figure 1. *Information Processing Occurring at a Process Step (above) and when it is automated (below).*

process step. A process step consumes information about input, does processing and produces output. The information about output records that the process has taken place. The types of processing occurring at a step can be classified as:

1. Checking of information (I-type): Here, the activity involves checking correctness of input, testing its validity, entitlement or applicability.

2. Making simple decisions and acting (S-type). Here, the activity is to check data value that leads to statically enumerable set of decisions (e.g., if a resource is unassigned, then it can be applied to a proposed task).

3. Making complex decisions and acting (C-type). Here, the activity is to check data values and apply the knowledge of sub-system behavior collectively to make decisions (e.g., if log file is 90% full, the database can be unstable. To improve performance, request for database re-configuration,)

Now, before automating a step, one needs to figure out the desirable role of a person performing the task. Is the person a generalist or a specialist? In a service delivery environment, the workforce is organized along specialist teams like Windows, Database - DB2, SAP, etc. In such a scenario, the performers are specialists. We argue that in such a setup with sub-system aligned teams, the performers should be focused on making (complex) decisions which uses the knowledge about the sub-system much more effectively. The information check and simple decisions are distractions to them and can be screened away.

3.2 The Spectrum of Automation Choices

Recall that a process step consumes information about input, does processing and produces output. To automate any process, control inputs are needed to let the process step know what behavior is expected (Figure 1 below). A correct automated process will use the control and input information, and produce the same output information as that produced by a non-automated process. Example of control information are:

1. Condition: Condition that is being checked at that step. The automated activity performer, who knows what to do in the condition, will simply check the condition and execute the activity accordingly.

2. Prescriptive policies[4]: Conditions that are important and what actions to take in different conditions. The automated activity performer knows what actions to take in each condition and will act accordingly based on the prevailing condition.

3. Declarative policies: Goal that is desired for a particular set of inputs. The automated activity

[4]The terms policies and rules are interchangeably used in the paper.

I_C Type	Effort to Specify and Validate I_C	Effort to Reason and Execute Step with I_C	Effort to Verify Process Step Behavior	Automation Achieved
Condition	1	1	1	Least
Prescriptive policies	2	2 / 3	1	Moderate
Declarative policies	3	4 / 5	2	Most
Utility policies	4	5	2 / 3	Maximum

Table 2. A qualitative comparison of the trade-off in using different types of control information (policies).

performer needs to find the course of actions that achieves the goals and execute them.

4. Utility policies: Utility of what goals are important in different conditions. The automated activity performer has to select the goals of the specific condition based on utility, decide about the course of actions that will achieve the goals, and execute them.

The Event-Condition-Action (ECA) rules[3] are an example of prescriptive policies. The area of planning in AI[4] deals with how to generate a course of actions to achieve goals. They can handle both declarative as well as utility-based policies[5]. An example of a system which can support many types of policies is ABLE[6]

Table 2 shows the trade-off in using different types of control information (policies). Our intention is to highlight that as we shift towards higher levels of automation, there is savings in performer effort but now more effort is needed to ensure that the control information is valid and that the automated behavior was correct. There are specific computational complexity results for different variants of the given types of control information, e.g. [7, 8, 9]. Since complexity results is not the focus of the paper, we give a qualitative sense of the hardness by following the following convention: 1 for constant time, 2 for polynomial, 3 for NP-Hard, 4 for PSPACE and EXPSPACE-Hard, and 5 for complexity beyond it. The last column rates the level of automation achieved if a complex activity is automated by a given type of policy.

3.3 The Mapping Between Step Processing Complexity and Automation Choice

Given the complexity of the steps and the spectrum of automation choices, one gets an outline of a possible strategy to automate a process.

