

Squeezing the Authorization Problem Through a Shrinking Window for Requirements

Darrell Raymond
Alternative Output Inc.
Waterloo, Ontario, Canada
darrell.raymond@sympatico.ca

ABSTRACT

Information system deployment is squeezed by a shrinking commitment to requirements definition and an expanding need to determine the security requirements of such systems, due to the emphasis on internet access, online transactions, and workflow. This paper investigates the causes and effects of this squeeze. For the users of engineering information systems, the most important aspect of security is the development and maintenance of an authorization matrix. We look at some reasons why authorization is difficult and getting harder, and discuss some of the implications for requirements gathering.

Categories and Subject Descriptors

D.2.1 [Requirements/Specifications]: Elicitation methods

K.4.4 [Electronic Commerce]: Security

General Terms

Security, Requirements

Keywords

Authorization, document management, product data management

1. INTRODUCTION

Engineering and manufacturing companies need to manage drawings, documents and other data, such as bills of materials and engineering change orders about their products. In the past, companies either developed their own systems or adapted existing software¹ to manage this data. The trend today, however, is to purchase commercial software from electronic document management (EDM) or product data management (PDM) vendors.² EDM/PDM systems are applications built on a relational database that provide the following capabilities:

- Storage of large numbers of documents in an electronic 'vault'
- Change management for documents, typically through the use of check-in/check-out and revision control

¹ A typical solution was to extend a manufacturing or CAD system to handle documentation.

² There are hundreds of vendors in this marketplace, including companies such as Agile, Auto-trol, FileNet, Documentum, SDRC, Unigraphics, MatrixOne, Parametric Technologies, and Eigner + Partner.

- Storage and change management of product data and product structure (component-assembly or 'bill of material' relationships)
- Storage and management of relationships between product data and documents
- Definition and execution of structured processes or *workflow*

Although touted as off-the-shelf systems, for the most part no EDM/PDM system is functional without a significant amount of customization. From the customer's point of view, these systems are essentially applications libraries from which a usable document and product data management solution must be developed.

The security aspects of EDM/PDM systems are significant, because data management and workflow imply knowledge of who can participate in certain activities, and what kinds of operations they are allowed to perform.

Over the last several years I have assisted major firms and institutions in the planning, design, and deployment of EDM/PDM systems. These projects generally have a medium budget (\$2—\$15 million) and a medium-sized user community (1,000—10,000 users). The typical EDM/PDM project involves the following phases:

- Develop a business case for the project
- Define the system requirements
- Select a vendor
- Define the system architecture and necessary customizations
- Migrate legacy data to the new system
- Define processes for maintaining data
- Deploy the system
- Enhance the system to provide collaborative engineering, integration with MRP/ERP/e-commerce systems, and so on

Many EDM/PDM deployments have hopes of reaching an enhanced stage of functionality, but the typical EDM/PDM deployment achieves only data migration and perhaps basic change management.

The basic characteristics of EDM/PDM projects are, I believe, representative of the majority of IT projects undertaken by engineering and manufacturing companies. Thus many of the issues I raise are relevant to a wide range of contemporary IT projects.

2. THE SHRINKING REQUIREMENTS WINDOW

While it is usually considered good practice to include a requirements phase in an EDM/PDM deployment, my colleagues and I have noticed a marked decrease in appetite for requirements work. Where several years ago it was not uncommon for a company to spend four to six months doing requirements before a major system deployment, it is now more likely for a company to spend as little as four weeks. Several factors seem to contribute to this trend.

Few companies can, in practice, use internal resources to develop a formal statement of requirements. IT departments are chronically understaffed and sometimes do not have the skills for this kind of activity. There is sometimes a lack of desire on the part of IT departments to do this work, due to existing conflicts between IT and its users. IT is not motivated to collect new requirements when it can hardly meet current needs, and operational departments are sometimes skeptical of IT's sincerity and ability to deliver new functionality. Consequently, requirements definition is often outsourced.

Outsourcing, however, sometimes results in a consulting boondoggle. Unless both outsourcer and client are highly motivated, it is all too easy to generate an impressive requirements definition that is ignored. Many companies have experience with consulting engagements that produced large and costly documents but led to no deployment, or to one that was not effective. These kinds of experiences may lead companies to believe that requirements documents are not a very solid deliverable and so not worth the investment.

The suspicion that requirements are a boondoggle is buttressed by emerging notions that challenge the effectiveness of a traditional requirements process. 'Extreme programming' suggests to companies that requirements can and should be done at the same time as development. The term 'lightweight requirements' carries an implicit pejorative against the traditional process, which becomes 'heavyweight' by definition³.

