Available online at www.sciencedirect.com
SCIENCE DIRECT®
Science of Computer Programming 61 (2006) 4–15
Science of Computer Programming
www.elsevier.com/locate/scico

Interface descriptions for enterprise architecture

Aditya Garg[a,*], Rick Kazman[b], Hong-Mei Chen[b]

[a] *FedEx Corporation, Dallas, TX, United States*
[b] *University of Hawaii, Honolulu, HI 96822, United States*

Received 18 March 2005; received in revised form 23 September 2005; accepted 5 November 2005

Abstract

This paper describes our experiences in implementing a repository-based system that records and manages architectural information for an organization-wide set of enterprise information systems. The system is called Interface Descriptions for Enterprise Architecture (IDEA). The IDEA system business case and system creation are discussed, and the use of the system is exemplified by applying it to a set of information systems within a large corporation. We show how the existence of a managed enterprise architecture description enables an important set of analyses: risk analyses, architecture analyses, and change management analyses. Finally, we discuss how the IDEA system has the potential to catalyze even more profound change by changing how the various systems' stakeholders interact.

Keywords: Software architecture; Enterprise architecture; Architecture analysis

1. Introduction

Large corporations with substantial IT infrastructures face a challenge in making those infrastructures responsive to the needs of the corporation's stakeholders. This corporate IT infrastructure is typically built from a diverse set of systems and frequently these systems do not work as expected because the systems do not fit together well. This is the well-known problem of architectural mismatch [5].

Maintaining a system's relevance to the business process is time-consuming and expensive and this process grows exponentially difficult with the number of systems in the enterprise. With the corporation's systems being interfaced to each other in ever more complex ways, we realized that we needed a mechanism for inventorying and managing the entire IT enterprise as a 'system of systems' on an ongoing basis. Such a goal seems obvious, but as yet there are no tools and techniques for achieving it. As Ben Menachem states: 'no one has identified commercial tools for building and maintaining an enterprise-level software inventory' [2].

Inventorying an IT enterprise is a long term goal. In this paper we will discuss one step towards that goal: managing the interfaces among the IT systems within an enterprise. As we will show, a view of interface characteristics assists in architectural analyses of change impact and in the identification of technical risk areas, such as overburdened and

* Corresponding author.
E-mail addresses: agarg@alumni.cmu.edu (A. Garg), kazman@sei.cmu.edu (R. Kazman), hmchen@hawaii.edu (H.-M. Chen).

0167-6423/$ - see front matter © 2006 Published by Elsevier B.V.
doi:10.1016/j.scico.2005.11.001

redundant systems, overly tight coupling, and single points of failure. This visibility not only helps the corporation address its most critical architectural bottlenecks in the short term but also helps in the longer term transition towards an agile Information Enterprise.

To this end we have created the IDEA (Interface Descriptions for Enterprise Architecture) system. In this paper we will discuss the problem space that the IDEA system addressed, and our experiences in implementing a first version of the system.

2. Background

As stated in the introduction, there is little research and no software tools that directly address the problem of creating an enterprise level IT inventory. However, there are a number of research areas that approach at least a part of the problem.

For example, much of the work in the field of Business Process Reengineering [6] assumes that an organization has some way of inventorying its IT assets, as a precursor to reengineering them (along with the business processes). BPR approaches, however, tend to assume that an appropriate inventory of assets already exists, or is easily gotten. While some attempts to systematize the relationship between BPR and reengineering (including reverse engineering) [1], this has been widely ignored in industry and in the research literature [12].

The field of enterprise architecture modeling has similar goals to those presented above, but enterprise modeling is almost entirely concerned with the problem domain—that of 'business models' [18], or 'business processes' [13], or generic models of architectures [19]—and is not focused on the solution domain. For example [4] notes: 'An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise.' As such, an enterprise model is crucial in questions of IT alignment [7] since it probes issues of business operations, but it sheds no particular light on the IT inventory problem.

The field of reverse engineering, on the other hand, does provide a potential solution to the problem of identifying IT assets. Considerable attention has been paid, in particular, to architecture reverse engineering (e.g. [11,17]) which addresses the problem of extracting architectural representations from source artifacts. However, this approach still fails to provide a solution to the problem of understanding IT architecture, which (a) is bigger than the architecture of any single system; and (b) requires an understanding of not just the architectural structures of a system, but also a knowledge of the interconnections established at run-time, the protocols used to realize those interconnections, and the run-time properties of those interconnections. Thus far even architectural reverse engineering has no satisfactory solutions to this challenge.

The ITIL (IT Infrastructure Library) is an industry-led forum aimed at improving the practice of IT service management [8]. It contains six 'service support disciplines', each of which addresses a key area of IT management. One of these disciplines is configuration management. While this discipline could, in principle, cover at least part of the problem involved in creating and maintaining an IT inventory, it would need significant tailoring and tool support to make this discipline both rich enough and usable enough to address the set of issues covered by IDEA.

3. The problem

Given the lack of available tools and techniques, the IDEA system project was created. It was targeted to provide the first solution of the 'IT inventory' problem initially addressing the needs of a single IT department within a large corporation that we will refer to as 'BizCo'. We will call this department the 'Web Products Department' (WPD). The department needed a mechanism to answer the following questions:

- What are the systems that exist in the department TODAY?
- How are they interfaced to each other and to systems outside the department?
- What are the properties of the interfaces?
- What is the impact of a change to an interface or a system?
- Which systems are burdened with excessive interfaces and which systems can be targets for architectural refactoring?

