

Knowledge Capture and Collaboration Systems

An Automae White Paper

<http://www.automae.com>

Information is the lifeblood of every development program and *access to the right information can determine when a development team is able to complete the job and when they cannot*. More to the point, it determines whether individuals on the team can do their work. In fact, during development of the system, one of the greatest causes for an increase in budget requirements and schedule slippage is the inability to find information. Typically the problem is not even due to the information being unavailable. Rather, required information has been stored somewhere by someone on the development team, but other members not intimately connected with that data do not know where to find it when they need it. Other times, information has been retained by the developers themselves and stored away in a personal directory, only to be unavailable when that person is unavailable. A worst case scenario with conventional program execution is that a team member leaves the program (or company) and information necessary for the program is actually lost because it was never properly transferred to a location where it could be stored or was stored in some obscure location on his computer or personal directory. Since nobody else knows where to find the information or what it looks like when they do find it, it is lost to the program.

In this paper, we will discuss the basic design of a knowledge capture system capable of providing tools for avoiding these problems on both large and small programs, thus increasing the efficiency of the programs by up to 40%.¹ We will discuss why these tools work and the fundamental minimal set necessary for these to be effective in aiding a program's productivity. Finally, we will discuss some basics that can be used by program managers to implement such a program.

Introduction

Typical project management methods have been successful in overseeing the creation and deployment of some of the most important and significant devices in human history such as the Space Shuttle and International Space Station, the B2 Bomber, United States Aircraft Carriers, and the Golden Gate Bridge. However, these same techniques also have a long track record of overseeing programs that under perform and over run their budgets. In many cases, program management simply does not employ tools that can make it successful. Instead, *programs often function as a political establishment with subteams employing a silo mentality* that dominates the operation of the program. The program ramps up quickly, as a tangible outward sign to the customer that the contractor is capable of accomplishing all that it must. Sub-teams receive tasks to perform according to specific capability sets and work begins. In due time, requirements are developed, but while the requirements have been under development, work has been done on every subsystem and organizational subteam, since most team members want to contribute to the program.

Problems develop when these teams, which have not been working together, start receiving formal subsystem requirements and begin to integrate their work with that of other teams. Each sub-team attempts to integrate with minimal impact to their own completed work and politics ensue. The technical health of the program suffers.

To make matters far worse, the very information developed on the program is also stored in silo's maintained by the subteams themselves. Finding information stored by another subteam is often virtually impossible to find when the user is not familiar with or do not have access to team-centric data repositories. At some point, every team establishes a central configuration management system so that they can maintain and provide program deliverables to the customer as required by the program milestones. Documents stored in the repository represent the final products at specific points in the program, but generally do not include the development of the work or the reasons behind specific designs and implementations. As a result, knowledge is invariably lost during the development of the program.

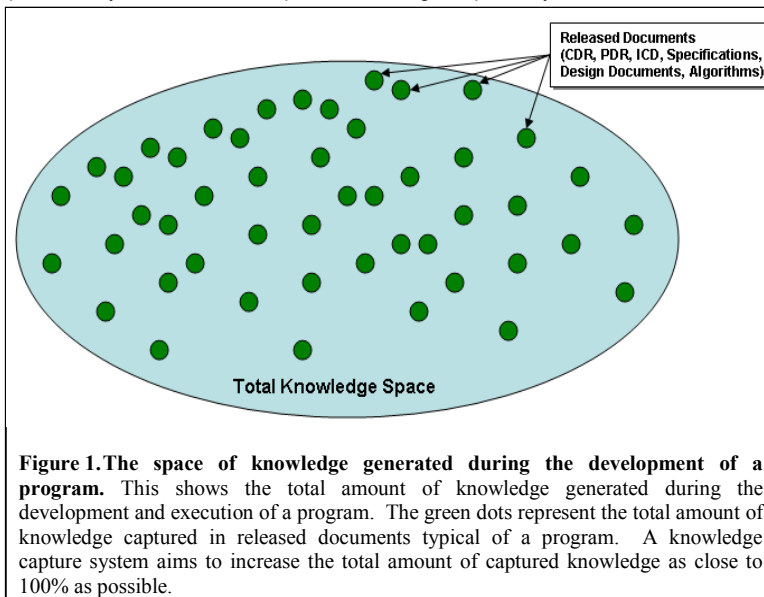
In fact, in a typical program, the majority of data pertinent to the system design is never captured during the lifecycle of the project. This results directly from the two-step development-document process. Most teams develop their product and deliverables and then try to capture it later. Unfortunately since the process of documenting the work is rarely completed and if it is, the result is a minimal set. Many times, collected data is never viewed again. Clearly, this management style creates makes the program dependent on specific individuals who ultimately develop the information and may take it with them (out of the program) once they² move off the project.