A thoroughgoing requirements process is too onerous for many firms even when the work is outsourced. The overhead of learning the requirements lexicon, getting used to diagramming techniques, and of purchasing and learning computer aids for developing requirements (such as UML modeling tools) is too much for many organizations. More importantly, companies cannot commit much of the time of senior managers to this kind of work. Some companies suffer what I call *requirements fatigue*. The first time requirements definition is done, a company may find it interesting and useful, but many companies have had to endure multiple occurrences of requirements or requirements-like interviewing for ISO certification, MRP deployment, CAD systems, project management software, customer relationship management, collabora-

³ The actual differences between lightweight and traditional requirements are not at issue here. To those looking for a reason not to write a formal requirements, the *idea* of lightweight requirements is sufficient. Promoting the latest innovation in software development is usually attractive, particularly when it seems to imply that things can be done with less rigor, less time, and less money.

tion tools, HR, enterprise resource planning (ERP), and so on. Every new deployment seems to start with several weeks of interviewing, self-analysis, process mapping, and diagramming, often with the same managers involved in roughly the same interviewing⁴.

An interesting (and I think largely unperceived) trend is the shift in requirements work from projects at individual companies to 'standardized requirements' done by cross-company organizations. A 'standard' is normally a formalization of good existing practice, but what some standards bodies engage in is the creation of models of what they think good practice ought to be like, in the hopes that vendors will be encouraged to build systems that support (their definition of) good practice, and users will thereby employ good practice. In effect, this is a requirements definition process. Because standards bodies are typically drawn from across organizations, the statement of requirements is also cross-community. Whether this process is effective or not, it can become another reason why a company doesn't need to develop a requirements definition for its own deployment. Consequently we now sometimes hear from clients that 'we want what everyone else has' or 'so long as it's compliant with OMG 12345, we will accept it'.

Not only do companies have requirements fatigue, they are also tired of vendor selection. Selecting a vendor can take months if it is not aggressively pursued, and without careful preparation, the selection process may be little better than random. Companies suspect that this phase of a project is unprofitable, and so seek to shorten or eliminate it. In the extreme case, requirements definition becomes the output of product selection, rather than its driver. More than once I have been told, 'whatever product X does, that is how we will work'⁵. Another common view is that 'all the products in this area are essentially the same, so let's just pick one and get going'.

Requirements definition is always a political process. People have significant disagreements about whether a system is needed, what it should do when and if it is deployed, and which vendor is the best one. A successful deployment may be a common good, but it is typically viewed as a particular evil by at least one person or group—whose main objection to the process may simply be that they are not leading it. The longer and more public the requirements definition process, the more chance for the project to become derailed or for resistance to the project to crystallize. Thus, there are political reasons to keep the requirements definition phase short.

⁴ If process maps and operating procedures already well documented, up-to-date, and conveniently to hand, then it is likely that some kind of requirements or BPR process has recently taken place. As every requirements gatherer knows, however, these documents are often only the veneer covering the real business processes at work in the company. In a slimmed-down requirements process, it is very difficult, as well as politically unacceptable, to spend time verifying the existing process maps and procedures.

⁵ This is particularly common with ERP systems. A major attraction of ERP systems is that they do 'everything' in a single integrated package. Thus, defining requirements appears to be superfluous—whatever the system does, that's what we want.

The shrinking window for requirements definition means that the focus of the requirements phase has shifted from determining requirements to determining differentiating factors between vendors⁶. The basic requirements question ‘what do we need?’ has been replaced by the query ‘tell me how the vendors differ, and then I will tell you which one meets my needs.’

3. THE IMPACT ON SECURITY REQUIREMENTS

The requirements process, never very highly valued, is apparently going through a downturn in perceived importance. It should not be surprising, then, that the time and effort spent on security requirements is being squeezed as well. This squeeze aggravates the many special problems that exist in collecting security requirements.

3.1 Lack of Knowledge of Security

The first and simplest problem in collecting security requirements is that business process owners are generally uninformed about security. Business process owners are very well informed about their data and work activities, have an acute understanding of what improvements need to be made to their processes, and also understand that tradeoffs need to be made in design; thus, requirements interviews on these topics are focused and fruitful. This is not the case for security requirements. Most engineers and managers need to be guided to the fact that authentication, authorization, privacy, and data integrity are the basic goals of a secure system. There is generally little understanding of the mechanisms required to reach these goals; typically, users have the vague notion that improved security is somehow mainly due to the use of digital certificates. Virtually none of our clients have thought it necessary to conduct a risk analysis to determine just what level of security is required by their organization, or what cost is involved in providing it.

There are several reasons why knowledge is lacking. First, security is the not the primary responsibility or interest of most people. Second, security is a very complex area that is changing rapidly; hence, it is hard to keep up with security issues. Third, learning from past experience is sometimes limited by the strong tendency to minimize knowledge of security breaches [12].