Past approaches to solving the problem of managing the intertwined dependencies of the enterprise architecture in the corporation were:

1. **Expensive**, because they required dedicated resources, who were often the most talented and knowledgeable architects.
2. **Time consuming**, because they required extensive special-purpose collection and analysis of data.
3. Frequently **incomplete**, because it became impossible to defend the expense and time commitment to document every aspect of every system's interfaces.
4. **Out of date** as soon as they were complete because the ongoing expense of maintaining this information was deemed to be unacceptable by management.
5. Of **limited use** because the resulting data was typically not universally accessible.

Although this system was only targeted to the WPD, the problems facing the department, in terms of managing its enterprise architecture, were substantial. The analysis that we performed, as we will show, revealed significant enterprise architecture risks.

The approach that we chose to address these issues was to delegate the responsibility for maintaining every system's description to the team that was responsible for supporting the system. These descriptions would be inexpensive to create and maintain because creating these descriptions would be part of the normal duties of system support teams, who know the domain well and could describe their systems and interfaces quickly, efficiently and completely. To motivate the users, instant feedback of their data entry was considered important. Giving every user in the company access to the same 'system of systems' view that is both accurate and up-to-the-minute is a great asset and this potential corporation-wide scrutiny serves as a motivation for everyone to keep their descriptions complete and accurate.

With access to this information, each department is able to have an up-to-date interface view of its own domain of systems. In addition, each department has the ability to analyze how these systems interact with each other. In effect, we were aiming to create small 'user groups' for each system in the WPD.

To realize this vision, the IDEA project created a tool to address the following requirements:

1. A web based data input mechanism, to allow each system and interface description to be entered and updated independently.
2. Descriptions to be stored in a commercial database.
3. The system would offer interface owners the capability of tracking consumers of their interfaces, as an incentive for these owners to input descriptions of their interfaces.
4. All company employees would have access to the repository on the intranet.
5. Users would have the ability to search for interfaces and applications by various parameters.
6. Users would also see a graphical view of applications currently described and the connectivity of the various applications through interfaces.

4. Risk management

It is imperative for a collaborative tool like this to achieve a critical mass of usage for it to be useful. To mitigate the risk that this critical mass might not be reached, the following strategies were implemented:

- Identifying a stakeholder group and getting buy-in (initially from the stakeholder group, eventually from a wider audience) was considered critical. The project therefore attempted to identify and involve influential people with a wide representation in the department.
- Identifying the most important requirements and implementing them first. Having a core set of features that showed users the 'system of systems' view would enable its early use and demonstrate its utility to a wider audience. To achieve this, the project continuously sought feedback from the stakeholders regarding the requirements prioritization as well as identifying requirements that were not yet covered.
- To sell the concept to various stakeholder groups within BizCo, a marketing strategy was devised which consisted of selling the usage of the IDEA system to the widest possible audience, finding key champions, and use the system itself as a 'viral marketing' tool by inviting every interface owner identified within the system to use the system.

5. Requirements

Our experiences with several unsuccessful enterprise architecture efforts in the past at several large corporations indicated a need to collect and display interface descriptions in a manner that:

- Is not effort intensive or expensive to create.
- Is easy to keep updated with minimal effort and expense.
- Allows imperfect information, overlaps, and gaps to exist.
- Is user friendly and does not force everyone to learn a specialized language or toolset.
- Is easily accessible for reference by everybody and provides visibility into the current state accurately.
- Supports analysis of change impact and other related queries.

5.1. Architecture drivers

The following business and quality attribute drivers influenced the architectural solution eventually chosen for the IDEA system:

1. **Open standards:** should be used where possible.
2. **Scalability**: the system needed to scale to approximately 5000 users. The initial user base was 150. Usage would be intermittent (less than daily), but could occur in bursts.
3. **Performance**: the system should be responsive (<1 s response per query).
4. **Reliability**: the system is required to provide service during the normal office hours but is not mission critical even during those hours.
5. **Security**: no access to the system will be allowed unless the user is identified. No update will be allowed unless the user is authorized to make that update after a significant amount of data is present in the system and it is in a 'live' state.

To ensure that these were the appropriate focus areas, the architect discussed potential pitfalls with stakeholders. The most important roadblock identified was the motivation for the interface owners to enter data completely and accurately. The incentives already identified were considered useful but not sufficient.

To add another major incentive, it was decided to introduce an interface change management process to the data collection tool. This would allow the interface owners to keep consumers of their interfaces informed of upcoming changes and to coordinate the actual change process. The consumers would realize the benefit of having assured early warning of interface changes. This was a major functionality addition and was considered a high priority.

6. Architecture and design

A deployment architecture diagram showing the IDEA system as it is being used in production is given in Fig. 0.

