Built-to-Order Service Engineering for Enterprise IT Discovery

Nikolai Joukov, Murthy V Devarakonda, Kostas Magoutis, and Norbert Vogl
IBM T. J. Watson Research Center
Hawthorne, NY

Abstract

Enterprise IT environments are complex: business applications rely on distributed middleware running on diverse hardware with components depending on each other in many unexpected ways. Discovery of applications' dependency on IT is a critical step in managing application and IT infrastructure together.

Many tools and practices have emerged to discover and report IT assets and applications' dependency on the IT assets. However, our experience in the field shows that there are significant challenges in effectively deploying the tools. There is a critical need to research and develop flexible processes, methods, and practices, and architecture-level support for them in the tools to enable successful discovery using a "built-to-order" approach.

In this paper we discuss our experiences with an advanced application-data relationship discovery tools in large scale enterprise environments and based on these experiences we identify three main challenges of effective discovery. They are: deployment process and related security issues; unavailability of software and administration-related information; and tool integration. To address these challenges, here we demonstrate a holistic approach that includes flexible processes, methods, and practices in the tools for achieving the necessary built-to order capability.

1 Introduction

Modern enterprises have sophisticated IT infrastructures. Their numerous servers run a diverse set of applications which serve various functions, interact with each other, and use distributed middleware [16, 14]. Servers and their operating systems are also diverse, configured differently, and managed by different people who, frequently, do not even interact with each other. As a result, there is usually no person or an existing automated way to derive information necessary for most IT assessment and transformation tasks.

Some typical reasons for needing information about the whole IT structure include: (1) It is necessary, for the purposes of intrusion prevention, to discover all enterprise servers and routers. (2) Software license management optimization may require finding all installed instances of a software package. (3) Before a service delivery begins it is usually a good idea to verify that the enterprise description provided in the contract corresponds to the actual infrastructure. (4) eDiscovery, in the form of software and data relationships discovery, can help find who and which applications may potentially access sensitive information to prevent its leaking. (5) Capacity planning and performance problems elimination require discovering the traffic patterns between all involved assets and thus also the relationships between the assets. (6) Planning of recovery actions and availability-related analysis of important applications require discovering all involved assets together with their mutual dependencies. (7) Virtualization and migration of servers and applications requires intimate knowledge about the software and hardware components and their detailed dependencies.

Not all potentially discoverable information is necessary for a given type of analysis. For example, statistics about the inter-node loads of requests may be sufficient for capacity planning and identification of performance problems. However, for most IT optimization tasks, one needs detailed information about the systems starting from the business processes down to applications, middleware, hardware, and their configurations (preferably at the level of individual data objects such as EJBs, database tables, messaging queues, and files) together with their dependencies. In addition, this information must be accurate so that no assets or dependencies are missing from the discovered information. For example, one needs comprehensive information about all data objects and their dependencies to plan recovery and data reliability at the level of data objects. Similarly, a single missed database dependency even if not used frequently may make a virtualization or migration effort impossible or undesirable. We call this form of discovery *comprehensive discovery*.

In the ideal world one would collect all the information about the enterprise IT assets, their dependencies, associated people, policies, business functions, and run-time uti-

978-0-7695-3283-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCC.2008.128

lization in one database [13]. Analysis tools would in turn analyze the information from the database and generate reports, policies, and recommendations or even transform the IT environments automatically.

Today very few organizations have deployed any automated discovery tools and even if such tools are deployed the information collected is barely enough for the very basic analysis. As a result, people who need the information for analysis have to either deploy the tools themselves or perform the discovery manually or mostly manually. Unfortunately, the last two options still overwhelmingly dominate.

In this paper we describe the real-world enterprise IT comprehensive discovery process based on our experiences with designing and using Galapagos—a comprehensive data dependencies discovery system [14] and deployment of IBM TADDM [15]. We analyze the reasons why in many cases people refrain from using the automated and comprehensive discovery. We have identified and described in details three groups of challenges that limit the use of discovery tools in the enterprises: (1) deployment process, (2) difficulties with the discovery of certain types of information, and (3) information integration. Finally, we present our own findings about the optimizations of the discovery process and present recommendations for the design of the discovery tools and practitioners.