In general, security is viewed as a system characteristic that is purchased, rather than one that derives from a mixture of mechanism and policy. Requirements texts tends to support this view, rating security as a ‘non-functional’ requirement similar to performance [9][15].

The lack of knowledge about security means that an educational process needs to be conducted before security requirements can be gathered. A squeezed window for requirements has no time for such an educational process.

3.2 Security is a Negative Goal

It is more difficult to collect security requirements than some other requirements because security is a *negative goal*—that is, the goal of avoiding a certain kind of result, rather than the achievement of a result [5]. In this sense, security is like system

reliability (also viewed by requirements texts as a non-functional requirement).

Negative results are hard to measure. A secure system stops or deters security breaches, but how does one measure security breaches that did not occur? A firewall log certainly captures information about attempts to access a system, but not every one of the disallowed accesses is a non-breach in security, nor are all the security breaches captured. The costs of a secure system are paid whether there are attempts to breach its security or not.

Negative results require an investment that few want to pay. An improvement in the security of a system is generally not enjoyed by users—their workstations are locked down⁷, they must use lengthy passwords that have to be frequently changed, and their access to and use of information is logged. These measures directly reduce the ease that people feel in doing their work, and in return they gain what appears to them to be a nebulous benefit. Contrast this with an investment in another ‘non-functional’ requirement, system performance: an investment in improved system performance often has an immediate and noticeable payoff that can be measured (for example, logs may show that users have faster access to documents and hence make better and more frequent use of information).

Negative goals are poorly rewarded. A system administrator who provides 99% uptime will not be congratulated every day that the system is up, but will be under severe pressure when the system is down. Similarly, a security administrator will not receive daily kudos for doing a good job, but will be the one on the spot when systems are compromised. Paradoxically, the better that systems perform, the more people come to view this performance as typical—and one doesn’t reward typical behavior. Hence, the better one achieves a negative goal, the more likely that the only feedback one gets on performance is negative.

Negative goals are not attractive to people, and so they have less interest in being associated with them. Consequently, fewer resources will be invested in determining the requirements for negative goals.

3.3 The Key Differentiators Approach Doesn’t Work

Earlier it was noted that the requirements process is sometimes whittled down to a ‘key differentiators’ approach. In this approach, a full requirements document is not produced; instead, only the key differentiating requirements between various products are examined. This approach to requirements is founded on the following assumptions:

- The company’s requirement for standard functionality is not significantly differ from that of other companies in the same marketplace
- All vendors are (thought to be) more or less equal in providing standard functionality
- Attention should be paid to the requirements that are of most interest to the company or that will involve the most cost-savings or increase in productivity

⁶ We might call this ‘lightweight vendor selection’.

⁷ Both physically—the machine’s box is locked—and logically—the software configuration cannot be altered.

Typical key differentiators in the EDM/PDM field include such things as:

- The vendor provides support for a specific ERP system
- The vendor provides support for a specific CAD package
- The vendor's product is used by other companies in our marketplace
- The vendor provides a specific kind of functionality that is needed
- The vendor's product employs technology that is well-understood by our staff⁸

Competently done, the key differentiators approach is effective in slimming down the requirements definition phase. But the key differentiators approach is not a good method for identifying security requirements for EDM/PDM. There are several technical reasons why this is the case:

- Most of the vendors provide roughly the same set of authorization and authentication capabilities
- There are no analyses comparing the security of the various vendor products
- There exists no database of past history about vendor security
- There are no commonly used benchmarks for testing a vendor's product

These technical issues reduce our ability to differentiate products on security-related capabilities.

Even if we could identify useful features distinguishing systems, it must always be borne in mind that a secure system results from a thorough understanding and application of both policy and mechanism—and of the two, policy is probably more important. Vendor products at the moment provide little help with policy.

3.4 Division of Responsibility for Security

A critical problem is that in most companies, the responsibility for security is divided among three or more groups. In theory, the division works this way:

- The IT department is responsible for the security mechanisms, including recommending, acquiring, and deploying firewalls, authentication systems, and other technology.
- The legal department (or the executives of the company) are responsible for setting policy: for defining categories of intellectual property, for defining the categories of users, and rules about who can operate on data.
- The business process owners are responsible for operating the mechanism according to the policy.

In practice the division of responsibility is more skewed. Generally only the IT department knows what issues constitute security issues, and only the IT department (if anyone) comprehends the

⁸ Or in some cases, uses technology that enables us to attract staff. One of our clients rated Java-based systems very highly since it was thought that it would be much easier to attract new personnel to their (rather remote) location by advertising a position involving Java training rather than one involving training in C.

mechanisms that can be used to address the issues. Thus, the IT department defines both the problems and the limits to the problems that can be solved. This weight of knowledge means that the security policy is generally under the control of the IT group, and only approved by the legal department.