6.1. Design decisions

The various design decisions that resulted in the architecture presented in Fig. 1 are listed below:

1. The application was deployed on the corporate intranet using the web browser as the client interface, as shown in Fig. 2. This made the application accessible to everyone in the company and obviates the need for distributing software updates.
2. A 3-tier architecture was implemented as shown in Fig. 0. All of the data resides in a relational database, there is a middle tier that contains the application as well as presentation logic (in separate layers), and a standard web browser augmented with applets serves as the presentation tier.
3. J2EE technology was used to deploy the application on a central server. Having a central server allows for easier capacity addition, higher development productivity and ability to better support the application in the production environment.
4. In the middle tier, Jakarta Struts [14] and Java Server Pages (JSP) [10] were used for managing user interaction. Both Struts and JSPs are widely used in the organization, have a proven track record, have tool support and are stable products. The architect, who is also the main developer, understands this technology well and thus it presented a low risk to the project.

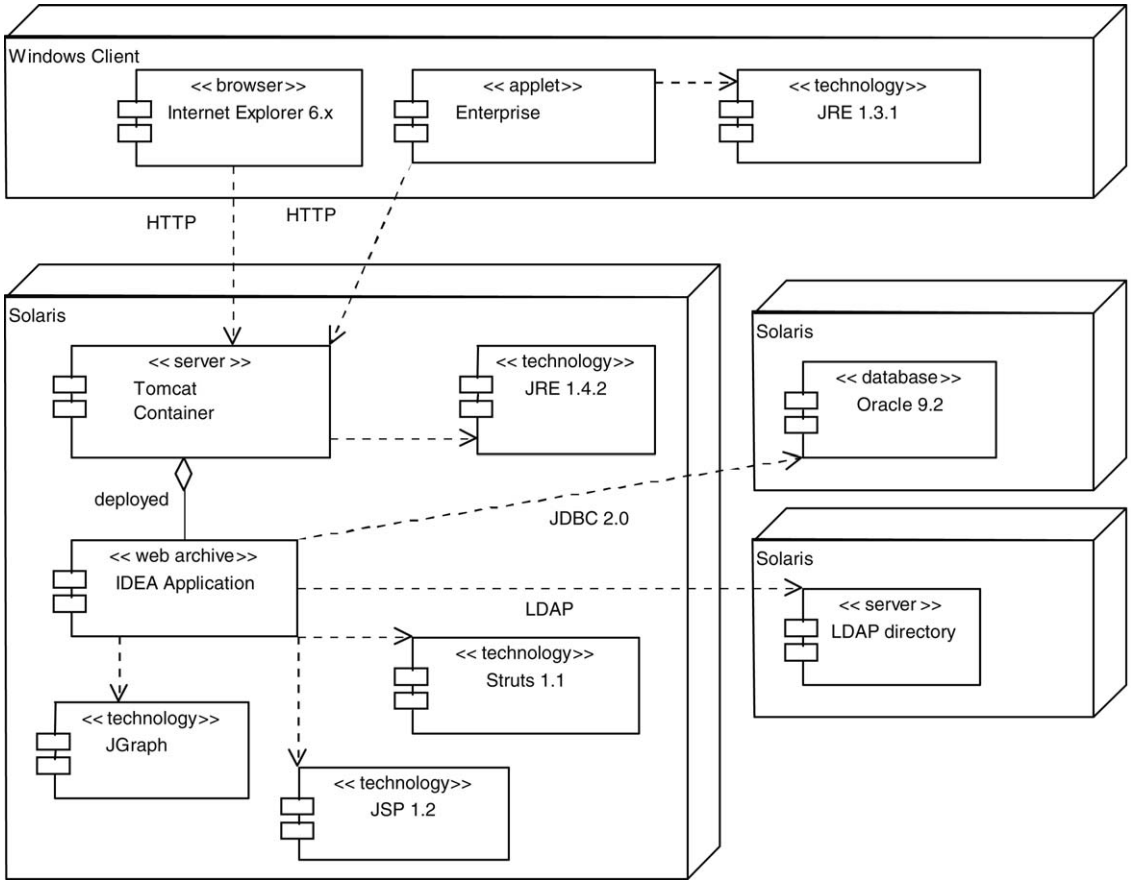

Fig. 0. IDEA system deployment and component diagram.

5. EJBs were not used for the IDEA system. While EJBs provide substantial infrastructure support, they are, in our experience, complex, cumbersome, and suffer from performance problems. Thus, their inclusion was not justified by the needs of the system.
6. Apache Tomcat Servlet engine was used as a servlet container. This is an open-source (and free) stable product with high performance. It has also been tested extensively in industry as well as by the architect, therefore presents a low risk. It is embeddable in the Apache Web server which is also open source (and free), stable and well tested. The Apache Web server will also be used for the production environment. This will allow for future scalability. The Apache Web server is also the company corporate standard for the Linux platform, thus presents a low learning curve for future support personnel.
7. Authentication is provided by using the company standard LDAP based directory. Each user will be identified for every session. Authorization is supported by the IDEA system directly, based on simple rules.
8. An automatic global identifier (GUID) is assigned to Component and Interface records on their insertion into the database. To keep this functionality independent of the database, a dedicated GUID table is used to mimic the functionality of Oracle sequences.
9. Online help and tips are provided on each page to assist in data entry and navigation as needed.
10. JGraph [9] was utilized for static diagram generation. This product is available free under public license, which satisfies the main drivers for IDEA. It also does not pose a significant risk. At worst, the product can be removed from the application if found unsuitable at a later stage without causing a significant loss in functionality.
11. Touchgraph [15] was utilized for interactive diagram generation. This product is also available free under public license. It also does not pose a significant development risk. At worst, the product can be removed from the application if found unsuitable at a later stage without causing a significant loss in functionality.

