How to Build a World-Class Solution

ROW Information System

With the current climate of heavy infrastructure investment, the deregulation of utilities, and the explosion of the telecommunications industry, the importance of managing right-of-way asset information has never been greater. In order to manage these assets, location information, grantors, grantees, signers, mortgagees, mortgagors, recordation data (courthouse, deed book/page, recordation date, recording fees), legal description, tax identification numbers, and other attributes must be tracked. In addition, documents such as deeds, permits, and work sketches must usually be maintained and made easily accessible for an associated right-of-way asset.

Computer-based information systems provide an opportunity to effectively manage these assets. However, because the right of way problem domain is inherently complex and requires significant expertise in business processes as well as in the relevant technologies, software engineering best practices must be used to successfully guide the specification and development of right of way information systems.

Right of Way Problem Domain

The primary characteristic of the right of way problem domain is complexity of the business processes. Right of way assets often involve multiple geographical and political boundaries, and there are typically multiple staff roles involved in asset negotiation and management. In addition, effectively tracking right-of-way assets also involves multiple documents such as distribution permits, work sketches, master-supplemental agreements, tenant-at-will agreements, disclaimers, and quit claims. There may be multiple: parcels per document, signers per parcel, locations, recordation entries, crossings, and associated work order or project.

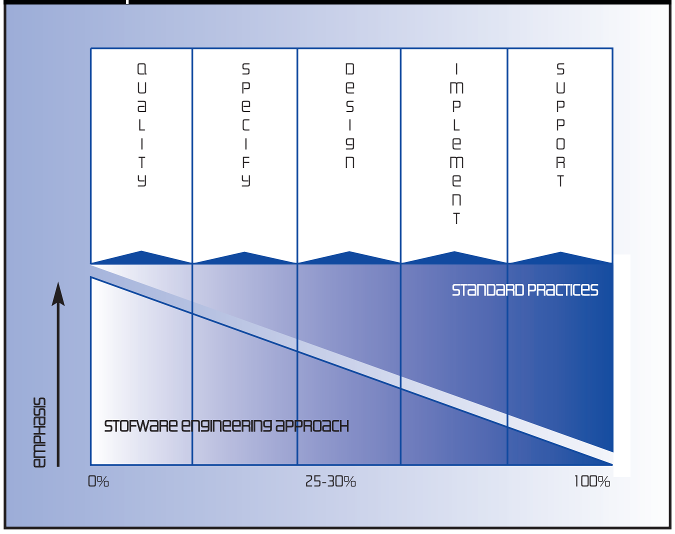
From a technical perspective, this complexity presents a significant challenge in data engineering as well as in facilitating the negotiation and tracking process through the use of imaging, web-based, and wireless technologies. Also, integration with a Geographic Information System (GIS) is essential in order to effectively present multiple views of the asset information.

Software Engineering Best Practices

The software engineering discipline applies engineering practices to the specification and development of software systems. The overall concept is to select a project lifecycle, team structure, and development platform based on requirements of the business problem. Software engineering is still relatively immature compared with other engineering disciplines. However, there are some efforts under way to certify software engineers with a professional engineer (PE) designation similar to PE certifications for other engineering specialties.[1]

The software engineering approach differs from pure programming by focusing on up-front activities, sometimes referred to as the fuzzy front end.[2] These activities include requirements engineering, architectural design, and software quality assurance. Research at IBM has shown that projects which focus early on these activities generally have optimal project schedules.[3] That is, there is no prolonged period of rework and bug fixes at the end of a project. Chart 1 illustrates this approach.

CHART 1



SOFTWARE ENGINEERING V. STANDARD APPROACH

Requirements Engineering

The most crucial phase of a software development project is requirements engineering. This phase involves meeting with clients to determine what the system is going to do as well as determining system attributes such as response time and usability requirements. Specifying requirements as precisely as possible is important since research shows that it costs anywhere from 10 to 100 times more to make a change later in the system development process.[4]

Many techniques are used to drive the requirements engineering process, including structured questionnaires, prototyping, and joint application design sessions. The essential problem is communication among humans, which is by its nature ambiguous. According to Fred Brooks, one of the early pioneers in software engineering:[5]

> "The most difficult work of software development is not in representing the concepts faithfully in a specific computer language (coding) or in checking the fidelity of the representation (testing). The essence of software development consists of working out the specification, design, and verification of a highly precise and richly detailed set of interlocking concepts."