Dividing the responsibility for security greatly lengthens the process of determining security requirements. Ask a business process owner about security classifications, and you are referred to IT department policies. The IT department's security group's policy is usually 'under development', and the legal department is currently reviewing it. If there have been no recent security breaches, then there is little urgency felt to complete the policy review. Paradoxically, if there *has* been a recent security breach, then policy review is considered essential but will be slowed down because no one wants to make a mistake at this point. Thus, there is often no firm policy that can be counted on to remain unchanged during the system's design and deployment.

The division of responsibility implies a division of liability, and it is typical for each of the groups mentioned above to exploit the situation in some way to limit their own liability for security breaches. The legal department is no rush to approve policy, because it understands (perhaps more clearly than the other groups) that approving certain kinds of behavior implies liability on their part if those behaviors result in security breaches [1]. Thus they tend to circle the wagons around the company's intellectual property, and start from the position of need-to-know⁹. IT will have deployed certain kinds of authentication and protection, based on what is considered good practice, and it wants these systems to constitute security by definition. Since IT wants to minimize administration costs, it generally mandates simple rules that apply to everyone, regardless of business need (e.g., 'no unauthorized software of any kind on any workstation'). Thus IT limits its liability by constructing an artificially restricted situation which it feels it can control. In order to limit their own liability, business process owners obey the letter of the security policy but often violate the spirit—hence, people will resign themselves to long passwords but express their resentment at the rule by writing the long password on a sticky note pasted on the bezel of the monitor. The consequences for actual security are obvious.

Buck-passing can be a nested process. It is not uncommon for the local IT department to be waiting for the corporate IT department to set general IT security policies. Meanwhile, corporate IT is waiting for corporate R&D to evaluate systems for doing authentication before it sets policy. The legal department, on the other hand, may know or expect a corporate merger or other change of management in the near future—hence, decisions about security policy will be put off until the new administration is in place.

⁹ One company developed a simple Visual Basic application to submit an electronic request for a drawing package. They were told by the legal department that they had to remove this application since otherwise 'any contractor could just request a drawing package'. When it was pointed out to them that in the course of normal business, contractors had for years been able to simply walk down to the print shop and fill out a paper form to request drawings that they needed, the developers were told, 'well, you should stop allowing that'.

The division of responsibility makes it difficult to decide whom to blame when a security breach occurs. Was it that the legal department wasn't clear enough about the policy, and so managers couldn't enforce it? Or was it that the IT group didn't choose the right mechanism? Or did the IT group not recommend the right set of options on which a policy could be based? In most cases, the business process owner is the one most directly involved in the security breach, and so they take the blame. As in other normal accidents, it is all too easy to put the mistake down to 'operator error', even when the operator is working with a system that encourages error [9].

The main effect of division of responsibility for security on requirements gathering is that it further complicates and delays the organization's definition and understanding of its security needs.

To this point, we have examined general characteristics of the problem of determining security requirements. In the next section we will discuss the issues surrounding a specific security requirement: the problem of determining authorization.

3.5 The Authorization Problem

The need for authentication, data integrity and confidentiality in an EDM/PDM system is roughly similar to the same needs in any moderately sized general-purpose computing facility; hence, system managers are on familiar ground in addressing these needs. The requirement for authorization, however, far exceeds the typical case.

We shall define an *authorization specification* as a three-dimensional matrix of people, objects, and operations. If the value of (x, y, z) is 1, then person x can apply operation z to object y . The authorization problem is the problem of populating the authorization matrix so that the following characteristics hold:

- We have reasonable guarantees that the matrix is correctly and completely populated
- The matrix can be maintained in a reasonably efficient manner

In a typical EDM/PDM deployment, this matrix is potentially very large. The number of objects y can easily be in the millions. A typical manufacturing firm maintains data on 10^5 parts and 10^6 component-assembly relationships. Each part may have several associated documents (specification, geometric drawing, parts list, schematic, gerber file, application drawing, etc.). Parts typically exist in multiple revisions, each of which will have a drawing and usually one or more engineering change orders. Some companies also maintain information on serialized parts—thus, y is potentially as large as the number of products made by the company.