1. We can fully automate information checking steps through conditions

2. We can simplify and partially automate simple decision making steps through prescriptive policies.

3. We can focus the teams to complex decisions and explore declarative and utility policies over time.

In complex decisions, it is possible to automate using AI planning[4, 6] techniques. However, such techniques work best when the step-specific activity information are completely and formally modeled[10]. There can be significant cost of doing it in a service delivery environment where there are multiple sources of variations from one customer to another. However, even without the declarative and utility policies, we can provide more context-specific information to the performer so that she can act fast. There is a role of tools to provide specialized information using the context available, e.g., Impact analysis tool.

3.4 Not Everything Automatable is Worth Automating

After identifying the automatable steps in a process (e.g., information checking steps), it is possible that the step is not cost-effective to automate. To see why this may happen, consider the trade-off between effort needed to automate and the potential gain (See Figure 2). To automate the step, we need to automate the access of step input as well as account for the effort needed in writing the control information (policies). The cost for writing adaptors to access the step inputs can sometimes be very high. Also, if the policies fail to execute because of an unforseen reason, e.g., runtime failure of the infrastructure or system bugs, the cost to by-pass and switch to a failover mode of operation has to be also considered.

Against these costs, the potential gain of automation increases if the process step is frequent and significant time is needed to manually perform the task correctly. If a process step does not have these characteristics and/ or the cost of writing adaptors is high, there may not be sufficient benefit over the cost need to do automation.

We consider these factors to develop the AutoSeek methodology.

4 AutoSeek

Figure 3 presents the AutoSeek method. In the first step, we read in the process model of the service delivery process that we want to analyze for automation.

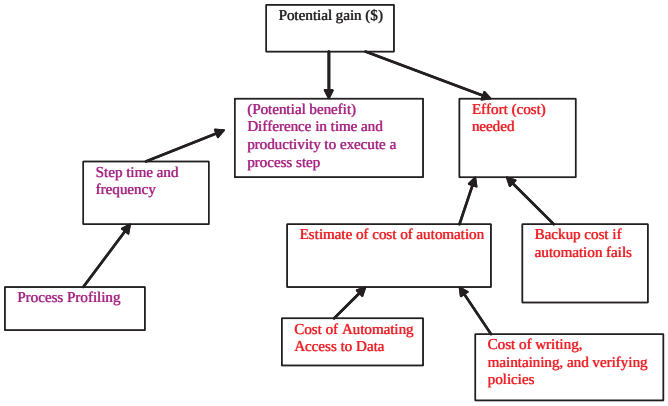


Figure 2. *Factors affecting the benefit and cost of automation of a process step.*

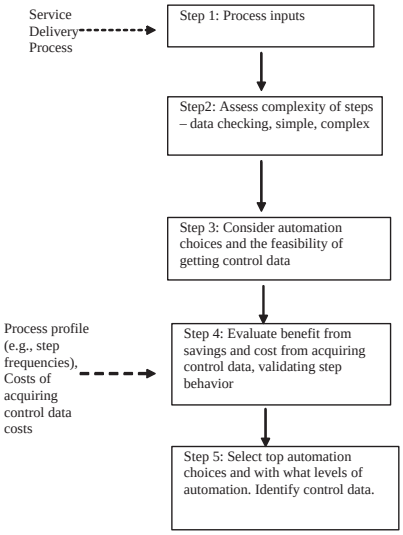


Figure 3. *AutoSeek Methodology.*

Next, we analyze the nature of activities. In the previous section, we categorized the complexity of doing processing (using information and taking decisions) at any step of a process. In the third step, we decide on a level of automation. In the previous section, we also categorized the available choices for automating the process steps. Then, we do the benefit v/s cost analysis of the proposed automation to decide which steps are feasible. Finally, AutoSeek outputs the recommended automatable steps with the corresponding type of automation.

We now apply the AutoSeek methodology in the example patching process to see how it works. We note that AutoSeek is not specific to any specific service delivery process. In fact, we have applied it to a number of processes and found it to be effective.

