Agile Development with Software Process Mining

Vladimir Rubin[1] vrubin@hse.ru
Irina Lomazova[1] ilomazova@hse.ru
Wil M.P. van der Aalst[1,2] w.m.p.v.d.aalst@tue.nl

[1]International Laboratory of Process-Aware Information Systems,
National Research University Higher School of Economics (HSE),
33 Kirpichnaya Str., Moscow, Russia.

[2]Architecture of Information Systems, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.

ABSTRACT

Modern companies continue investing more and more in the creation, maintenance and change of software systems, but the proper *specification and design* of such systems continues to be a challenge. The majority of current approaches either ignore real *user and system runtime behavior* or consider it only informally. This leads to a rather *prescriptive top-down* approach to software development.

In this paper, we propose a *bottom-up approach*, which takes event logs (e.g., trace data) of a software system for the analysis of the *user* and system *runtime behavior* and for improving the software. We use well-established methods from the area of *process mining* for this analysis. Moreover, we suggest embedding process mining into the *agile development lifecycle*.

The goal of this position paper is to *motivate the need for foundational research* in the area of *software process mining* (applying process mining to software analysis) by showing the relevance and listing open challenges. Our proposal is based on our *experiences* with analyzing a big productive touristic system. This system was developed using agile methods and process mining could be effectively integrated into the development lifecycle.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: General; D.2.8 [**Software Engineering**]: Metrics—*process metrics*

General Terms

Experimentation, Design

Keywords

Process Mining, Software Process, Agile Methods

1. INTRODUCTION

ICSSP'14, May 26–28, 2014, Nanjing, China
Copyright 2014 ACM 978-1-4503-2754-1/14/05...$15.00
http://dx.doi.org/10.1145/2600821.2600842

Despite of the growing importance and increasing investments to the area of information systems, the *design and development* of high-quality software continues to be a challenge [5]. Plenty of *software projects* experience significant problems [20, 28, 18]: *software changes* are expensive, software engineers need to deal with pure *technological problems* on a daily basis, business requirements are often defined inaccurately, etc. Moreover, the *complexity of modern software systems* containing millions of lines of code and thousands dependencies among components is extremely high; such systems can be hardly maintained by human beings.

Still, the majority of software design and management approaches continue to be *top-down*: people try to anticipate the possible software changes and to prescribe the whole system behavior in advance. However, emerging *agile methods* [2] provide opportunities for *enhancing* the classical top-down approaches with the *bottom-up* ones. Since software system is delivered and used not at the end of the project, but after every iteration or "sprint" (using Scrum terminology), the running system can be formally analyzed in order to improve the design and development on subsequent steps.

The area of *process mining* provides mature theoretical and practical techniques and tools, which can be used for effective analysis of *software at runtime*. We call this research area *"software process mining"*. In this area, people focus on (1) analysis of runtime traces for improving the architecture and performance of software systems, (2) analysis of the user behavior for improving the design and the usability of software systems.

In this paper, we look at the software process mining from the *software processes* perspective. We show how process mining can be embedded into the *agile development lifecycle* and how the software design and development can benefit from it.

2. PROCESS MINING RESEARCH

Process mining deals with discovering, monitoring and improving real processes by extracting knowledge from event logs available in modern information systems [21, 22]. Nowadays, the event data is constantly growing [10, 14]. This motivates further research in the area of process mining, which basically focuses on bridging the gap between classical *data mining* techniques and *process management*. The *Process Mining Manifesto* released by the *IEEE Task Force on Process Mining* [11] in 2011 is supported by more than

50 organizations, more than 70 experts contributed to it. Today there is a range of successful research and commercial tools available in this area: ProM [26], Disco (Fluxicon), ARIS Process Performance Manager (Software AG), ProcessAnalyzer (QPR), Perceptive Process Mining (Perceptive Software), Celonis and others.

There are three major types of process mining: (1) discovery, (2) conformance checking and (3) enhancement.

Discovery takes an event log and produces a process model from it. Thus, the model is directly discovered from real events recorded in the log. The model can be represented using different *formalisms*: Petri Nets, EPCs, BPMN, Casual Nets and others. A variety of powerful techniques has been developed in this area [23, 24, 7, 9, 25]; they enable creating a *suitable process model* from event log. It is important that this model is as simple as possible and is neither "underfitting" not "overfitting" the log.

Conformance Checking compares process model with the event log. This is used to quantify the differences and to diagnose the deviations. Thus, conformance checking verifies the differences between the model and the real life.

Enhancement takes the log and the existing model and improves or extends the model using additional data available in the log (e.g., to reveal bottlenecks).