The rest of the paper is organized as follows: We describe existing methods of IT asset and dependencies discovery in Section 2; Sections 3, 4, 5 describe the challenges of deploying the discovery tools, discovering custom and human-related information, and information integration respectively together with some solutions and recommendations; we conclude in Section 6.

2 Existing Discovery Methods

We estimate that there are more than 60 commercial offerings related to the infrastructure discovery in addition to the research prototypes. We will not describe all of them here. Instead we describe several typical projects.

Tools to discover hosts on the network by sending SNMP (Simple Network Management Protocol) requests have been around for a while. For example, nmap [10] was first published in September 1997 [9]. Many modern tools such as IDD [15] and Infrastructure Solutions Inc. (ISI) Snapshot [3] use this as a form of initial discovery. SNMP-based discovery is usually followed by a scanning of open ports and other probing requests. Analysis of the target system behavior allows them to discover signatures, identify OSs and OS versions, and infer some information about the other hardware and software components. Such network scanning tools require special handling by the firewalls and intrusion detection systems. As a result the scans are usually performed from the nodes located behind firewalls.

Modern routers are usually equipped with the special ports that allow monitoring the network traffic. Some discovery tools such as Aurora [12], eMulsa [11], and EMC Smarts Application Discovery Manager (ADM) [8] exploit this possibility to derive the information about the network nodes and their software. The main benefit of this approach is that it is usually not hard to get access to the routers and their monitoring ports (if supported). Also this method allows discovering the relations between software components and nodes by inspecting the packet contents [4]. In addition, observation of the network traffic provides information about the server load, which is important for tasks like capacity planning and performance problems debugging.

Observing the nodes from outside does not allow to discover detailed and complete information about the systems. This is simply because accessing configuration information and file system contents requires credentials to avoid information exposure and system compromise. A number of special software interfaces allow accessing such configuration with the proper credentials (e.g., [2]). However, such specific management interfaces have limited discovery capabilities because they are usually not designed for discovery. Also, they are not available for all software components and usually not deployed.

Monitoring or inspection of the logs of the run-time activity on the target systems is similar to the network traffic analysis on the routers but can reveal information of the system internals. For example, IBM Tivoli Application Dependency Discovery Manager (TADDM) [15] can use `lsof` command to discover a list of files accessed by any applications. This type of run-time analysis is useful for the performance analysis and capacity planning purposes. Unfortunately, run-time monitoring does not allow to discover all dependencies (e.g., a list of files that an application may use in the future or uses very rarely) and also may require instrumentation of the production systems. This creates problems for several types of analyses (e.g., recovery planning) and can significantly complicate the tool deployment.

2.1 Galapagos and other agent-less tools

Agent-less discovery tools such as Galapagos [14], TADDM, and HP Discovery and Dependency Mapping (DDM) [1] have per-application or per-application type sensors that log-in to the target systems and execute system commands or query application-specific interfaces. With proper credentials such sensors can use expert knowledge about the software components and installation-specific information obtained from the system to discover most if not all details about every system.

Galapagos is a novel discovery technology that finds individual data objects such as EJBs, database tables, and files and their interdependencies. For example, Galapagos can

map every database table to all lower files that a table implemented by DB2 depends on. The core idea behind this discovery is the modeling of the software components. In the example above, a Galapagos model of DB2 contains knowledge about the way DB2 maps tables to files based on DB2 internal rules and configuration files.

In this paper we focus on the use of Galapagos and TADDM in the enterprises.

3 Comprehensive Discovery Process

The process of discovering comprehensive information about an IT environment is usually divided into three stages, sometimes called levels:

1. The scanning process begins by discovering the information at the network level (e.g., the list of servers);
2. Next, common information about the servers is collected (e.g., CPU speed, list of running processes, and mounted file systems);
3. Lastly, sensors specific to particular IT components derive information specific to these components (e.g., data base table names and their dependencies).