The number of operations z is larger than the standard 3 (read-write-execute) of common file systems. A typical EDM/PDM system may support as many as 30 different operations, including:

- Create—generate a new object
- Read—read the attributes of an object
- View—read the files associated with an object
- Modify—edit the attributes of an object
- Revise—generate a revision of an object
- Print—allow the files associated with the object to be printed

- Check-out (check-in)—export (import) files associated with the object
- Delete—delete an object
- Lock (unlock)—disallow (allow) file checkout
- Promote (demote)—move the object to the next (previous) stage in its lifecycle
- Grant—allow other users privileges
- Set privileges—change any of the authorizations for an object
- Link (unlink)—create (delete) a certain kind of relationship to another object
- Execute—execute one or more programs associated with an object

Although the number of users in an EDM/PDM system is moderate, each of the users can belong to multiple groups and can participate in multiple roles. A 'role' or a 'group' is essentially just a set of persons that is identifiable for some business purpose. Most EDM/PDM systems treat a group or a role as a substitute for a person. In other words, x can be set-valued, with sets chosen from the set of persons.¹⁰

Workflow adds another dimension to the authorization matrix, that of state: person x can apply operation z to object y only when that object is in state a . The set of states is typically small, including such things as 'draft', 'in work', 'review', 'release', 'superseded', and 'obsolete'. It is not unusual to have different sets of states for different kinds of objects; thus, parts and drawings will be subject to formal release control, but documentation may be handled more informally.

In a typical case, then, the authorization matrix is on the order of 10^{12} entries. This makes the authorization problem quite significant. In the past, the authorization problem has been tractable because of the use of simple rules that dramatically reduce the size of the authorization matrix, to wit:

- Everyone who works for us has read access to everything.
- No outsider has any access at all.
- All other operations are permitted only to a small configuration group that does all data entry.

Under these rules, y is reduced to 1 (because all objects are treated the same); z is reduced to 2 (read and all other operations); a is reduced to 1 (there is no state dependency) and x is reduced to 3 (internal, public, and CM).

Unfortunately, this simple scenario is increasingly inapplicable.

3.5.1 The new economic climate

Companies are highly interested in participating in the 'new economy', joint ventures, downsizing, e-commerce, Web portals, and other trends that directly increase the size of the authorization matrix.

The simplification 'everyone who works for us has read access to everything' is disappearing because the notion of 'us' is becoming more complex. Manufacturing companies are increasingly less

¹⁰ Some EDM/PDM systems allow full boolean combinations of person and role sets.

vertically integrated, and are moving towards outsourcing, collaboration and joint ventures as their principal mode of business. This change increases the complexity of the authorization problem.

Outsourcers should be shown only the data they need to do their job, but not other data. It is important to limit outsourcer access to information for many reasons: one is that they may also work for competitors (hence, could leak proprietary information); another is that access to too much information enables them to bid more effectively on new work (hence, they may cost more to hire than they would otherwise). Outsourcers need more than just read access to data, because they are often actively generating new engineering and manufacturing information.¹¹ Workflow becomes more important because outsourcers may not be physically on-site, or may be in different time zones.

Collaborative ventures are undertaken with a variety of firms, including current and future competitors. While collaborators should have free information access to data related to the project on which they collaborate¹², it is very important to ensure that competitors do not gain access to other projects. As with outsourcers, collaborators need to participate in change to information, not just have read access to it.

Joint ventures are semi-autonomous organizations that are formed by otherwise competitive firms. A joint venture may have its own IT department with its own notions about how to do authentication and authorization; for example, it may see less need for authorization than do its parent companies, because the joint venture's information is wholly open to either parent.

Customers are beginning to expect online access to information about products and their as-sold (and even in some cases, as-installed and as-maintained) configurations. Customers of course should only have access to their own configurations, and not those of other customers. Customers are not monolithic entities; engineering personnel at a given customer site should have different kinds of access than accounts payable personnel, for example. As an additional complexity, the rapid pace of mergers, consolidations, and divestitures means that what counts as a single customer changes over time.

These factors increase the size and complexity of x and y as well as increase their rate of change. It is not unusual for a merger, a joint venture, a collaboration, or some other significant corporate event to occur during the deployment of an EDM/PDM system.

Workflow will soon not be a matter of choice, but a matter of survival. Configuration management departments that once had 40 people to manage engineering change are now reduced to 4. Shuffling paper change forms is no longer a viable option—particularly when so much of the change is happening 'outside' the firm. This means significant increases to z and a .

The demands of the new economic climate simultaneously increase the importance and the difficulty of the authorization problem.

¹¹ In some cases, control of engineering changes may be outsourced.

¹² Although almost certainly not all of it; for example, a company will keep its margin and cost information hidden from its collaborators.

3.5.2 Categorization and role-based authorization

One way to ameliorate the authorization problem is through the use of categories and roles. A category is a set of documents, and a role is a set of people. Generally we allow documents to belong to more than one category, and people to belong to more than one role. The authorization matrix is expressed as (X, Y, z) , where X is the set of roles and Y is a set of categories.

The use of roles and categories reduces the size of the authorization matrix (and hence, makes for a more tractable authorization problem) when it is the case that $|X| \ll |x|$ and $|Y| \ll |y|$. Role-based authorization is founded on this assumption [2].

