Visual SDLC: Improving Requirements Engineering for Object-Oriented Systems

Marc Raygoza
Visual SDLC
Aliso Viejo, CA 92656
marc@visualsdlc.com

ABSTRACT
In theory, requirements engineering solves many of software engineering's fundamental problems. The stakeholders know what the developers are building, why they are building it, when they are building it, and even to some degree, how they are building it. If requirements engineering resolves some of the basic communication issues between IT and the business, why aren't more companies actively practicing this discipline? In practice, requirements engineering is almost impractical without a commercial automation tool. The critics argue that the current automation tools do not convincingly demonstrate its value proposition, or fulfill the longstanding promises of the leading requirements engineering experts. This paper describes how the enterprise software development lifecycle management solution, Visual SDLC, addresses some of the outstanding issues of the present requirements engineering tools.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]:
Requirements/Specifications – *elicitation methods, languages, methodologies, tools*

General Terms
Design, Documentation, Management, Reliability, Standardization

Keywords
Enterprise Software Development Lifecycle Management, Requirements Engineering, Software Development Life Cycle, use cases, test cases

1. INTRODUCTION
Future requirements engineering tools need to provide more functionality than capturing textual specifications, and providing columnar-based tracability matrices. Since requirements interact with every facet of the software development lifecycle, requirements engineering tools should inherently provide the same round-trip functionality as enterprise lifecycle management software.

OOPSLA'03, October 26–30, 2003, Anaheim, California, USA.
ACM 1-58113-751-6/03/0010.

Visual SDLC's objective is to resolve requirements engineering issues, and in the process provide an intuitive enterprise lifecycle management solution that:

- Cultivates cross-functional team interaction
- Promotes software specification reuse
- Enhances integration between textual specification and modeling tools
- Promotes product lifecycle traceability
- Automates generation of object models from use cases

2. SOLUTIONS

Cross-functional team interaction
Requirements engineering is an area of software engineering that is more likely to involve more parts of the business than any other software engineering discipline. Business stakeholders provide the high-level ideas. The analysts capture these ideas, and transform them into meaningful requirements. The developers and testers verify these requirements and then develop and test, respectively. Visual SDLC enables software development teams to collaborate collectively using a requirements engineering workflow system that cognitively traces the ideas to the final product.

Software specification reuse
Why should the term reuse be an exclusive concept to programming languages? Visual SDLC facilitates perpetual reuse of specifications. Some of the leading commercial requirements engineering tools use general-purpose tools, such as Microsoft Word ™, to capture its software specifications. In Microsoft Word ™ to share information between two documents, one must perform a manual copy and paste between the source and target documents. What happens when the specification's content in the source document is updated? Obviously, the target document now has content that does not coincide with the source. Our agile document management solution manages the synchronicity of the source and target documents. This ensures information is

disseminated properly, and developers are not reading from an obsolete specification.

Integration between textual specification and modeling tools

Couplings between textual specification and modeling tools are immature and seldom used [1]. Engineers are required to develop specification using two or more tools: a tool for textual specifications and one or more tools for model-oriented analysis and design [1]. Visual SDLC is integrated lifecycle product that allows teams to visually model and textually define the software system in a single integrated environment. Hence, significantly minimizing the risk of redundant work and artifacts.

Product lifecycle traceability

Traceability is an undisputable characteristic of robust requirements engineering. There are two challenges with traceability: usability and maintainability. Rightfully so, most commercial requirements engineering products use a table-based traceability matrix that permits a many-to-many relationship. The graph data structure is appropriate since the notion is to map any traceable link to another traceable link. However, why does the underlying data structure need to be synonymous with the user interface? One of the basic problems experienced with a traditional table-based traceability matrix is that requirement specifiers almost blindly map high level ideas to functional and non-functional specifications thus defeating the purpose of traceability. Instead of assuring proper requirements coverage, and traceability to downstream specifications, it becomes an overlooked report since everything traces to everything. Visual SDLC provides an intuitive specification workflow that demonstrates the power of both requirement and document traceability. This presents a very powerful quality assurance feature for developing complex object-oriented systems.

Automatically generate object models from use cases

Migrating from use cases to object models still remains a challenge for some developers. The leading cause for this problem is poorly written use cases, and the level of granularity necessary to start generating an object model. This leaves a large margin of room of error for the developer. To resolve this issue, Visual SDLC instills proper use case development by providing a unique use case parsing algorithm to ensure use cases comply with simple active tense English sentences. Subsequently, Visual SDLC performs noun+verb identification extraction, and builds the object model automatically. The notion is not to generate a perfect object model from the use case specifcation; no lexical parsing program can knowingly achieve such greatness, yet to automate a meaningful percentage of the object model.

3. CONCLUSION

There is no "Silver Bullet" to improving requirements engineering for object-oriented systems. The closet thing could be bridging the ever prevalent communication gap between IT and business. As object-oriented systems become increasingly more complex and software development teams become more geographically dispersed, requirements engineering will only become more challenging. Visual SDLC offers a unique solution to address the outstanding and future requirements engineering issues.

REFERENCES

[1] M. Weber and J. Weisbrod, *Requirements Engineering in Automotive Development: Experiences and Challenges*, *IEEE Software, 20,* 1 (January-February 2003), pp. 16-24.

[2] D. Leffingwell and D. Widrig*, Managing Software Requirements: A Use Case Approach*, Addison Wesley, 2003

[3] V. Leino, *Documenting Requirements Traceability Information: A Case Study*, Helsinki University of Technology, December 2001