These three levels of discovery are, in most cases, followed by the discovery of relationships between assets and their business roles (including their importance). There are two main related reasons behind the staged discovery:

Nearly every discovery attempt challenges the security policies of an IT environment. One needs credentials and authorization to perform discovery. Naturally, the scanning process is divided according to the authorization levels - first, go behind the firewalls; next, obtain user-level access to the servers; and lastly, obtain application-specific access to specific files or applications.

Each discovery level is performed at a "deeper" layer of system abstraction and an earlier-level discovery provides information necessary for the later levels. For example, one needs to know which OS is running on a server to deploy level two and level three sensors that are different for different OSs. Similarly, one needs to know which credentials to request and which application administrators to contact to perform application-specific discovery.

The exact information discovered at different levels varies greatly depending on the technology and implementation. For example, NMAP-based network discovery tools may provide server IP addresses and OSs, whereas tools that monitor network traffic can discover some applications and certain dependencies between them. Furthermore, multiple tools from different vendors may be used concurrently because they discover different subsets of information that can be derived at a given level.

3.1 Discovery Process Steps

The above described discovery process is shown in Figure 1 for the environments we studied. Each discovery level consists of a similar sequence of steps and multiple tools may be used at any given level. Nearly every step may alternately be performed with manual discovery via interviews and templates. The entire process may have to be restarted because of security policy changes or turnover of the personnel in managing the assets. In addition to these rare occurrences, relatively frequently, one may end up contacting wrong people or be presented with wrong or incomplete corporate policies. Let us now consider each discovery step.

2. Find people The deployment process begins by finding the right people for a given level of discovery. For the first level one needs to find people who know about the network topology and network intrusion detection policies in place. For the levels two and three it is necessary to find people responsible for the servers or individual applications respectively together with the corporate security supervisors. In many enterprises, the number of individuals to contact may be very large since different portions of an organization or even different classes of applications may have different policies and people responsible for them. In reality, this step consists of multiple meetings. During these meetings the set of people involved gets refined. Also, frequently meetings get rescheduled.

3. Send spreadsheets Upon finding the right people with sufficient knowledge of the required information they are required to fill out standardized surveys usually in the form of spreadsheets. Such spreadsheets allow a degree of automation while collecting the initial information about the enterprise. Also, they become reusable artifacts that can help other forms of analysis [6]. Unfortunately, the spreadsheets are designed for a general case and usually have more questions than is necessary for a particular engagement.

4. Learn policies and describe scanning After identifying the right asset owners and getting the common information via spreadsheets, it is important to get "buy in" from asset owners so they personally feel comfortable with the scanning procedures. Asset owners are typically concerned about the performance impact of the discovery tools and the amount of work that they personally will have to do. Also, at this step, it is important to learn the specifics of security-related policies.

The information necessary to create custom or update existing sensors is also gathered at this step. We will describe custom sensors in Section 4.

6. Modify/create sensors It may turn out that an existing asset sensor is not general enough to cover some existing configuration. In that case the person performing the discovery may request that tool maintainers update the sensor.