¹ Estimated.

² See, for example, Leveson, Nancy G., "Intent Specifications: An Approach to Building Human-Centered Specifications," IEEE Transactions of Software Engineering, Vol. 26, No. 1, January 2000, pp. 15-35.

To further frustrate development efforts, since much of the information is not captured in the official configuration management system (data repository), much time is spent by team members during development of the system and its subsequent maintenance searching for information that should be readily available when it is needed, either through repository search tools or through personal contact.

While the problem is fundamental to the management culture, some important changes in the capture and handling of information can alleviate many of the problems encountered by members of such a team in developing their own portions of the final product. In fact, careful planning and development of such technology will properly connect the productivity tools into a complete knowledge capture system that allows users to develop a living, growing accessible instantaneous snapshot of the state of almost all information developed on a program.



The Basic Components

Proper knowledge handling serves the purpose of providing the right information to the right people when it is needed. Put another way, it provides the right information to people when they request it easily and quickly. This means that the information they require must have been captured at some time prior to its need and that is available (easy to locate and accessible) to the team member that needs it.

Immediately, we are faced with a limitation that arises from basic human nature. A program is paid to develop a product, not document it and the need to document the work immediately creates a conflict in program directives. In order to document the project, a substantial effort needs to be taken to record the design, effort, and thought processes. Since most programs only officially deliver specific documents, all other information is documented haphazardly and is not controlled or integrated in any systematic fashion.

Due to the nature of the tasks, changing the status quo can only be done if documentation and task execution are joined. These are two completely different tasks and both are quite substantial in size and scope. Documentation does not refer only to the development of deliverables, it refers to the capture of the totality of the decisions, process, and deliverables of the development process. This includes the discussions that occur on email, the hallway chatter, the forks in the road and the reasons for taking one path as opposed to the other.

The simplest way around natural resistance to documentation is to take the two tasks and merge them into one. Indeed, doing documentation while doing work is not new. Many initiatives have started and met with varied success. As an example, software was developed on a program in the late 1990's at an aerospace company using comments throughout the code in a specific form that allowed an automated document generation program to grab the comments and develop the supporting developers manuals. This was a well-developed technology, but even at that aerospace company, only one or two people picked it up and used it. The problem was that it was attempting to *hide* the development / document dual task dilemma, but only succeeded in moving it. Nobody was fooled. The documentation was still not done.

In order to merge development and documentation, documentation itself needs to disappear as a separate task. Indeed, the development process produces huge amounts of data and it is this data that needs to be captured. While the previous example did not get the two tasks merged, it did do one thing very well – it collocated the documentation and the design. Going a bit further, the solution becomes clear. ***Make the design the documentation itself.*** That is, the design and all of the efforts taken to define and implement that design becomes complete documentation.

The program will still be required to develop the same deliverables irrespective of how it the program is documented. For an aerospace or military program, for instance, this will include the Preliminary Design Review (PDR) documentation, Critical Design Review (CDR) documentation, the specifications from which the design is developed, the design documentation, the control and operations documents, and any such similar documents. When the documentation and the development process are merged in this way, all of the side conversations, the development notes, the white papers, and the individual efforts that occur naturally (and are naturally lost) in the course of completing the project are captured in the record.

Once this has all been captured, the information can be used judiciously to build an easily searchable repository. This is the sum purpose of a knowledge capture system. In particular, such a system will be composed of three parts, together sufficient to deliver information to the team member as efficiently as possible. These three components are:

1. Knowledge Capture

We have given considerable time to the discussion of knowledge capture as a component of the knowledge management system. In particular, we have discussed that the knowledge capture (documentation) component of a task must be done in tandem with the development process of the system. With that understanding, what processes can be used to implement such a process?

Instead of trying to merge two tasks by creating one big task, as was discussed earlier in an example, the simple solution emerges from the examination of the development task itself. The development task is composed of basic collaboration between developers and individual synthesis of that information into specific products that interface with one another to create a product. The collaboration takes place through phone, meetings, and email. The first two are limited in their effectiveness in directing a result unless the ideas are captured through meeting notes or similar records of the discussion and the third is itself a record. Typically, email is referenced over and over because it contains data that was developed during the course of the email and response process of working through an idea or contains the outcomes of meetings and phone conversations.