4.1 Applying the Methodology on the Patch Example

In Table 1, the characterization of the steps based on step complexity is also shown. This corresponds to Step 2 of AutoSeek. We note that the numbers of steps of the different type are:

- Checking information (I-type) in which the correctness, validity, entitlement or applicability of the input is checked. 3 of the 12 steps (25%) are of this type in the example process.
- Make simple decision (S-type) where input along with prescribed actions in different conditions leads to action determination. 4 to 7 of the 12 steps (33-58%) are of this type in the example process.
- Make complex decisions (C-type) where input along with sub-system knowledge leads to action determination. 2 to 5 of the 12 steps (17-42%) are of this type in the example process.

In the above, the number of steps was used to estimate the distribution of the activities in the process. Another way to estimate the distribution would be to measure the time taken at each activity. The time would vary based on the process step type characteristic and would be exactly known only from a careful profiling of the process in a live environment. For now, let us choose the number of steps for estimation.

Now, we apply Step 3 of AutoSeek. A modest approach for automation is to only consider information checking steps for automation, simplify and partially automate simple decision steps using prescriptive policies, and focus the teams to make complex decisions manually. The addressable scope for improvement by automation, in our example, is minimum 25% due to information checking steps and the maximum is 83% through checking and simple actions. A conservative estimate for improvement is about 50%.

However, automation of steps also means that the access of inputs needed for processing information checking and simple decision steps has to be automated, if not already done. In Table 3, the process is shown after applying Step 3 of AutoSeek. The last column shows the ease of accessing data for the process steps.

Now, in Step 4, we need to consider the benefit v/s cost trade-off for automation. The column on potential benefit assesses the frequency and extent of the

Step No. and Name	Action Type	Data Needed	Date Generated	Tool(s)/ Database(s) Used	How Easy to Get Data
1. PAR Notification	C	N/A	Email notification	Lotus Notes	
2. Patch Relevance	I	Server Information, PAR No.	Applicable information		Low (account information)
3. Create CMR	S	Server Information, PAR No.	CMR	ManageNow/ CPMA	Low (account information)
4. Check for Patch Dependence & Schedule Approval	S or C	PAR No.	Pre-requisites, co-requisites, post-requisites	Patch site	High depending on application group
5. Download Patch and Test	S or C	PAR No.	Patch download	Patch site	High if testing of patch has to be done
6. Upload patch package	S	PAR No.	Patch package		Medium
7. Build Change	C	Technical details of patch		ManageNow/ CPMA	High as Platform knowledge is needed
8. Get Customer Approval	S or C			ManageNow/ CPMA	Medium or High (depends on account)
9. Implement change	S	Patch package		Terminal Services	Low (package and account information)
10. Propose to Close	S	Implementation successful?	Server Hotfix update	ManageNow/ CPMA	Medium
11. Close Change	I	Verification with customer		ManageNow/ CPMA	Low (account information)
12. Close PAR	I	CMR		Patch Notification System	Low (account and patch information)

Table 3. After applying AutoSeek Step 3 in the example.

step while the column on cost assesses the cost of automation. One can be as elaborate as needed to do an informed assessment of benefits and costs using Figure 2. Although detailed quantitative analysis is accurate, it takes time and effort. Moreover, in many cases, the experience of the delivery team can be used to get an initial qualitative scale for assessment and this can be refined with detailed quantitative analysis if we indeed decide to implement the automatable steps. We use a qualitative scale here. Based on the assessment, an initial list of candidate process steps for automation is arrived at. In Table 4, we show the result of Step 4 for the example process. The last column has a list of automation candidates.

The output of AutoSeek Step 4 was independent of any specific delivery environment – e.g., account, tools, region-specific compliance practices. In Step 5, we revisit the benefit v/s cost issue but with respect to a specific delivery environment. As a result, the candidate set for automation gets further refined.

Let us assume that the patching process is for an account in which the customer wants the servers to be aggressively patched against flaws whenever there is a notification from the platform vendor. They want all such changes to happen between midnight and 6am, their local time. Thus, Step 8 becomes Simple. Further more, they do not want to invest in a separate test environment to evaluate every patch before installation. Hence, the process steps 4 and 5 in Table 4 for this delivery environment becomes Simple. Also, the exact cases of what decisions to take become enumerable. Table 5 shows the final output after AutoSeek Step 5. It differs from Table 4 in its recommendation for process steps 5 and 8.