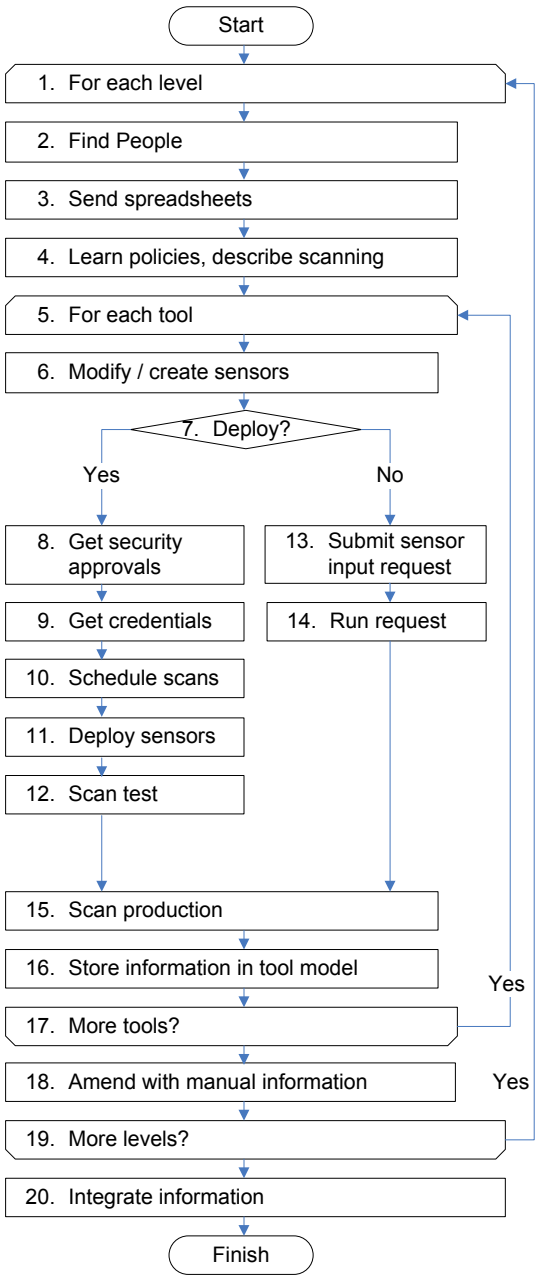

Figure 1. A typical comprehensive enterprise IT discovery process. For simplicity we do not show the arrows here depicting events that can happen at every step: a person performing the discovery can give up at any step and switch to the manual discovery via interviews and spreadsheets; the level may have to be restarted at any step if there is a significant policy change or change in the set of people responsible for the assets.

Similarly, it may be necessary and possible to create custom sensors for rare or unique applications. This topic will be described in Section 4.

Security and deployment policies may require modifications of the main discovery tool or sensors. In particular, a sensor may use a system command that is not supported or contradicts with the security policies. In that case the command itself may be reimplemented to perform more specific and allowed actions (e.g., access `/proc` interface instead of using `lsof` command) or the sensor may be changed to collect less data. In the worst case, one may be forced to change the scanning tool interface for accessing the sensors. For example, `ssh` access to the servers by automated tools is sometimes forbidden but is allowed via an `http` port; Java-based sensors are not allowed in some environments or require complicated approval procedures.

Obviously, such code updates cannot be performed for every engagement. However, for important engagements or if the problem persists many times tool and sensor updates are more than justified.

7. Deploy? Sometimes, it turns out that the differences between the existing tool and its deployment process and allowed enterprise policies is so big that it is unpractical to perform the deployment and update the code of the tool itself. In that case one may skip the tool deployment process as we will describe in Section 3.2.

8. Get security approvals It is important to get formal approval of the deployment and scanning process from the bodies responsible for security of the scanned assets. Sometimes such a body is a formal review board where one needs to satisfy a list of formal requirements. There may be even several separate boards, e.g. one responsible for the application deployment (if the discovery tools is classified as application) and another for credentials. In other cases for smaller organizations one may get a permission based on less well defined requirements. In any case, it is critical to get a formal approval of the deployment process from the security organizations. Without formal approval, asset owners may resist the deployment process because of the fear of violating corporate security rules.

During the security review process the system is usually inspected in terms of what kind of credentials are necessary and how these credentials are stored. Many per-application sensors require credentials that may potentially give access to the sensitive corporate data. In these cases it is desirable to minimize the level of access requested. For example, Galapagos needs access to information about IBM DB2 database table names but does not require access to the data stored in the tables. Whereas IBM DB2 `db2admin` group members can access both tables and data, members of the `db2mon` group can see only the table names. However, the `db2mon` group may not be defined by default, and therefore it may have to be created for the scanning process.

If the permission level required by the sensors is deemed to be too intrusive and no lower permission groups exist, it is sometimes possible to obtain policy exemptions by showing the specific commands executed by the sensors. Unfortunately, this requires extensive familiarity with the sensors and practitioners may not have such familiarity. This implies that sensors must be well documented.