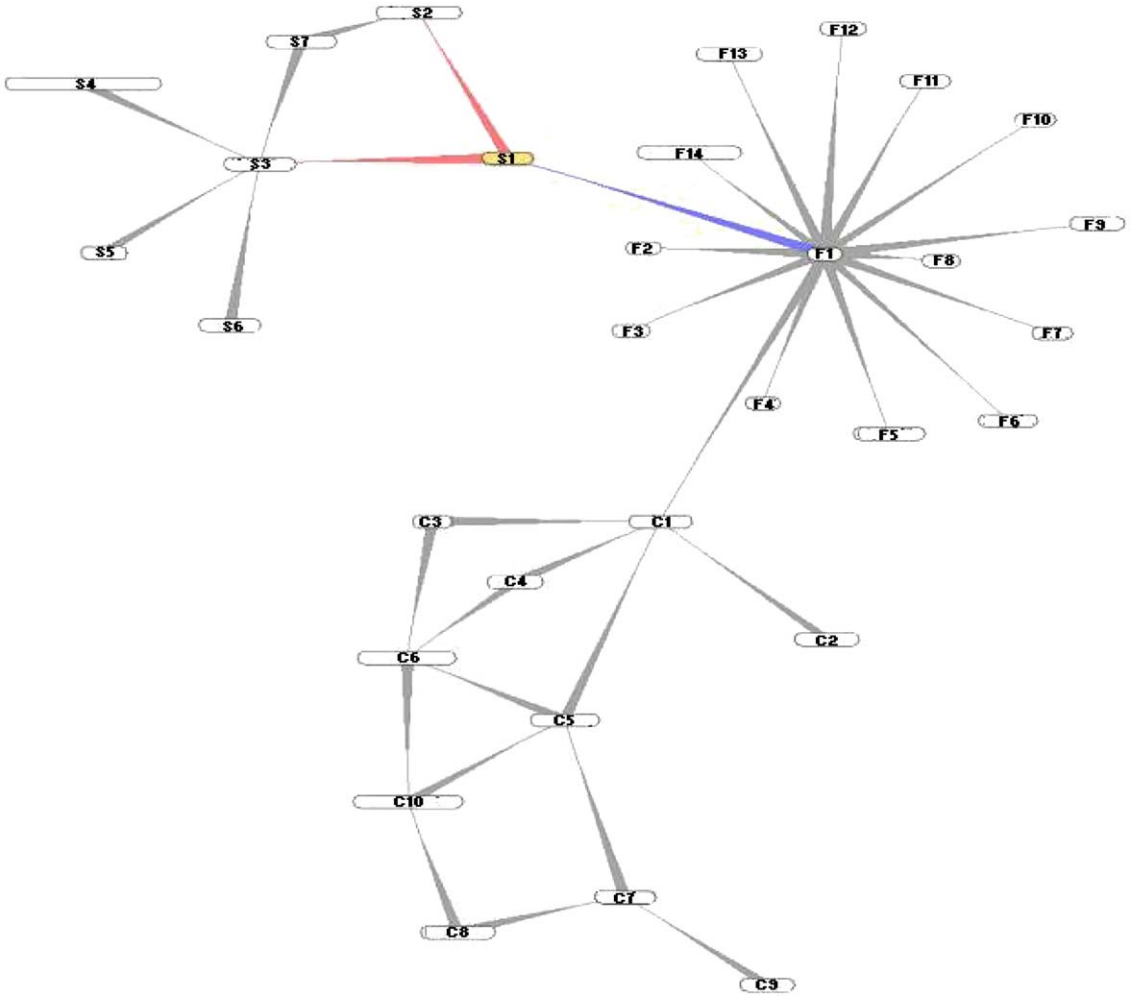

Fig. 1. Applications and dependencies.

7. Implementation

Once the project proposal was finalized, a group of stakeholders was identified and their feedback was sought on the proposal. Based on the feedback, an interface change management process was added to the list of requirements. An activity diagram was created to map out the business process and it was refined over several iterations in consultation with the stakeholders. Simultaneously, the use cases were identified and documented. The Use Case document was formally reviewed with stakeholders.

Simultaneously, a marketing strategy was created. This marketing strategy was aimed at elaborating the mitigation strategy for the primary risk to the IDEA project—the inability to reach a critical mass. During use case elaboration, some analysis was also conducted to establish the relationships between the major data entities: Component, Interface and User. Per the marketing strategy, specific applications were identified for initial data entry consideration to build up the critical mass.

To design a good flow between web pages of the application and to work out the usability issues, an early prototype was created. Paper prototypes of screen layouts were also created in order to finalize the look and feel of each page. This flow and layout was validated with a small group of stakeholders.

After the architecture and detailed design of the application were finalized and documented, an architecture analysis [3] was performed and the primary technical risks were identified. These were then targeted to be mitigated using risk reduction prototypes before the main development began.