From a database design perspective, role-based authorization is a normalization that reduces the redundancy in the authorization matrix—instead of specifying independently that individuals $x_1, x_2, x_3, \dots, x_n$ can each read each document $d_1, d_2, d_3, \dots, d_n$, we simply define two sets and establish a single authorization constraint between them. As for database normalization, authorization normalization has the attractive property that updates are simplified—we can add a single user to the set X and gain access to all the documents Y ; hence, a set-based constraint has a multiplicative effect under update. However, there is a corresponding downside: error in defining the sets is a multiplicative defect. If person x_k is mistakenly assigned to a role, then that person gains much greater access than would be the case if a single error occurred in a pairwise constraint system. Thus, role-based authorization can reduce the cost of populating the authorization matrix, but may increase the requirement for validating the matrix.

Mistakes in assigning persons to roles or documents to categories are not unknown. The meaning and use of security classification is fundamental to security, but organizations do a remarkably poor job of establishing workable classifications. Consider the ease with which managers might conflate such categories as 'unclassified controlled nuclear information' and 'sensitive but unclassified nuclear information' [8]. Or consider the distinction between 'classifying' data and 'categorizing' it: 'PARD: Protect As Restricted Data' is not a classification level per se but 'a handling method for computer-generated numerical data or related information, which is not readily recognized as classified or unclassified because of the high volume of output and low density of potentially classified data,' and ranks between unclassified and confidential, the lowest level of classification' [12]. In general, categorization for security purposes is considered one of the most problematic areas of governmental security [14]. It is much easier to define categories than it is to ensure that managers apply them consistently and correctly. I know of at least one Fortune 50 company in which there are only four possible security classifications for documents, but whose managers consistently gave me different and incorrect definitions of those classifications (the correct definition came from documentation). Categorization for authorization is subject to many of the same problems that arise in other categorization contexts.

Role-based authorization is less attractive in situations where the role doesn't completely define object access. Consider for example subcontractors: 'subcontractor' appears to be a role, and it is certainly true that there is some information that no subcontractor should see, but on the other hand, each subcontractor needs read and modify access on distinctly different sets of data. Hence, in addition to requiring a role definition, it appears that we still need individual specifications for each subcontractor. In such circum-

stances, role-based authorization may not reduce the size of the authorization matrix.

3.5.3 Discussion

Although most of the attention in security goes to systems for authentication, many of the most notorious security breaches are really problems in authorization. The notorious case of Wen Ho Lee at Los Alamos is a recent example. Lee was not charged with an authentication violation (that is, pretending to be someone he was not, or attempting to gain access to information to which he did not have access), but with transferring material from a secure computer to a non-secure one—that is, with carrying out an unauthorized operation [12]. There were, apparently, no authorization mechanisms in place to guard this transfer.

Authorization is a security function that is of primary interest to business process owners. Deciding who gets access to what is an essential part of the functionality of an EDM/PDM system. The best indicator of this fact is that access to information is a saleable commodity, and for some companies a significant source of revenue. But authorization does not get the same publicity that is given to cryptography, digital certificates, and other more glamorous areas of security research, and so there is little information available at a general level to assist companies in solving their authorization issues. EDM/PDM systems are, in many cases, the most elaborate example of the authorization problem that companies have faced, and their manual precursors do not provide a very useful guide to how to manage automatic authorization.¹³

A desirable authorization system is one that can be applied to a wide spectrum of information resources. One barrier to a wide-spectrum authorization system is the lack of a standard notion of *things*. All systems share the same notion of persons, expecting them to be unique and stable entities, and so authentication systems at least have a common base to work with (facilitating, for example, the idea of ‘single sign-on’).¹⁴ But authorization deals with computer objects, and what constitutes an ‘object’ depends largely on which software package you use: operating systems deal with a universe of files, relational databases deal with a universe of tables and rows, ORBs deal with a universe of CORBA-compliant objects, Web servers deal with a universe of URLs, and EDM/PDM systems control many of these elements and add workflow as well. The ‘impedance mismatch’ between these systems makes it difficult to have an authorization mechanism that spans all of them.

One way to reduce impedance mismatch is to funnel all data through a single namespace. This is a fairly old technique: think for example of mapping objects such as sockets into a file system namespace. Today more attention is given to the URL namespace. Netegrity and Dascom are two companies with products that provide access control to a URL namespace; if you can give all your objects URL names, then authorization for these objects can be defined through Netegrity or Dascom. This kind of approach is

¹³ In most cases, the manual authorization mechanism is based entirely on personal knowledge: the person authorizing an operation knows both the object and the requestor and makes an independent one-time decision about that specific operation.

¹⁴ Looming on the horizon are issues such as how to separate human clones, how to safeguard bio-identification techniques against advanced surgery and genetic techniques, and so on.

useful for data access, but is less satisfactory for workflow and other operations that require update. Another namespace possibility is an object-based namespace, possibly with something like the CORBA security standard [1]. There are two problems with this specific solution, however: most organizations aren’t CORBA-centric (so the solution isn’t wide spectrum); and the standard just defines the parameters of a security mechanism—policy must still be developed.