Rather than trying to convince team members to document the work, a better solution is transitioning the work into a system that *automatically* captures it. To this end, a system capable of doing so should support both the development of documents (basic report documents as well as presentations such as Powerpoint and spreadsheet-style documents with computational capability), designs (e.g., computer programs, mechanical drawings, simulations), and support team collaboration. To date, there is no integrated system that fully implements this requirement, but WIKI technology comes close for some of these needs.³

2. Data Search Capability

Users must be able to search for information stored within the system. While most solutions that are based on shared drives and data repositories (e.g., Microsoft™ Sharepoint, PIMS) provide some search capabilities, the best search engines available were developed for the internet. These provide keyword search on the title, stated keywords, and product content within the information stored in an intranet or the internet. When data is developed within a knowledge capture system, it should be stored in such a manner that it is available for indexing in much the same manner as an internet web site, including all of the content within any document, design, schematic, or similar source of knowledge. Unfortunately, this is often slow or impossible in repository or shared drive based data storage centers. With a WIKI or similar technology, it is almost always available as a part of the system. That is because these systems seamlessly integrate search engines – either those that they develop themselves or using popular solutions such as the Google or Atomz search engines.

³Automae is developing a system called Epicentre™ that provides basic WIKI capabilities as well as collaboration capabilities specifically in support of product development and sustainment processes. This technology includes features such as integrated collaboration tools, capture of data as it is developed and automatic cross-linking of that data by self-operating systems. Automae also provides consultation services to aid organizations and companies in implementing their own collaboration managed knowledge capture systems with Epicentre™ or their own technology platforms.

3. Plenteous Interactive References Between Data

A search engine is only part of the story in being able to quickly locate knowledge stored within the information system. The truth is that a search engine is limited in a certain sense. While a document may contain all of the search criteria (keywords or phrases), the actual focus of the document may still not be what is necessary for the person searching for information. If that is the case, proper hyperlinking within the document can quickly lead the reader to another document that may more effectively provide the desired information. Unfortunately, since a search engine does not automatically link the user to that data, the only recourse without this type of linking is to either back up and choose another returned document or conduct another search and go through the same procedure all over again.

The solution is to annotate the documents so that they point to related documents within the system that have been identified as resources for specific search terms. In typical systems, this is a hyperlinking process and the process itself takes some time. In fact, in conventional processes and systems, both the documentation of a project and the hyperlinking of information within the documentation are separate tasks. The result of these being separate tasks from the basic task of completing the development process effectively ensures they are almost never done. Fortunately, there are intelligent ways to do so without requiring extra effort on the part of the user beyond identifying keyword terms to the system and allowing it to annotate the information stored within the system based on keywords and embedded textual information.

Further Considerations

An important component of this process is its automation. *The user should not be required to do what the system can do for him.*⁴ Therefore, the process of data linking between documents using hyperlinks, the inclusion of links, the capture of knowledge, the integration of knowledge, and the handling of deliverable data sets (configuration management) should not increase the workload. Any effort to do so would impose increased requirements and responsibility on team members and represent a barrier to the adoption of the process. However, once the process is implemented and data captured, synthesized so it is searchable, and hyperlinked so that “close enough” is good enough when searching for data, the system will provide a substantial increase in capability to capture, manage, and find knowledge during the execution of a program.

All of this has only scratched the surface of such a knowledge capture and collaboration managed system- the “barebones” knowledge capture and collaboration management system, as it were. This basic design requires the components outlined here in order to provide our basic desired results. Without the basic design, though, the knowledge capture and collaboration management tool will be quite unable to enhance knowledge availability and capture. There is no reason to limit ones system to that set of capabilities, though. Indeed, it can be immediately enhanced with such features as configuration management tools and processes, coordination of data that appears in several documents and locations within the knowledge capture system, use of external development tools, annotation of referenced documents (Word™, Powerpoint™, Scilab, Matlab scripts, C++ files, pdf files) and so on. In fact, there are many logical advanced features that could be seamlessly included. We will discuss these in another upcoming report including detailed information regarding how these techniques were used in real systems and development programs.

Conclusion:

Knowledge management systems provide a good way to store limited amounts of data that are carefully configuration controlled and require careful maintenance. However, they lack the capability to provide a means for capturing information during the process of completing a development project. Data is often not easily found in such a system and is limited in accessibility due to non-robust search and data relation capabilities. A knowledge capture system based on the premise that *all development processes should be carried out within the knowledge capture system and automatically related and prepared for searchability* is far superior because it automatically traces the information as it is developed and provides it whenever it is needed. When it contains the three components of automatic knowledge capture with the capability to search through all this information and relate it with embedded hyperlinks, the data itself

⁴ This is a basic tenet to the systems made by Automae in order to increase the productivity of our customers.



becomes available to all team members immediately upon being captured and it is located easily when needed. This yields the most efficient knowledge capture and collaboration system available.