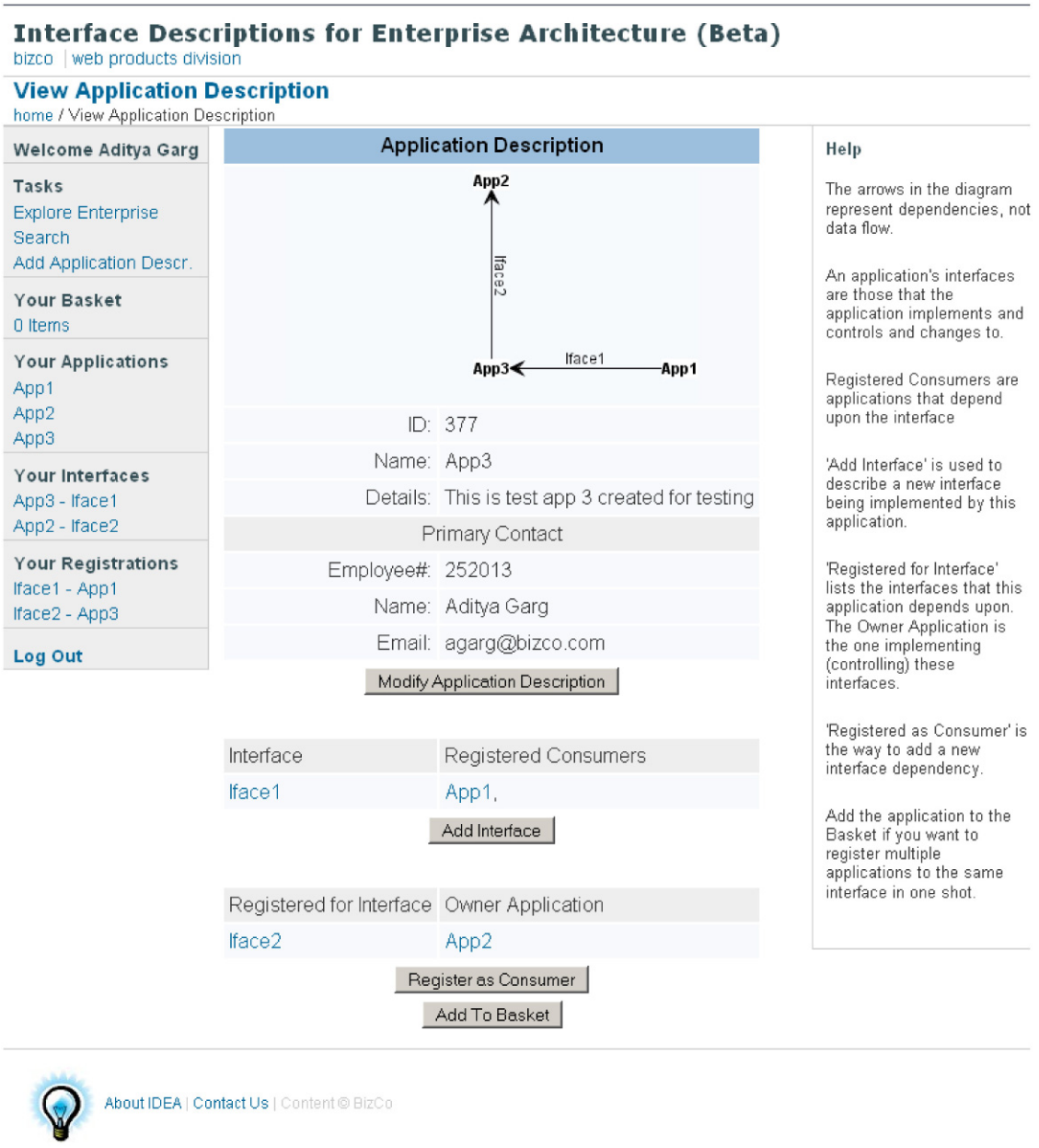

Fig. 2. IDEA's web-based interface.

The strategy for controlling schedule risk involved deferring some functionality, such as diagram generation, towards the end of the project with the expectation that, if required, this functionality would not be implemented for the first product release. After the first fully working release iteration, the next major iteration was expected to implement the change management process and more detailed capture of interface properties. Evaluation of the marketing strategy and feedback received from stakeholders indicated that this may not improve the marketability of the product. It was felt that a major selling point would be automatic diagram generation, which would present a picture hitherto hidden from the common view, thus immediately engaging and inspiring the prospective user or stakeholder. Therefore, the requirements priority was changed to research and implement automatic diagram generation.

The system was deployed to a Solaris based server backed by an Oracle database instance in the development environment. This allowed remote users to easily access the system. Online help and an introductory tutorial on the IDEA application was added at this time. An email was sent to several targeted groups to help promote the application.

With more feedback, testing and data entry of real applications and their interfaces, the system has become sufficiently usable and many usability problems have been fixed. Recently, the database and application servers have been migrated due to changes in the development environment.

8. Results

The IDEA tool was deployed on the BizCo intranet following the component diagram given in Fig. 1. It was demonstrated to a group of WPD stakeholders, and made available to the stakeholders to enter data for the initial set of applications.

Based upon the first three months of adoption among the initial stakeholder group, the following results have been realized in the deployment of the IDEA system.

8.1. A 'system of systems' view

Perhaps the most important result is that the IDEA system has made available to stakeholders a view of the enterprise that was hitherto extremely difficult to synthesize. This view has been enabled by the tool's ability to collect data and make it visible in a form that is vetted for accuracy and visualized in a meaningful way. Although not a complete view of the enterprise, it meets the project objectives and is a solid beginning towards the eventual goal.