Thus, process mining is valuable for the following reasons [22]: (1) It provides *insights* about organizational processes, (2) It allows to check the *process discipline* of the organization (whether processes are executed according to the rules and within boundaries), (3) It is valuable for optimizing and improving the processes and their performance, (4) It enables *prediction* for running processes.

Moreover, after deriving a model containing different process aspects (control flow, informational, organizational), various types of model analysis (*from verification to simulation*) can be done. Since the model is derived from active event data and *reflects the reality*, people diagnose real problems and propose efficient improvements.

From the point of view of a *software engineer*, process mining is a "smart" way of analyzing the logs. Tremendous amounts of event data are produced by modern software systems. From the point of view of a *process miner*, software runtime analysis is a big prospective application domain; in this case mining can be used not only in process-based information systems (e.g. BPMS, ERP, CRM) but in a majority of productive software systems.

3. EXPERIENCE REPORT

Here we present our experiences with the design, development and maintenance of a big *software system* for one of the leading European touristic companies. This system belongs to a class of *Computer Reservation Systems* (CRS) and is used for booking hotels, flights, cars, excursions, activities, trains, etc. This system integrates multiple Global Distribution Systems (GDS) for searching, booking and selling tickets. The system can be accessed by individual customers through web interface, by touristic experts using rich client platform, by travel agencies through special touristic interfaces as shown in Figure 1.

The system has been developed during 5 years by a team of more than 100 persons. The system is based on Java Enterprise platform. The overall source code contains more than 8 million lines of code including server and client code. During the project, the team switched from waterfall approach to an

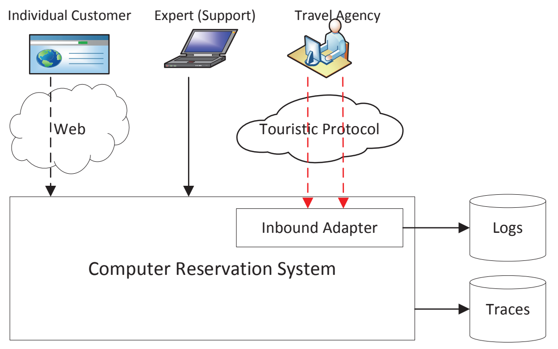


Figure 1: Computer Reservation System

agile (scrum-based) approach [19], which brought significant improvements in communication with the end-users and in the quality of the delivered software. The details of such an agile approach are generalized and discussed in Section 4.

One of the crucial arguments for using the agile approach was obtaining early *feedback from the user* and involving the user in the development lifecycle. After each iteration, the software product was delivered and the users (initially Beta-testers) executed the acceptance tests and utilized the product in their everyday job. After the first iterations, the development team noticed the following issues:

- The users worked with the system differently, not exactly how it was specified in the functional specification and how it was initially expected by the designers. Moreover, the development team did not know exactly how the users worked.

- The system had critical performance issues, their importance was underestimated during the the technical design phase. Moreover, nobody had an appropriate overview of the system and could identify the critical paths.

After experiencing the issues listed above, it was decided to extend the logging and tracing capabilities of the product in order to track the user behavior and the system runtime behavior, see Fig. 1. For example a special logging inbound adapter was created for tracking the behavior of the users in travel agencies[1]. Moreover, very detailed traces of runtime service calls with input and output parameters were introduced to the system.

The analysis of textual logs and traces was complicated for the developers and architects: people were constantly lost in details of particular services and exceptions. Thus, we decided to look for "clever" methods of log analysis and came up with a pragmatic *process mining approach*, which is sketched here.

3.1 Mining User Behavior

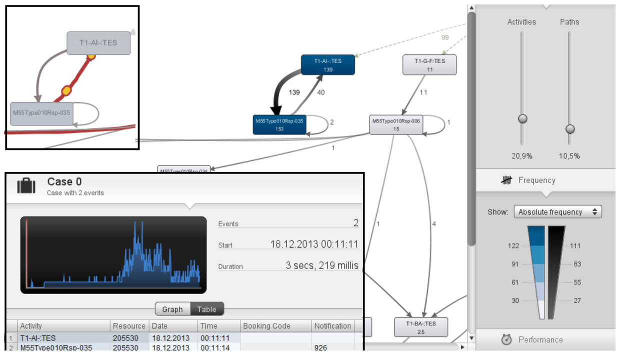
Since most of the reservations are created by travel agencies, we concentrated on these users and on the special proprietary touristic protocol, which was used for client/server communication (see Fig. 1). This protocol is string-based and precisely describes all user operations. Every request

[1]The information about tracking was announced to the users and the users supported the initiative.

string transferred to the server encodes exactly the actions executed by the user. The work with the system always starts and finishes with messages of corresponding types. Thus, we could parse the protocol strings and got exactly the way of work of the user.