Interestingly enough, we observed that in some cases it is possible to convince the approvers that a scanning tool is secure and get permission to broad access rights for all systems. However, this process requires negotiations and depends on the personalities involved. This approach has drawbacks but helps speed up the deployment process because, instead of requesting many per-application credentials it is possible to request one set of broad access rights (e.g. `root` level access).

9. Get credentials The actual process of requesting credentials is usually automated for the large enterprises. However, even this automation requires efforts of supplying application names and other information to the automated systems. In addition, system administrators may have to create additional groups such as `db2mon` described above.

10. Schedule scans Frequently no perturbations of the IT infrastructure are allowed during the times of important business activity. For example, no scans or reconfigurations may be allowed during the end of quarters because of the importance of the accounting activity that may not be risked or interrupted. This means that the discovery process may have to be postponed till the end of such IT change freezes.

11. Deploy sensors Before the actual automated discovery process can begin one may need to install additional software or hardware. For example, one may have to replace routers that do not support Netflow information collection. Similarly, it may be necessary to install some monitoring software tools that are standard but not present on all systems by default such as `lsof`. This is yet another example where sensor improvement may be justified to make the deployment easy. Instead of expecting `lsof` and similar commands to be present on the system one may create sensors that use the `/proc` interface directly. In other cases it may be necessary to install sensors on the target systems because no external log-in interfaces exist.

12. Scan test To make sure that the sensors will run correctly and show to the application owners that no performance degradation is expected, it is sometimes desirable or required to run the discovery on the test servers. Such servers frequently exist for each corporate application and are used for development and testing purposes.

15. Scan production With the proper approvals, credentials, sensors, and prepared target systems it is now possible to run the discovery on the actual production systems.

16. Store information in tool model The discovery tool stores the discovered information in its own output format.

17. More tools? If a given tool does not provide all the information necessary (e.g., has no sensor for some middleware component), it may be possible to run another tool and merge the results later.

18. Amend with manual information As we discuss in Section 4, not all aspects of IT systems can be captured automatically even with perfect sensors and all credentials. Therefore, it is necessary to add that information for a given level manually.

19. More levels? After a given level of scanning is complete one may run the next level to get more detailed information if necessary for the analysis.

20. Integrate information Information collected by different tools at different levels must be integrated for the subsequent analysis. Even if the discovery process was performed with only one tool it is still necessary to make the results compatible in their representation with the analysis tools. Section 5 addresses this step further.

3.2 Comprehensive Discovery Without Credentials and Deployment

As we can see, steps 8–12 are lengthy, nontrivial, and require serious involvement of the asset owners. In fact, they usually correspond to the biggest portion of the whole discovery effort. These efforts can be justified for long-term deployment when the discovery tools are deployed once and then used for years. However, for short-term and single-time discovery engagements, the cost of deployment is usually not justified and that is why manual discovery is used in so many cases. While deploying Galapagos and performing the above steps, we realized that there is a way to achieve the same accurate and automatically derived results with significantly less efforts.

We noticed that most if not all sensors issue a small number of commands or read a small number of configuration files and these are the only operations when they need credentials and access to the target systems. After that the sensors perform analysis of the output of these commands or configuration files. Therefore, we found it much easier to request an asset owner to issue a couple of commands or send us some configuration files than going through the whole deployment process. Instead of going through steps 8–12 we do the following as shown in Figure 1.

13. Submit sensor input request We write a detailed description of the files or commands that our sensors need and send them to the asset owners.

14. Run request Asset owners perform the actions requested and send us the output sometimes simply via email.

It is important to note that this manual acquisition of the information necessary for the scans is different from the manual scans. Instead of asking the asset owners about a complete and big set of properties and dependencies we request them to execute simple commands or copy configuration files. It is the scanning step that we do automatically that discovers accurate and comprehensive information. However, instead of performing the discovery on the target system we perform it on our own host with the same sensors but with the production system's data as input.