Data has been collected for 32 applications, which are dependent upon each other through 59 interfaces and registrations. This network is shown in Fig. 2. To understand the directionality of the dependency links, consider the S1 application. S2 and S3 are dependent upon S1 and S1 is dependent upon F1.

8.2. Project objectives

The IDEA tool adequately addresses the questions that were defined as the project objectives. It is able to identify the systems that exist in the WPD today, it shows how they are interfaced to each other and to systems outside the department, and it shows the properties of the interfaces. In addition, it is able to aid an analyst in determining the impact of a change to an interface or a system, as we will describe. We will also describe how the IDEA system aids in an analysis of which systems are burdened with excessive interfaces and which systems can be targets for architectural refactorings.

The IDEA tool also satisfies the business drivers:

1. The tool is highly usable by a typical IT user without a need for specialized knowledge. The only knowledge needed is the knowledge of the system/application whose data is being entered and an elementary knowledge of using a web browser. Using a browser also supports the capability for universal access. This has been verified using a survey, detailed later.
2. The content requires only minimal effort to maintain. This has been enabled by deploying the application on the Intranet and by empowering the system support teams to enter data about their own systems. They do this to maintain accurate, reliable control over who is using their interfaces.
3. The tool does not depend on any proprietary technologies.
4. The tool allows overlaps in content by allowing duplicated definitions of systems and interfaces to be created. Additionally, the entire 'system of systems' does not need to be defined for this tool to be useful, thus tolerating gaps in content.
5. The tool allows identification of systems and people impacted by an interface, and thus facilitates impact analysis. Future versions of the tool are targeted for supporting change management by allowing the interface owners to send out notifications of change and collaborating with interface subscribers about features and timings of a change release.

Some results are already possible to discern in terms of analysis of existing data, and will improve with further quantity and detail of data collected. Sample analyses are discussed below.

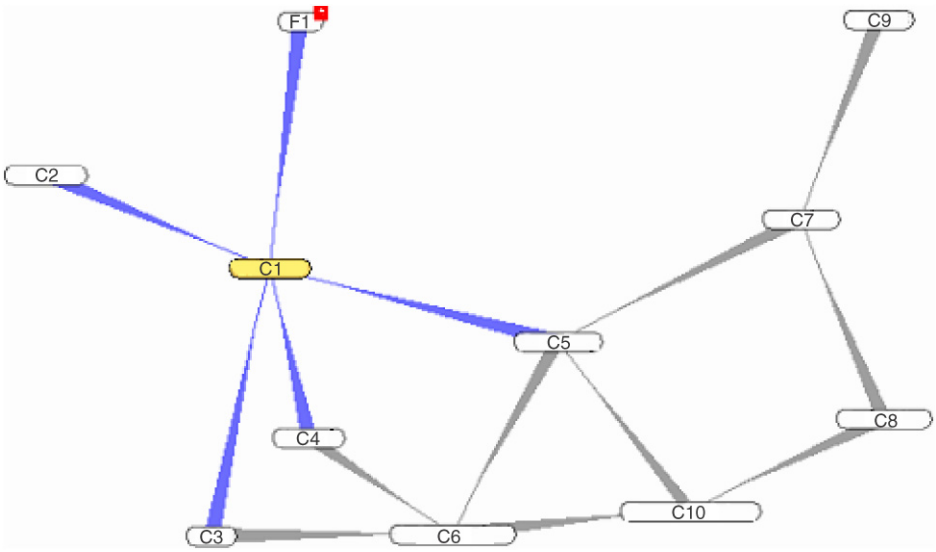

Fig. 3. C1 dependency network.

8.3. Architectural analyses

Fig. 3 shows the interface dependencies of the C1 application, as created by the IDEA system's Enterprise Explorer applet. Links with the narrow ends towards C1 (C2, F1, C3, C4, and C5) indicate that C1 is dependent upon these systems (or, more precisely, on the published interfaces to these systems).

Reliability: Initial analysis from Fig. 3 that C1 is dependent upon its immediate interfaces to 5 applications. However the nature of the interfaces is such that C1's ability to provide correct service is dependent upon the interface dependencies of those systems. Therefore, C1's reliability (its ability to offer correct and timely service) is dependent on not only the 5 applications listed above, but also on C6, C7, C8, C9 and C10. If any of these applications is not able to provide correct service on the relevant interfaces due to a fault, C1 will be impacted. This network of dependencies means that the reliability of C1 is inversely proportional to the size of this network.

Although this information was, in principle, available prior to the creation of the IDEA system, it was held in principally in the minds of a select few architects. In practice, this information was never packaged, validated, and visualized in a way that made it usable to the average developer. The IDEA system makes this information available in a single broadly accessible repository.

Redundancy: Another point about the network presented in Fig. 3 is that many of these applications only provide a thin wrapper around existing functionality. For example, C5 provides an EJB interface to C10 and C7. C1 provides the BizCo.com wrapper to this functionality. C4 provides Web Service and EJB wrappers around other tracking applications (not shown here). Their justification for existence as separate applications in their own right is tenuous. Rather, they are acting as integration glue and could be targets of architectural simplification.

A portion of the information needed to reach these conclusions is not currently collected in the IDEA system, but could be easily collected and utilized directly in future stages. Currently, the C1 architect was an additional source of information for this analysis. But simply having the view shown in Fig. 3 was an important discussion and elicitation device, so even without the additional information being in the repository, the IDEA system still served as an important aid to analysis.