This effort is made more complicated in that clients often do not know precisely what they want until they see something implemented. Therefore, it is important to plan for change and to develop generic structures that allow clients to adjust the technology to fit business practices as required.

Once requirements have been defined as precisely as possible and approved by the client representatives, the project moves into a formal change management process. Change requests are documented, examined for cost and schedule implications, and formally approved before they are implemented. Some research has shown that using a formal change process is one of the most crucial aspects for eventual project success.[6]

Architectural Design

The next project phase consists of designing an appropriate architecture. Building the right architecture is especially important in Internet and other distributed systems where many clients may be using the system at once and, in the case of Internet systems, where visibility is high and the consequences of system outages are public. Designing an appropriate architecture consists of identifying system software components and the relationships between these components. Vendors frequently provide generic architectural frameworks, which provide a foundation for developing a system-specific architecture. Examples include Microsoft Windows DNA or Sun Java J2EE.

Implementation and Integration

Implementation consists of programming and testing the individual software components. Once these components are working, they are integrated into a build that is then subjected to system testing. Testing at both the individual component and system level consists of white-box testing, or examining the behavior of the components, and black-box testing, or viewing the system as a black box with a set of expected outputs based on a given set of inputs.

Software Quality Assurance

Software quality assurance (SQA) is involved early in a software development project. SQA activities including assisting with requirements inspection, participating in design reviews, developing the system test plan, and testing the various software components. In addition, SQA is responsible for monitoring adherence to coding and development standards and compliance with the development methodology. Many commercial development firms try to test in quality at the end of a project rather than focus on quality from the start. As described earlier, this approach typically leads to a long test and fix cycle near the end of the project.

Benefits of the Software Engineering Approach

Using software engineering best practices results in systems that deliver correct business functionality and that are less expensive in the long run to enhance and maintain. This approach also encourages client feedback early and often. By encouraging this feedback, clients participate fully in the requirements engineering process and to take ownership of the applications.

Another benefit of this approach is that by engineering a solution before actually implementing it, any mistakes or omissions are caught early in the process and are much less expensive to correct.

Case Study: PRISM

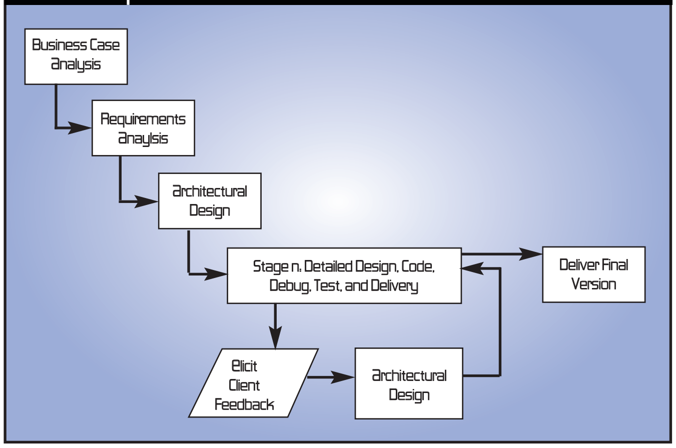
An example of applying software engineering best practices to the right of way domain is the Property Record Information System Manager (PRISM) developed by Computer Technology Solutions, Inc. (CTS) for Alabama Power Company (APCO).

APCO is a subsidiary of Southern Company with over 4 million property rights records, increasing at an average rate of 500 new documents per month. Over the years, multiple database and paper systems were developed to manage right of way assets.

"We tried off-the-cuff methods rather than a software engineering approach to build our right-of-way asset management system and wasted time and money in the process," says APCO project manager Wayne Boone. "In addition, field engineers were not getting the information they need in any kind of a timely manner, if at all."