We can now measure the degree to which automation will be achieved with AutoSeek's recommendation. We measure the percentage of automation (A%) with Equation 1. We find that a total of 8 steps of the possible 12 steps in the patching process could be cost-effectively automated leading to a 67% realization of automation. We also measure the gain due to automation as a percentage (AG%) with Equation 2. The expression can be suitably modified to support different types of scaling for benefits and costs. With our implemented tool, as will be discussed later, the gain due to proposed automation in the example process is estimated as 45% (see Figure 5).

$$A\% = \frac{\sum_{S_j:true}}{\sum_{S_j}} * 100 \quad (1)$$

$$AG\% = \frac{\sum_{S_j:true}(S_j^{Benefit} - S_j^{Cost})}{\sum_{S_j} Benefit} * 100 \quad (2)$$

5 Using AutoSeek for Process Improvement

We now look at the utility of AutoSeek as a broad framework towards systematically making process improvements. Specifically, we see how inter-group coordination can be improved in our example patching process. Recall from Table 5 that the process Step 4 is about getting patch dependence information and involves knowing the pre-requisite, co-requisite and post-requisite patches, if any, of the proposed patch. Moreover, the groups can give their approval for scheduling.

In Figure 4, this step is shown as a detailed subprocess. Based on the type of application(s) which are running on the concerned server, requests are made to the corresponding groups for their feedback. So, if a

Step No. and Name	Action Type	Potential Benefit	Potential Cost	How Easy to Get Data	Candidate
1. PAR Notification	C	All Security Events	Too Risky	Low	No
2. Patch Relevance	I	All PAR	Little cost	Low	Yes
3. Create CMR	S	All Relevant PAR (R-PAR)	Little cost	Low	Yes
4. Check for Patch Dependence & Schedule Approval	S or C	All R-PAR	Information based in most cases	High	No
5. Download Patch and Test	S or C	All R-PAR	Testing is needed for compliance	High	No
6. Upload patch package	S	Drudge work for all R-PAR	Information available	Medium	Yes
7. Build Change	C	All R-PAR	Infeasible - Platform knowledge needed	High	No
8. Get Customer Approval	S or C	All R-PAR	Risky for adoption	Medium or High	No
9. Implement change	S	All Approved R-PAR	All information is in place	Low	Yes
10. Propose to Close	S	All R-PAR	Too risky for adoption	Medium	No
11. Close Change	I	All R-PAR	All Information is in place	Low	Yes
12. Close PAR	I	All PAR	All Information is in place	Low	Yes

Table 4. After applying AutoSeek Step 4 in the example.

Step No. and Name	Action Type	Potential Benefit	Potential Cost	How Easy to Get Data	Candidate
1. PAR Notification	C	All Security Events	Too Risky	Low	No
2. Patch Relevance	I	All PAR	Little cost	Low	Yes
3. Create CMR	S	All Relevant PAR (R-PAR)	Little cost	Low	Yes
4. Check for Patch Dependence & Schedule Approval	C	All R-PAR	Information based in most cases	High	No
5. Download Patch and Test	S	All R-PAR	Testing is needed for compliance	High	**Yes**
6. Upload patch package	S	Drudge work for all R-PAR	Information available	Medium	Yes
7. Build Change	C	All R-PAR	Infeasible - Platform knowledge needed	High	No
8. Get Customer Approval	S	All R-PAR	Risky for adoption	Medium or High	**Yes**
9. Implement change	S	All Approved R-PAR	All information is in place	Low	Yes
10. Propose to Close	S	All R-PAR	Too risky for adoption	Medium	No
11. Close Change	I	All R-PAR	All Information is in place	Low	Yes
12. Close PAR	I	All PAR	All Information is in place	Low	Yes

Table 5. After applying AutoSeek Step 5 in the example – for a specific account.