A short example of one log entry is: *"Case 1, Show Reservation, 08:30, User 1"*; it contains an id of the corresponding case (instance of a process), activity name, timestamp and user id. We imported a log containing such entries in a process mining tool *Disco (Fluxicon)*[2] and derived a visual model of the user behavior, see the screenshot in Fig. 2. This model was effectively filtered and analyzed, so that we focused on particular activities (e.g. Book Flight, Search Hotel), successful cases or failures, frequent or suspicious behavior, etc.

In order to give a sense of a process mining tool, we outline its basic functionality in Fig. 2. In the background, there is a graph-based process model representing activities as nodes and the order of activities with edges. On the right side, there are different frequency-based filtering options (performance and time-based filtering is also available). In the upper left corner, we show a fragment of a token animation game: the model is simulated. A statistical case-oriented view is shown at the bottom left corner.



Case 0
Case with 2 events

Events 2
Start 18.12.2013 00:11:11
Duration 3 secs, 219 millis

Show: Absolute frequency

	Activity	Resource	Date	Time	Booking Code	Notification
1	T1-AI-:TES	205530	18.12.2013	00:11:11		
2	M55Type010Rsp-035	205530	18.12.2013	00:11:14		926

Figure 2: Process Mining Tool: User Behavior

Using process mining we could do the following: (1) Visualize behavior and discuss it with the user, (2) Visually present successes and failures to the management, (3) Monitor the real usage of the system, (4) Discover the bad usage patterns (later we used this information for improving the GUI), (5) Gather scenarios and develop realistic acceptance tests, (6) Identify most frequent and critical parts of the system, (7) Align system failures with concrete exceptions and create bug fixing issues for the team.

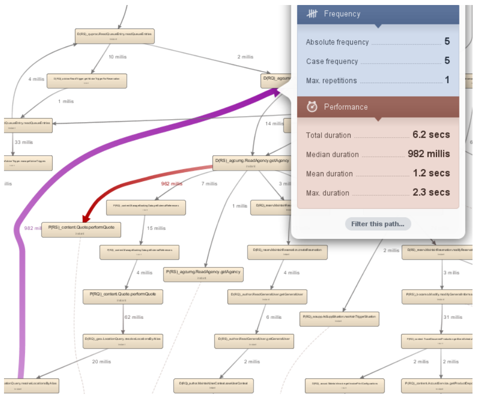
3.2 Mining Runtime Traces

As mentioned in Section 3, the users experienced performance problems. The architecture of the system was built on services, which were orchestrated by the processes. Thus, for each runtime call of a process or a service, we created a trace entry including input and output parameters. Each execution of a process was assigned a unique identifier so that the scope of the process and its service calls could be tracked.

[2]Disco was used because of its simplicity, usability and performance when dealing with huge logs.

A small example of one trace entry is: *"Process 1, Request, GetCustomerInformation, 8:30"*. It contains an id of a process instance, type of call (request or response), service name and timestamp. Again, we used Disco for mining these processes. The software runtime processes were much more complicated than the user processes, Fig. 3 shows a fragment of such model. With the help of filtering by business domain, software component and frequency, we found out the desired performance information.

In the model in Fig. 3, the nodes represent the services, and the edges – the sequence of calls. This is a performance view on the process: service calls with the highest median duration are highlighted and detailed performance information on the selected calls is shown.



Absolute frequency 5
Case frequency 5
Max. repetitions 1
Total duration 6.2 secs
Median duration 982 millis
Mean duration 1.2 secs
Max. duration 2.3 secs

Figure 3: System Runtime Behavior

In the context of runtime analysis, with the help of process mining we could do the following: (1) Visualize and filter real executed processes and services (we discussed also particular behavior with technical designers and architects), (2) Discover performance bottlenecks (later we used this data for improving the algorithms and using caching), (3) Find the most widely used services and the most critical processes, (4) Gather statistics about frequency and message payloads.

All in all, *software process mining* and model analysis worked effectively, so that we decided to embed them into the development lifecycle. After each sprint, we analyzed the user logs and the runtime traces. Our formal feedback was given to the functional and technical designers, appropriate "user stories" and sprint tasks were created and resolved on subsequent iterations. This practical observation motivated us to focus further on software process mining and on its integration into the agile process.