In the fall of 1997, APCO decided to re-engineer and build a state-of-the-art system to manage property rights. What result-

CHART 2



ed was PRISM, a right-of-way asset management system that allows organizations to develop and maintain an accurate, centralized record of their property rights and holdings, and provides rapid retrieval and dissemination of associated permit and deed documents. With PRISM, a field request is answered with a document image in less than twenty seconds.

Prototypes were built with rapid development tools so that they could be easily changed based on new knowledge of business requirements.

Because of the complexity of the domain, the number of field engineers and other personnel involved, and the lack of success in past efforts to address all of the relevant business processes, CTS chose to use working prototypes for requirements engineering.

Working meetings were held with over 80 business subject matter experts to build prototypes that supported actual data entry of the various property attributes as well as scanning the associated documents. These prototypes were built with rapid development tools so that they could be easily changed based on new knowledge of business requirements. This iterative process continued until the business experts were satisfied that the prototypes addressed their business needs. "We first built the system in clay so that we could mold it before building it in steel," says Boone.

For development of the production version, CTS selected an evolutionary delivery lifecycle model. This approach involves delivering a build, soliciting client feedback, and then incorporating this feedback into successive deliveries whenever possible. Chart 2 illustrates this lifecycle.

Evolutionary Delivery Lifecycle Model

Data models from the various prototype applications were combined into a single data model and the various applications were partitioned into presentation, business, and data layers. This approach facilitates updates and enhancements as business requirements change. The primary data retrieval function was developed as a web-based query application, which could store both pre-defined and ad-hoc queries. This approach supports setting up the most frequently used queries before delivery.

Software quality assurance activities occurred during both the prototype and production development phases. The goal was to uncover at least 95 percent of any software discrepancies before delivering a build to client representatives. Load and stress testing were performed to verify response time and other system behavior.

"PRISM has been immediately accepted by end clients since they were instrumental in designing the applications and have ownership of them," says Boone. "The software engineering process has contributed greatly to the success of this system. Also, the field engineers like the fact that we have reduced turnaround time for inquiries from days to seconds."

Conclusion

The software engineering discipline, while still relatively immature, offers some best practices for using information technology to support complex problem domains such as right-of-way asset management. In particular, focusing on up-front activities such as requirements engineering, architectural design, and quality assurance helps to achieve optimal project schedules and to ensure that the system correctly addresses the business problems. ■

Steve Atkins is President and CEO of Computer Technology Solutions, Inc. (CTS), a software engineering company specializing in developing distributed applications. He is an adjunct professor for both the Department of Computer Science and the School of Engineering at the University of Alabama at Birmingham. He earned a BS and MS in Computer Science at the University of North Texas and an MBA at the University of Pittsburgh. Contact: (205) 943-6605 Fax: (205) 943-6606. satkins@askcts.com.

NOTES:

[1]For example, the state of Texas offers a PE for software engineers based on years of experience and recommendations from current Professional Engineers and is working with the ACM and IEEE Computer Society to develop a PE examination.
[2] Steve McConnell, *Rapid Development: Taming Wild Software Schedules* (Redmond, Washington: Microsoft Press, 1996).
[3] Capers Jones, *Programming Productivity* (New York: McGraw-Hill, 1986).
[4] Barry W. Boehm and Philip N. Papaccio, "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering*, vol. 14, no. 10 (October, 1988).
[5] Frederic P. Brooks, Jr., *The Mythical Man-Month, Anniversary Edition*, (Reading, MA: Addison-Wesley, 1995).
[6] Walker Royce, *Software Project Management: A Unified Approach* (Reading, MA: Addison-Wesley, 1998).

Searching for property records?
Managing your assets has never been easier.

PRISM, Property Record Information System Manager,
is the integrated solution for maintaining your
right-of-way assets and fee properties.

Including al
Ava
Let us show you h
right of way and
and increase asset
Er

Including all associated documents.

Available over the Internet.

Let us show you how PRISM can centralize
right-of-way and fee property information
and increase asset management efficiency.

Phone: 205.943.6605
Fax: 205.943.6606
Email: prism@askcts.com

www.askcts.com/prism