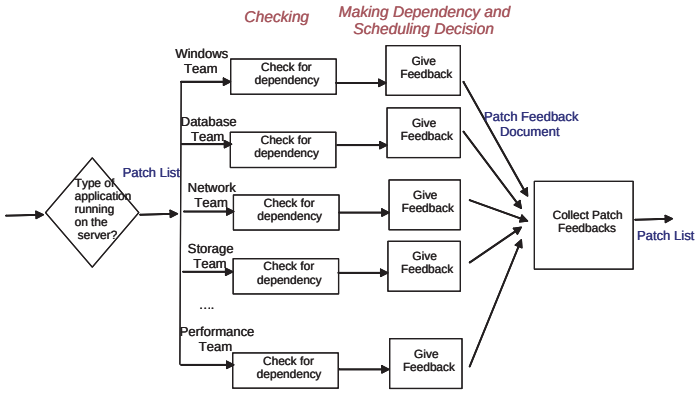


Figure 4. *The patch dependence step of the example patching process seen as a sub-process.*

machine is running Windows and has IBM DB2 database, the concerned teams would be atleast Windows, Database and Performance. The teams check the proposed patches for any interactions it may have with their concerns on that server. So, if the Database team finds that the proposed patch needs another patch as a co-requisite, it will give this information in its feedback.

Now, one can run AutoSeek on this sub-process to see how it can be improved. Specifically, we can identify the input data used by all the approvers and their processing on it. If the processing is information checking or simple, and the control information can be easily specified, and the input data can be inexpensively accessed, then the role of some of the groups can be changed from mandatory reviewers to be-informed using automation.

A way to improve the step now emerges. Is it possible to automatically recommend all the patch dependency information and schedule feedback for a group for a particular type of delivery environment? If so, we can make the checking and decision phases be-informed and improve the overall Step 4 of the patching process through automation.

6 Discussion and Related Work

AutoSeek is a general-purpose method to understand what to automate in a process and how. Even when automation is not the goal, the method can be used to build more contextual information about the activity steps so that input data at the various steps is optimally used. Its advantages are:

1. It seeks to balance the benefit of automation with the cost of modifying the process to implement the automated step. Thus, it recommends activities with high impact and low-risk, low transformation-cost (information checking, taking actions on pre-known and validated generic conditions) to be automated. Drudge and often error-

prone work, is eliminated.

2. It helps in indentifying high-risk and process-critical activities on which sub-system expertise should focus on.

3. It can help in improving processes by recommending re-arrangement of process steps (e.g., to allow traceability) even when no automation itself is recommended. This is because only relevant data is made available to performers when they need it.

4. The process gets better documented.

Implementation: To implement AutoSeek, one needs tools to capture the process model (Steps 2, 3) and then analyze the benefit v/s cost trade-off with the various different automation choices (Steps 4, 5). For the former, as previously mentioned in Section 2, standard tools like WBI Modeler, Visio or tables from word-processors can be used. The latter can be implemented in the process modeling software itself (e.g., WBI Modeler) or as a separate spread-sheet or software application.

We have implemented the analysis part of AutoSeek in a spread-sheet so that it can be directly used by teams managing the service delivery operation. One practical consideration is that the tool should be useful in both extreme case: (a) when there is little profiling data about the process in some delivery environment, and (b) when detailed data is available. To support them, we need to define scales for each factor of assessment and suggest values corresponding to qualitative intuitions when exact numbers are absent. When detailed data is available, they can also be directly scaled to the ranges.

In Figure 5, the implementation is shown for the example process corresponding to Table 5. However, the actual policies that need to be written for the automatable steps needs manual effort and understanding of the process model details. The overall summary of automation achievable is given both in terms of the number of steps and the scales of benefit/cost measures.

The tool can be used for a new process by simply changing the process and step details and entering their corresponding values.