4. SOFTWARE PROCESS MINING IN AGILE PROJECTS

One of the main principles of the agile development is "*Deliver working software* frequently, from a couple of weeks to a couple of months..." [2]. The other crucial principles are continuous *learning* (reflection on how to become more effective) and continuous *communication* with the customer (the end-user). We believe that software process mining approach (1) improves communication with the customer, since we get formal feedback from the user, (2) initiates continuous improvement, since formal results are easily accepted by the designers and included to the product backlogs. We emphasize that almost any real *agile environment fulfills the preconditions* for process mining: software is constantly delivered and used, logs/traces are continuously written.

In Fig. 4 we show a classical agile lifecycle, as it is given in the *Scrum* specification [19]. The planning starts with the presentation of the product backlog, which contains functional requirements. Then, the sprint tasks are defined and the sprint starts. After finishing the sprint, the product is delivered and reviewed. The *review* is normally done by the product owner and by the real users.

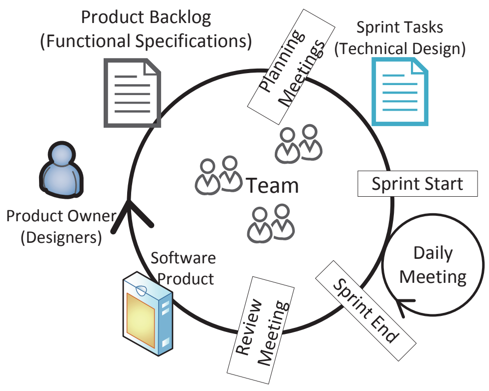


Figure 4: Agile Lifecycle

In Fig. 5 we show how *software process mining* can be embedded into the agile lifecycle. Product owner, users and designers use the product, which is delivered after each sprint. The behavior of the users and the runtime behavior of the system are logged. Process mining uses these *logs*, discovers models and produces the analysis results, which can be used on *further sprints* both for improving functional specification and technical design. The analysis is done by an *expert*, who applies mining tools (see example in Fig. 2) for deriving the most valuable information and presenting it to the *management, designers, developers and testers.*

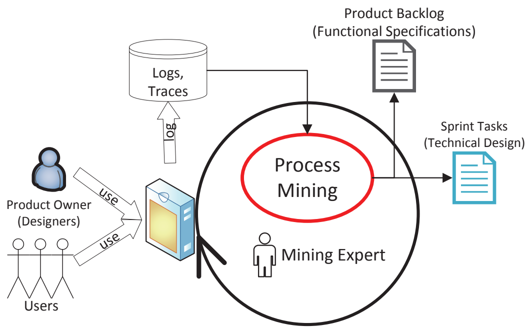


Figure 5: Agile Lifecycle with Process Mining

The results of the analysis of the user behavior usually reveal new or changed *business requirements* and, thus, influence the functionality of the product. They are given to the product owner, he includes them to the *product backlog* and sets the priorities, so that they can be considered in subsequent sprints.

In case of runtime analysis, we discover *non-functional issues* concerning performance and technical infrastructure, these issues should be considered by the *scrum team* when planning the implementation of the next sprints. In case of principal *architectural changes*, they should be discussed by the scrum team, product owner and cross-cutting architecture team, which usually exists in big projects.

5. RELATED WORK

In the area of *descriptive process modeling* people start with the idea of describing the software process as it is being performed [27]. This idea is very similar to the idea of process mining. Cook and Wolf [7] used process mining techniques to discover models of *software processes*. This was one of the first examples of process mining. However, later applications of process mining rarely addressed software process improvement. Today, process mining is mostly used for discovering business process models [21, 22]. Although, we could effectively use process mining for *discovering software process models* from software repositories [17, 12], this topic received little attention.

Software process mining is a developing research field. However, a lot of work was already done in the area of mining user behavior, especially in *web usage mining* [15] and applying Markov models for web usage mining [4]. Also further steps concerning using historical states for improving web mining were done in this area [6]. Also the adjacent research field dealing with mining *search query logs* is very prospective [3].

The other area is dynamic analysis of software execution traces. *Performance analysis* using system execution traces is described in the work of Peiris and Hill [16]. A trace analysis framework for creating test suites is described here [13]. A separate research domain is *program comprehension* using dynamic analysis, a deep survey of current research is done in the work of Cornelissen et al. [8].

The ideas about integrating performance analysis into software development lifecycle are nicely summarized in [1].

6. FUTURE WORK AND CONCLUSIONS

Many modern software development projects are organized according to the *agile principles*. Early *feedback of the user* is a crucial success factor for such projects. With the help of *process mining*, a *model* of user behavior is discovered and analyzed in order to get deeper insights into the functionality of the delivered software. Additionally, a model of software runtime behavior is derived in order to localize performance problems and architectural challenges.