Overall, we would like to make two conclusions from our discovery experiences: First, it makes sense to make the discovery tools as flexible as possible—inflexible tools in terms of deployment have limited applicability especially for the short-term engagements. Second, in some cases it is possible to discover precise and comprehensive information without a completely automated discovery process.

4 Sensors and Information Discoverability

Information about IT systems is protected with various security mechanisms. For example, one needs special credentials to access software configuration files. As we described in Section 3.1, the less privileged credentials a sensor needs the easier it is to deploy and use it. For example, the complexity of the discovery level one sensors is mostly due to the fact that they are inferring information about the systems and even their software stack without logging on to the systems. Such sensors may know that a particular OS's IP stack implementation may have a characteristic behavior in handling a particular type of a network handshake protocol and identify that implementation. However, even when the maximum possible level of access to all systems and all components is available it is hard or even impossible to discover some classes of the IT-related information:

1. Custom-made applications are common in the existing enterprises. They require creation of the custom-made sensors, which is long, expensive, and error-prone.

2. Some important information that is frequently critical for analysis is not contained in any common digital form and thus cannot be discovered without human intervention or heuristics that do not always work.

We describe these problems related to the discovery of information even if all credentials are available below.

4.1 Custom-made Sensors

Automated discovery tools use predefined mechanisms and templates to collect information. They have limited intelligence to understand the system properties. Therefore, it is much easier to discover information about standard and widespread systems and manual intervention is usually necessary to collect information about the custom-made systems. As a result, it is easier to discover information at the lower discovery levels: custom-made hardware and operating systems is a rarity. However, software middleware and application components are more diverse and thus harder to describe by the sensors in all cases.

Figure 2 shows a diagram of a real custom-made enterprise application R that consists of about two million lines of Java code. We will use it to illustrate the difficulties in the discovery of custom-made applications and their dependencies. First, from a discussion with the application team manager we learned the high-level business purpose of the application and the fact that it runs on a single dedicated server together with a dedicated database and is accessed by other applications via a messaging queue. This allowed us to create a first-approximation sensor that assumed that application R depends on all database tables that exist on the server and provides services via a messaging queue with a given name.

Soon we were able to meet with one of the application architects. It turned out that the application consists of two parts: R1 and R2. R2 reads in a big file and generates tables that R1 uses to answer external requests coming via a messaging queue. The file itself is being fetched via `ftp`

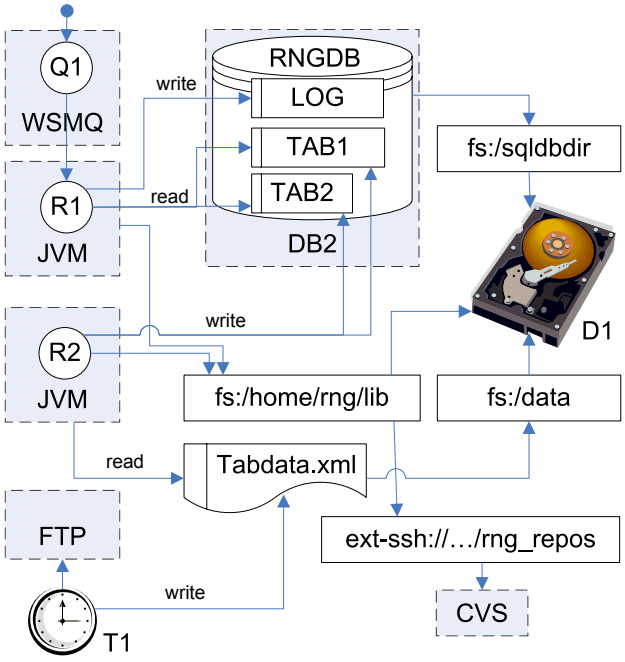

Figure 2. A real-world application R with external and internal dependencies.

from a remote server once in two days by a `cron` job. We were also shown the location of the application binaries and the data file. After looking at this information we realized that there is a missing dependency between the application binary and its source files. It turned out that the source was kept on a separate CVS server. This information was sufficient for us to create a model of the application. We plugged it into the rest of the discovered information about the database and the servers that communicate with application R remotely to find all mutual dependencies in the enterprise. For example, another application located on another server that is sending requests via the same messaging queue depends on database RNGDB.