Method Assessment: We have applied AutoSeek on changes related to patching, storage management, hardware changes and software installation for different customers. Moreover, the method and tool is in the process of being made available to a wider set of delivery teams. Initial feedback is that the method serves as a good framework to understand process improvement opportunities via automation and policies. Also, it is general enough to be applied for different types of service delivery processes. The users also like the fact that while the intent is to promote delivery optimization, we also try to factor in costs and risk perception.

Related Work: We are not aware of any methodology to evaluate delivery processes on what steps should be automated and up to what extent. The area of policy-based systems is an active field of research[11][5] and it has been applied to different aspects of IT management, e.g., networking[12]. There are also a number of approaches for quality-driven process improvement like Six Sigma[6]. AutoSeek is complimentary to them as its output about candidate automation process steps can be implemented under a quality-driven execution plan.

7 Conclusion

We presented a methodology to systematically analyze a process to find steps which can be automated in a cost-effective manner and also identify what should be the appropriate level of automation. In doing so, we balance the need for efficiency from automation while considering the cost of implementation and maintenance to perform it. AutoSeek has been applied to different types of delivery processes and has been found effective as a broad framework towards systematically making process improvements. We illustrated AutoSeek with a near-reality patching process and how it may lead to better understanding of process improvements possibilities. To our knowledge, no prior method like AutoSeek exists that guides automation of service delivery processes.

8 Acknowledgments

We will like to thank Guruduth Banavar, Raghavendra Udupa, Milton Hernandez, Larry Davis, Krishna Kummamuru, Vijay Naik, Prasad Deshpande and Sivakiran Yellamraju for helpful comments, and Vandana Srivastava for help with Excel.

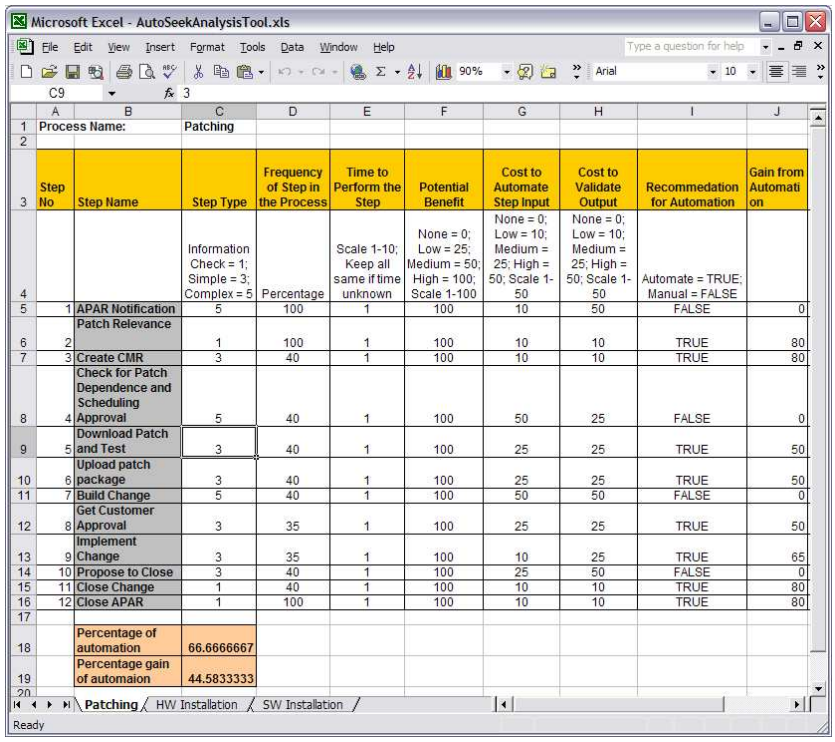
References

[1] OGC, "It infrastructure library (itil)," in *http://www.itil.co.uk/.*, 2006.

[2] J. Kephart and D. Chess, "The vision of autonomic computing," in *IEEE Computer, Vol. 36, No. 1, pp 41-50.*, 2003.

[5] For example, see papers at International Conference of Autonomic Conferences 2004, 2005, 2006.