Based on our experiences with the development of a touristic system, we have shown how to integrate process mining into the *agile development lifecycle*. Since most of the software systems produce a variety of logs and traces, process mining can be effectively applied for analyzing these systems. This paper is just a first step in the area of *software process mining*, it opens numerous research directions and challenges.

Further research should be done in the following directions: (1) Mining different perspectives, e.g. data and organizational perspective, (2) Application monitoring with the help of process mining (model differences and model repair), (3) Mining and filtering on different levels of abstraction (using process-based OLAP cubes), (4) Efficient mining of big data (gigabytes of logs), (5) Predicting user behavior and system failures, (6) Discovering architectural anti-patterns and redesigning the system, (7) Discovering usability anti-patterns and improving the "quality in use" of the software.

Acknowledgments

Support from the Basic Research Program of the National Research University Higher School of Economics is gratefully acknowledged.

7. REFERENCES

[1] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, 2004.

[2] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development, 2001.

[3] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: Model and applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[4] J. Borges and M. Levene. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Trans. on Knowl. and Data Eng.*, 19(4):441–452, Apr. 2007.

[5] F. P. Brooks. *The Design of Design: Essays from a Computer Scientist.* Addison-Wesley Professional, 1st edition, 2010.

[6] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really markovian? In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 609–618, New York, NY, USA, 2012. ACM.

[7] J. Cook and A. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

[8] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis. *IEEE Trans. Softw. Eng.*, 35(5):684–702, Sept. 2009.

[9] A. A. de Medeiros, A. Weijters, and W. van der Aalst. Genetic Process Mining: An Experimental Evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[10] M. Hilbert and P. Lopez. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025):60–65, 2011.

[11] IEEE Task Force on Process Mining. Process Mining Manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer-Verlag, Berlin, 2012.

[12] E. Kindler, V. Rubin, and W. Schäfer. Activity Mining for Discovering Software Process Models. In B. Biel, M. Book, and V. Gruhn, editors, *Proc. of the Software Engineering 2006 Conference, Leipzig, Germany*, volume P-79 of *LNI*, pages 175–180. Gesellschaft für Informatik, Mar. 2006.

[13] R. Lencevicius, E. Metz, and A. Ran. Tracing execution of software for design coverage. In *Proceedings of the 16th IEEE International Conference on Automated Software Engineering*, ASE '01, pages 328–, Washington, DC, USA, 2001. IEEE Computer Society.

[14] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers. Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, 2011.

[15] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, Aug. 2000.

[16] M. Peiris and J. H. Hill. Adapting system execution traces to support analysis of software system performance properties. *J. Syst. Softw.*, 86(11):2849–2862, Nov. 2013.

[17] V. Rubin, C. Günther, W. van der Aalst, E. Kindler, B. van Dongen, and W. Schäfer. Process Mining Framework for Software Processes. In Q. Wang, D. Pfahl, and D. Raffo, editors, *International Conference on Software Process, Software Process Dynamics and Agility (ICSP 2007)*, volume 4470 of *Lecture Notes in Computer Science*, pages 169–181. Springer-Verlag, Berlin, 2007.

[18] C. Sauer, A. Gemino, and B. H. Reich. The Impact of Size and Volatility on IT Project Performance. *Commun. ACM*, 50(11):79–84, Nov. 2007.

[19] K. Schwaber and M. Beedle. *Agile Software Development with Scrum.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[20] The Standish Group. Chaos manifesto 2013. `http://versionone.com/assets/img/files/ChaosManifesto2013.pdf`, 2013.

[21] W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer-Verlag, Berlin, 2011.

[22] W. van der Aalst. Process Mining. *Communications of the ACM*, 55(8):76–83, 2012.

[23] W. van der Aalst, V. Rubin, H. Verbeek, B. van Dongen, E. Kindler, and C. Günther. Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. *Software and Systems Modeling*, 9(1):87–111, 2010.

[24] W. van der Aalst, A. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[25] J. van der Werf, B. van Dongen, C. Hurkens, and A. Serebrenik. Process Discovery using Integer Linear Programming. *Fundamenta Informaticae*, 94:387–412, 2010.

[26] H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst. ProM 6: The Process Mining Toolkit. In M. L. Rosa, editor, *Proc. of BPM Demonstration Track 2010*, volume 615 of *CEUR Workshop Proceedings*, pages 34–39, 2010.

[27] Y. Wang. *Software Engineering Processes: Principles and Applications.* CRC Press, April 2000.

[28] E. Yourdon. *Death March.* Yourdon Press Series. Prentice Hall Professional Technical Reference, 2004.