8.4. Risk analysis

Fig. 4 shows the dependency diagram for F1. The dependency links with the broad ends towards F1 indicate that F1 provides the interface(s) that the corresponding applications are dependent upon. It is obvious when looking at this picture that F1 affects 15 applications, or, more precisely, that 15 applications depend upon F1. Knowing the nature of the interface as being a synchronous query/response, it is again possible to deduct that F1 is a single point of failure and presents a high risk exposure. In addition, the high transaction rate imposed on this application by the

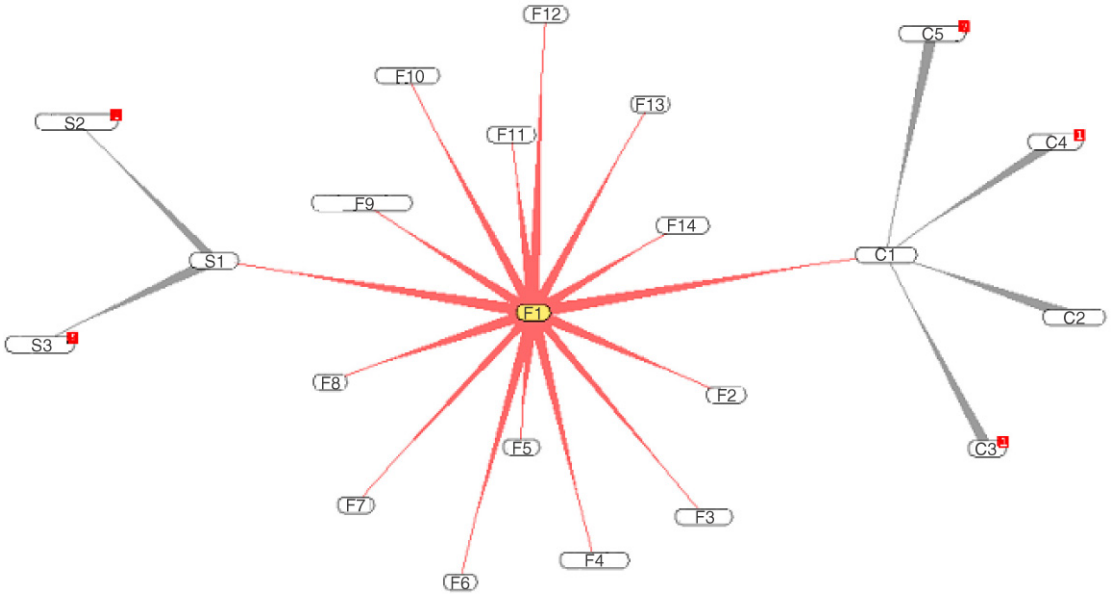

Fig. 4. F1 dependency network.

other applications is also a concern. This is a preliminary analysis and it uses some information that is not yet captured in the IDEA system (which can, however, be easily added). But this example illustrates the kinds of analyses that are possible to support with greater enterprise architecture information.

This is a preliminary analysis and it uses information that is not yet captured in the IDEA system (which can, however, be easily added). But this example illustrates the kinds of analyses that are possible to support with greater enterprise architecture information.

8.5. Other potential results

Other possibilities for analysis exist as well. Some of these have been identified by the stakeholders as current capabilities and some are future capabilities. These are listed below.

1. In the quarterly move to production process today, many applications with complex inter-dependencies are moved into the production environment. It is usually very difficult to determine the impact of a single application being unable to make the deadline. Using the IDEA tool, it may be possible to evaluate the risk profile of these 'quarterly move to production' processes based on the dependencies in the targeted application set.
2. It is possible to communicate scheduled and unscheduled system outages to affected consumers via the system's dependency information and the fact that every interface description has associated with it a primary and a secondary contact person.
3. The IDEA tool enables the human network by making it possible to discover people. For example, it is possible to find fellow consumers of the same interface and query them about interface details while doing research on a new interface or troubleshooting a problem thus enabling virtual user groups.
4. It may also be possible to create a dependency graph several levels deep based on functionality, interface type and direct dependencies. This would allow a more realistic impact analysis. Specific functional paths can be traced through various systems to narrow down the dependencies. This is useful for determining exactly which systems need to collaborate for completing a specific use case. It is also useful in determining the entire dependency network needed to complete all use cases for a single system.
5. An interface change management process tied to the software development process could ease the problems currently facing software development. The tool could allow much more sophisticated interface change management by allowing the interface owners to send out notifications of change, allowing all existing consumers to approve the change. If everybody approves, the interface gets 'promoted' on the target date. If everyone does not approve, either the upgrade is cancelled, delayed, or the interface is forked to allow some consumers to upgrade later.

8.6. Survey results

We conducted a small survey with 6 of the system's most important potential stakeholders to evaluate whether the initial problem addressed was worth solving, and whether it was solved appropriately. Some of the major findings of the survey are described below.