Despite of the fact that the process described above is manual it allows us to perform the discovery automatically, no matter how many times one needs to perform the discovery. Another interesting observation is that the actual sensor implementation effort is much simpler than the application investigation, which is required for both: manual and automatic discovery.

In other cases we observed applications that use small but numerous shell scripts, which source may be read by the discovery team. Similarly, a web server may have a large number of small CGI scripts. In these cases source code analysis may be an easy way to create the models.

Numerous security-related projects automatically analyze application access patters by observing the run-time behavior and by analyzing the code. We believe that in the future some of these technologies could be used to construct the application models automatically.

4.2 Human-Involved IT Dependencies

There are at least two types of information that cannot be reliably discovered completely automatically:

Manual IT processes (e.g., manual backup, application recompilation, and data movement). For example, application R's sources are stored remotely on a CVS server and a person who wants to renew the application manually retrieves it from that server. This means that one needs to know this dependency when planning the recovery of application R. In this particular case one may infer the CVS repository location by observing the presence of the `CVS` directory but this heuristics has a limited applicability.

Business value and purpose Another example of information that cannot be in general discovered automatically is the business value and purpose of applications and application data objects. For example, we can observe that the LOG database table is only used for writing and no IT services depend on it. However, we cannot automatically decide that we do not need to back it up because it may be necessary for several types of manual analyses. (In case of application R it turned out that all database tables were regeneratable locally and the LOG table was unimportant so no database tables required backup.)

Deriving the business value of an application is even more challenging and is almost always done completely manually. The only rare exceptions are heuristics such as dependence of the server or application name on the business role if such rules are enforced in the enterprise.

5 Integration of Tools and Information

Discovery tools can store the information they discover in three ways:

1. Use a custom data schema, which implies manual or custom analysis of the discovered information;
2. Use an existing asset database that is supported by other discovery analysis tools; and
3. Integrate the tool into a federated service.

When we first designed Galapagos we used a custom data representation scheme because it is abstract, flexible, and other existing models that we considered did not support the types of objects and dependencies that we needed (because others do not discover them). We stored the information in an SQL database and complex SQL queries were able to generate sophisticated reports about end-to-end data and application dependencies [14].

As we used Galapagos we realized that (1) Galapagos needs information from other tools to start the discovery; (2) it would be convenient to share credentials and UI with other tools like TADDM; (3) there are existing analysis tools that support TADDM's data representation but do not support our custom model. As a result we started the effort of merging Galapagos and TADDM into one tool. It even turned out that the changes of the existing TADDM model are small and non-intrusive.

In todays highly competitive IT services market, solutions to a variety of services consulting and delivery engagements are provided through the concerted action of a set of tools. In addition, clients often demand the use of cross-brand tooling in the delivery of integrated business solutions. For these reasons, Service Providers require information-integration technologies to enable the seamless interoperation of tools provided by different software vendors, as well as those prototyped and deployed by research organizations. Oftentimes the use of research tools can provide unique, differentiating capabilities to a Service Providers offering portfolio.

A solution to the problem of interoperability across a set of tools can be provided by technologies that support metadata federation services (MFS) [5, 7]. MFS offer integration across a portfolio of service offerings through the

use of a common, shared metadata repository for metadata exchange between the tools. This exchange is particularly suited for model-driven tools, whose underlying metadata are captured and described in a formal model.

We demonstrated information integration between IBM Rational Data Architect (RDA) and Galapagos, enriching RDAs information with the relationships between database entities (tables, columns, etc.) and the applications that are accessing them. This allowed us to integrate Galapagos into a framework that lets other existing and future tools to use Galapagos-provided data for analysis automatically.

6 Conclusions

Automated IT discovery tools is a key enabling technology for automation of the IT management and optimization tasks. Unfortunately, comprehensive discovery tools are hard to deploy and use. We have presented our experiences with the real-world deployment of two state-of-the-art comprehensive discovery tools: Galapagos and TADDM.