[6] See http://www.isixsigma.com/sixsigma/six_sigma.asp.



Microsoft Excel - AutoSeekAnalysisTool.xls

File Edit View Insert Format Tools Data Window Help

C9 fx 3

	A	B	C	D	E	F	G	H	I	J
1	Process Name:		Patching							
2										
3	Step No	Step Name	Step Type	Frequency of Step in the Process	Time to Perform the Step	Potential Benefit	Cost to Automate Step Input	Cost to Validate Output	Recommedation for Automation	Gain from Automation
4			Information Check = 1; Simple = 3; Complex = 5	Percentage	Scale 1-10; Keep all same if time unknown	None = 0; Low = 25; Medium = 50; High = 100; Scale 1-100	None = 0; Low = 10; Medium = 25; High = 50; Scale 1-50	None = 0; Low = 10; Medium = 25; High = 50; Scale 1-50	Automate = TRUE; Manual = FALSE	
5	1	APAR Notification	5	100	1	100	10	50	FALSE	0
6	2	Patch Relevance	1	100	1	100	10	10	TRUE	80
7	3	Create CMR	3	40	1	100	10	10	TRUE	80
8	4	Check for Patch Dependence and Scheduling Approval	5	40	1	100	50	25	FALSE	0
9	5	Download Patch and Test	3	40	1	100	25	25	TRUE	50
10	6	Upload patch package	3	40	1	100	25	25	TRUE	50
11	7	Build Change	5	40	1	100	50	50	FALSE	0
12	8	Get Customer Approval	3	35	1	100	25	25	TRUE	50
13	9	Implement Change	3	35	1	100	10	25	TRUE	65
14	10	Propose to Close	3	40	1	100	25	50	FALSE	0
15	11	Close Change	1	40	1	100	10	10	TRUE	80
16	12	Close APAR	1	100	1	100	10	10	TRUE	80
17										
18		Percentage of automation	66.6666667							
19		Percentage gain of automaion	44.5833333							
20										

Patching / HW Installation / SW Installation

Ready

Figure 5. *Implementation of AutoSeek analysis in a spread-sheet.*

[3] J. Bailey, A. Poulovassilis, and P. Wood, "An event-condition-action language for xml," in *Proceedings of the 11th Int. Conf. on the World Wide Web*, 2002, pp. 486–495.

[4] D. Weld, "Recent trends in planning," *AI Magazine*, vol. 20, no. 2, 1999.

[5] Gerald Tesauro and Jeffrey O. Kephart, "Utility functions in autonomic systems," in *ICAC '04: Proceedings of the First International Conference on Autonomic Computing (ICAC'04)*, Washington, DC, USA, 2004, pp. 70–77, IEEE Computer Society.

[6] B. Srivastava, J. Bigus, and D. Schlosnagle, "Bringing planning to autonomic applications with able," in *Proc. IEEE International Conference on Autonomic Computing (ICAC-04), New York, USA.*, 2004.

[7] S. Coste-Marquis and P. Marquis, "Complexity results for propositional closed world reasoning and circumscription from tractable knowledge bases.," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99) , pages 24–29, Stockholm, Sweden.*, 1999.

[8] T. Bylander, "The computational complexity of propositional STRIPS planning," *Artificial Intelligence*, vol. 69, pp. 165–204, 1994.

[9] J. Rintanen, "Complexity of planning with partial observability," in *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling, pages 345–354. AAAI Press*, 2004.

[10] B. Srivastava and S. Kambhampati, "The case for automated planning in autonomic computing," in *Proceedings of the 2nd International Conference on Autonomic Computing*, New York, USA, July 2005.

[11] Jeffrey O. Kephart, "Research challenges of autonomic computing," in *ICSE '05: Proceedings of the 27th international conference on Software engineering*, New York, NY, USA, 2005, pp. 15–22, ACM Press.

[12] D. C. Verma, "Policy-based networking: Architecture and algorithms.," in *New Riders Publ., ISBN: 1-57870-226-7.*, 2001.