1. Most of the respondents felt that researching unfamiliar applications and interfaces currently was difficult and took anywhere from 50 to 300 h per interface per project. (Additional comments elicited in interviews with the stakeholders indicated that this effort was also repeated for the same interface at a later date.)
2. To conduct research into enterprise applications and their interfaces the major mechanism that the respondents used was to ask their colleagues and, in some cases, to use the corporate intranet. In other words, the process of research and discovery was completely ad hoc.
3. The respondents used a variety of methods for tracking consumers of interfaces they control. Some form of mailing list was the most common.
4. Most of the respondents indicated that impact analysis of proposed changes was either not conducted or, if conducted, was based on guess-work.
5. Most of the respondents indicated a high likelihood of using IDEA for application and interface research, especially for finding people/systems related to their own systems.
6. Most of the respondents expect a significant improvement in interface change management once the IDEA system is broadly deployed.
7. The Enterprise Explorer functionality with the interactive diagram was considered very useful and most respondents judged it to be the best feature of the IDEA system.

These results provide some preliminary support for two conclusions: that the problem is indeed a serious one that was worth solving, and that the problem was solved appropriately by the IDEA system.

9. Conclusions and future work

In its short lifetime the IDEA system has already provided important insights into BizCo's enterprise architecture that was hitherto unavailable. The benefits that it has provided to BizCo's WPD are as follows. First, it has provided a 'system of systems' view, where none existed. Secondly, it has provided a means of centrally storing enough information about enterprise systems such that overlaps were able to be identified (and explicitly blessed, if need be). Third, and perhaps most important, it has enabled a number of analyses that were either missing or were difficult and costly to perform previously. These architectural and risk analyses provided new insights into BizCo's system of systems. IDEA helped to highlight a number of significant risks relating to redundancy and reliability within the enterprise architecture.

Several of the lessons learned in this exercise were reinforcements of concepts familiar from prior experience. These are discussed next. (1) The IDEA system concept is simple, yet it is quite powerful. It illuminates the structure of IT in the company in a manner that no other tool or technique currently does. This shows the power of visualization, when properly coupled to data that allows users to 'drill down' to the details when necessary. (2) Face time is crucial in promoting a new idea. This fact was reinforced by our attempts to promote the product to people who were off site or traveling. (3) It is important to manage risk continuously and doing so can result in a successful implementation. The IDEA system was considered to be difficult to implement by stakeholders within BizCo's IT culture. But by employing a strategy of targeting the primary risk of reaching critical mass early and by continuously addressing the primary concerns of the stakeholders we were able to achieve a successful implementation. (4) Finally, usability issues should never be underestimated. An incremental approach and constant involvement of users in the usability design of the application is crucial.

Going forward, the IDEA tool needs to be enhanced and more broadly deployed. The main areas for future investigation are: upgrading the user interface to a 'Wizard' workflow; adding additional architecture and interface properties to the interface descriptions; populating the application with more content and deploy more broadly throughout BizCo; conducting sample architectural analyses with the data collected to further demonstrate the utility of the system; and identifying metrics that can be collected to make a case for more widespread use of the application.

In retrospect perhaps the most profound implication of the IDEA tool is that it enables a *human network* of system (of systems) stakeholders, by making it possible to discover the right people who are associated with a set of interacting

systems with an enterprise. In an organization with a complex system of systems, putting the right people together is a significant challenge. No tool will ever replace the need for 'face time'. IDEA's function is to facilitate the face time, and to ensure that the time is well spent, with all stakeholders armed with the more accurate and up to date information.

References

[1] P. Aiken, L. Hodgson, Synergy between business process and systems reengineering, Information Systems Management (1998) 55–67.
[2] M. Ben Menachem, G. Marliss, Inventorying information technology systems: Supporting the 'Paradigm of change', IEEE Software (September/October) (2004).
[3] P. Clements, R. Kazman, M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2001.
[4] M. Fox, M. Gruninger, Enterprise modeling, AI Magazine (Fall) (1998) 109–121.
[5] D. Garlan, R. Allen, J. Ockerbloom, Architectural Mismatch, or, Why it's hard to build systems out of existing parts, in: Proceedings of the 17th International Conference on Software Engineering, April 1995.
[6] M. Hammer, J. Champy, Reengineering the Corporation: A Manifesto for Business Revolution, Harper Business, 1993.
[7] J.C. Henderson, N. Venkatraman, Strategic alignment: leveraging information technology for transforming operations, IBM Systems Journal 32 (1) (1993) 4–16.
[8] Software Asset Management, The Stationery Office, London, UK, 2003.
[9] http://www.jgraph.com/.
[10] http://java.sun.com/products/jsp/.
[11] R. Kazman, S.J. Carriere, Playing detective: Reconstructing software architecture from available evidence, Journal of Automated Software Engineering 6 (2) (1999) 107–138.
[12] W. Kettinger, J. Teng, S. Guha, Business process change: A study of methodologies, techniques, and tools, MIS Quarterly 21 (1) (1997) 55–80.
[13] M.P. Papazoglou, W.J. van den Heuvel, From business processes to cooperative information systems: An information agents perspective, in: M. Klusch (Ed.), Intelligent Information Agents, Springer-Verlag, 1999.
[14] http://struts.apache.org.
[15] http://www.touchgraph.com/.
[16] http://www.turbotax.com/.
[17] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, C. Riva, Symphony: View-driven software architecture reconstruction, in: Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, WICSA'04, IEEE Computer Society, 2004, pp. 122–134.
[18] Y. Wand, C. Woo, S. Hui, Developing business models to support information system evolution, in: Proceedings of the Ninth Workshop on Information Technologies and Systems WITS '99, pp. 137–142.
[19] J. Zachman, A framework for information systems architecture, IBM Systems Journal 26 (3) (1987).