From a system management point of view, it seems reasonable to have a single centralized authorization matrix, because this reduces administrative overhead. From a security point of view, however, there is an argument for *not* centralizing authorization. A centralized authorization system suffers the problems of any centralized system: it is not robust to failure and error. Multiple redundancy is a common approach to increasing robustness; in other authorization contexts, the classical form of redundancy is the *two-man* rule. An ICBM launch, for example, requires the coordinated activities of a number of distinct people; this redundancy (in people) tends to make the system as a whole more robust to failure in any single component (such as, for example, error or malicious intent on the part of one of the persons). Given the examples of authorization failure mentioned earlier in this paper, it may be useful to think about deploying redundant authorization as much to overcome human error as to overcome system error.

It is worth rethinking the distinction between authorization and authentication. One proposal is for merged authentication and authorization systems, by incorporating authorization information in a certificate used for authentication [11]. One problem with this approach is that authorization is trending towards a time-sensitive, rule-based property, rather than an unchanging attribute of a single individual. An alternative approach that works in some contexts is to relinquish the need to authenticate a person’s identity, and instead authenticate their access permissions, as is done in trust management systems. This kind of approach is unknown to most vendors and consumers of EDM/PDM products today.

3.6 Other security issues specific to EDM/PDM

Looking beyond the authorization issue, many EDM/PDM systems have security holes and problems of various kinds. In general, these problems spring from two characteristics:

- EDM/PDM systems are closed source efforts that implement their own authorization tools; hence, they have all the problems of systems that attempt to achieve security through obscurity [6].
- As noted earlier, EDM/PDM systems are more like application libraries than turn-key solutions. Thus, not only must we worry about security in the basic product, we must worry about security holes introduced through customization and integration. There are a variety of places where problems can be introduced.

3.6.1 Full text search

One typical hole is found in systems that have recently had a search engine ‘bolted on the side’ to provide Internet-like search capability. When the search engine is not integrated with the authorization model, it is sometimes possible to know that data

exists and some of the content of data even if one is not officially authorized to access the data.¹⁵

3.6.2 *Printing and scanning devices*

EDM/PDM systems generally do not come with software for batch printing, application of watermarks and banners, and other output-device-specific software that is essential to usability. This software is typically:

- customized or locally developed; hence its security is suspect
- arcane; hence, it is difficult to evaluate its impact on security
- produced late in the deployment cycle, so there is pressure to get the software 'out the door'; hence, security is put on hold in order to complete the project
- full of interactions with intelligent devices that can be points of entry for a determined hacker.

3.6.3 *MRP/ERP systems*

PDM systems maintain information that is also needed in production systems, particularly bills of materials. The transfer of PDM bills to MRP/ERP systems is usually a task requiring third-party or custom software. Since this data transfer usually updates data, it requires substantial authorization privileges. The data transfer is often unprotected against spoofing.

3.6.4 *Interface software*

Almost no company likes the out-of-the-box interface of EDM/PDM systems, and most undertake extensive and elaborate development of an interface that is typically Web-based. The security of the interface (which includes the transfer of authentication information) is subject to the quality of the company's developers or the outsourcers they hire to do the job. Systems in the recent past have been known to transfer authentication information in the clear or store transaction IDs in cookies and other vulnerable locations.

3.6.5 *Intelligent objects*

Documents managed in an EDM/PDM system often have authorization information that exists outside the EDM/PDM's authorization system. Adobe's Portable Document Format, for example, supports password-protected authorization controls on certain operations such as modification, printing, content selection, and annotation. These authorizations are controlled by passwords that are set on the PDF file itself, rather than through an authorization layer in a system environment. More recently, systems have been designed which involve online authorization requests from distributed documents; examples of such systems include IBM's Cryptolopes and Xerox's ContentGuard, and there are also new systems proposed to control access to electronic entertainment media [4]. A comprehensive security policy must decide how these kinds of technologies should be used in concert with an EDM/PDM system security. No current EDM/PDM system integrates these kinds of capabilities into its overall security framework.

4. RECOMMENDATIONS

The following represents not an ideal solution to the difficult problems of collecting security requirements, but our best current approach.

The division of responsibility for security is unlikely to disappear. Since business process owners will become the operators of the security system (and hence the most likely to be blamed for breaches of security), they should act as if they bear the whole responsibility.

Categorization is more fraught with danger than it appears. Experience shows that even relatively small categorization schemes can be misinterpreted and misapplied, while complex schemes are often subverted, thus invalidating any reasoning that might be done based on the assumption of proper use of categories. The simplest possible categorization should be used, and steps must be taken to audit the use of the categorization scheme.