1. We described a step-by-step deployment process and showed that most development efforts spent on the tool flexibility are more than justified. Powerful but inflexible tools are hard to deploy, which makes them useless for short-term engagements. We also demonstrated that a combination of the automated and manual discovery can significantly cut the deployment efforts and still provide most of the benefits of a completely automated deployment.

2. We demonstrated that custom-made software and human-related IT dependencies are hard or even impossible to reliably discover automatically. However, it is not hard to discover them manually and integrate into the overall reusable automated discovery process.

3. We showed that there is still no common way to address the data and tool incompatibility problems in the industry. Therefore, modern tools have to support a variety of data representations and invocation methods concurrently. A custom data model may be best for a stand-alone deployment; tight integration of UI and data models may be good for tools frequently used together; an MFS-based integration is useful for cross-brand and wide-scope integration.

Acknowledgments We would like to thank Alain Azagury, George Galambos, Phil Harvey, Birgit Pfitzmann, Marco Pistoia, Harigovind Ramasamy, and Raymond Twitchell for guidance, numerous discussions of the project, and participation in the Galapagos deployment.

References

[1] HP discovery and dependency mapping (DDM) inventory software. `www.hp.com/hpinfo/newsroom/press_kits/2007/softwareuniversebarcelona/ds_inventory.pdf`.

[2] IBM tivoli storage manager. `www.ibm.com/software/tivoli/products/storage-mgr/`.

[3] ISI-snapshot... agent-less accurate and rapid IT infrastructure inventory, configuration and utilization collection using a single tool. `www.isiisi.com`.

[4] S. Basu, F. Casati, and F. Daniel. Web service dependency discovery tool for SOA management. In *Proceedings of the IEEE International Conference on Services Computing (SCC 2007)*, pages 684–685, Salt Lake City, Utah, July 2007. IEEE.

[5] M. Beyer. Why metadata matters to business intelligence initiatives. Technical Report G00144814, Gartner Research, September 2007.

[6] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Proceedings of the 5th International Conference on Business Process Management (BPM 2007)*, pages 288–204, Brisbane, Australia, September 2007.

[7] M. Blechar. IBMs federated metadata management strategy. Technical Report G00147616, Gartner Research, April 2007.

[8] M. Bowker, B. Garrett, and B. Laliberte. EMC smarts application discovery manager. Technical report, ESG Lab Validation Report, July 2007.

[9] Fyodor. The art of port scanning. *Phrack Magazine*, 7, September 1997.

[10] Fyodor. *NMAP(1)*, 2003. `www.insecure.org/nmap/data/nmap_manpage.html`.

[11] H. Kashima, T. Tsumura, T. Ide, T. Nogayama, R. Hirade, H. Etoh, and T. Fukuda. Network-based problem detection for distributed systems. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, pages 978–979, Tokyo, Japan, April 2005. IEEE.

[12] A. Kind, P. Hurley, and J. Massar. A light-weight and scalable network profiling system. *ERCIM News*, January 2005.

[13] H. Madduri, S. S. B. Shi, R. Baker, N. Ayachitula, L. Shwartz, M. Surendra, C. Corley, M. Benantar, and S. Patel. A configuration management database architecture in support of IBM service management. *IBM Systems Journal*, July 2007.

[14] K. Magoutis, M. V. Devarakonda, N. Joukov, and N. Vogl. Galapagos: Model-driven discovery of end-to-end application-storage relationships in distributed systems. *IBM Systems Journal*, February 2008. to appear.

[15] C. R. Rich. TADDM's flexible approach to discovery. Technical report, IBM Corporation, July 2007. `ftp.software.ibm.com/software/tivoli/whitepapers/TADDM_s_Flexible_Approach_to_Discovery.pdf`.

[16] J. Scheck. Business technology: Taming technology sprawl—HP hits snags in quest for savings through systems consolidation. *The Wall Street Journal*, January 2008.