Most EDM/PDM systems have some ability to construct and manage an authorization matrix, but little or no functionality for testing the matrix. This makes it difficult to reason about the effectiveness of the system. Organizations thus need to develop their own test plans and methods for auditing their authorization policies.

It is essential that business process owners become knowledgeable about authorization. Authentication, privacy, and data integrity are important, but are more easily managed independently by the IT department. Business process owners must understand that authorization is a functional requirement: the definition of access and change to the intellectual property of the business. As such, it is primarily a business issue, not a systems issue. It must also be understood that a robust authorization system does not result from simply purchasing an off-the-shelf product—it requires significant policy development and commitment of resources for long-term management and auditing.

EDM/PDM systems tend towards a tightly coupled, highly complex environment. According to normal accident theory, in such systems the possibility of security breaches cannot be eliminated. Thus, the organization must understand that failures will occur, no matter how well-trained the operators [12]. This implies that the organization needs to develop a realistic understanding of the risks involved in electronic management of information, and therefore needs to explicitly evaluate risk and determine the costs involved in risk-taking. What needs to be developed, in the end, is a business case for security.

5. CONCLUSIONS

In the world of medium-sized software projects, the need for requirements definition is expanding, but the appetite for doing requirements is contracting. This is as true of security requirements as of any other kind. The problems enumerated in this paper could, by themselves, easily consume the entire time that is typically available for requirements definition—but of course the time must be spent on many other aspects of the system as well. Thus we should not be surprised if security requirements are often poorly captured and poorly implemented.

The main security issue that business process owners should be concerned with is authorization. They need to specify their authorization requirements in the most straightforward manner, and they need a method for validating their specification as it

¹⁵ Thus, I could use the search engine to find my own name in the file `DOWNSIZE.DOC`, and know that this document is owned by my manager, all without having read access to the document.

changes. The technology for deploying an authorization matrix exists; what is needed now are practical guidelines for its effective use.

6. ABOUT THE AUTHOR

Darrell Raymond assists companies in acquiring and deploying systems for document and product data management. He has done requirements work for many companies, including Amtrak, Cummins Engine, Eastman Kodak, GE Nuclear, Los Alamos National Laboratory, Oxford University Press, and Siemens Electrocom. Most of this work was conducted through The Gateway Group, an independent consultancy that specializes in document and product data management. Raymond holds a Ph.D. in Computer Science from the University of Waterloo and is presently an adjunct assistant professor in UW's Department of Computer Science.

7. REFERENCES

- [1] Anderson, Ross J., *Liability and Computer Security: Nine Principles*, ESORICS 1994.
- [2] Barkley, John F., Cincotta, Anthony V., Ferraiolo, David F., Gavrilla, Serban, Kuhn, D. Richard, *Role Based Access Control for the World Wide Web*, NIST, April 8, 1997.
- [3] CORBA Security Service Specification V1.2 CORBA Services Specification (December 1998) Chapter 15.
- [4] Content Protection System Architecture: A Comprehensive Framework for Content Protection, Intel Corporation, International Business Machines Corporation, Matsushita Electric Industrial Co., Ltd., Toshiba Corporation, February 17, 2000.
- [5] Dörner, Dietrich, *The Logic of Failure: Recognizing and Avoiding Problems in Complex Situations*, Perseus Press, 1996.
- [6] Garfinkel, Simson and Spafford, Gene, *Practical Unix and Internet Security*, O'Reilly & Associates, 1996.
- [7] Hughes, Larry J. *Actually Useful Internet Security Techniques*, New Riders Publishing, 1995.
- [8] Science at its Best, Security at its Worst: A Report on Security Problems at the US Department of Energy, Report of the Special Investigative Panel of the President's Foreign Intelligence Advisory Board, June 1999.
- [9] Perrow, Charles, *Normal Accidents: Living With High-Risk Technologies*, Princeton University Press, 1999.
- [10] Robertson, Suzanne and James, *Mastering the Requirements Process*, Addison-Wesley, 1999.
- [11] Rubin, Aviel D., Geer, Rubin, Ranum, Marcus J. *Web Security Sourcebook*, John Wiley & Sons (1997) p. 317.
- [12] Sagan, Scott D. *The Limits of Safety: Organizations, Accidents, and Nuclear Weapons*, Princeton University Press, 1993.
- [13] Schwartz, Stephen I. Scientist, Fisherman, Gardener...Spy? *Bulletin of the Atomic Scientists* 56(6) (November—December 2000) pp. 24-30.
- [14] Smith, Jeffrey H. *Redefining Security: A Report to the Secretary of Defense and the Director of Central Intelligence*, Joint Security Commission, February 28, 1994.
- [15] Wiegers, Karl E. *Software Requirements*, Microsoft Press 1999.