Quality Assurance for TDWG Standards and Software

Introduction

Software products, including software standards, can fail due to (1) complexity prevents completion, (2) the features change until resources are exhausted, (3) the product does not match user's needs, (4) user's cannot obtain help when they encounter problems, or (5) updates are not made to repair defects and maintain system compatibility. Causes 1 and 2 are examples of issues with managing the definition of the product, 3 is a problem in implementation and testing, while 4 and 5 are maintenance issues.

A software lifecycle defines the process of creation of the software and critical steps to minimize the problems mentioned above. It has been repeatedly shown that moving to a formal software process not only improves the quality of products but also reduces overall development costs.

Definitions

Users of standards and software products from the Taxonomic Database Working Groups (TDWG) include individuals who use a standard protocol, software toolkit, or the resulting system. This includes scientists, researchers, software developers, and IT professionals.

Stakeholders include funding organizations and TDWG executives.

The software product includes protocol standards, software toolkits, and associated documentation.

Software Lifecycle Flow

The software lifecycle begins by interviewing users to develop Use Cases that describe how they will use the product (Figure 1). Analysis of the Use Cases will then resolve some cases with existing products while other cases can be resolved with a combination of existing products and the product under consideration. The resulting features are added to the Requirements Specification along with other information such as system requirements, performance requirements, compatibility requirements, and the user's level of expertise.

The Design of the product is based on the Requirements Specification and guides implementation decisions to insure the product features meets the user's needs. The Test Plan is also based on the Requirements Specification and guides the Testing to insure the features, performance, compatibility, and robustness of the product meets the user's needs.

Defects from internal Testing and from Users are used to improve the product through fixing defects and by improving the Requirements for future releases.

Jim Graham, Ph.D.
Natural Resource and Ecology Laboratory
Colorado State University

Version: 1.1
11/5/2007

This circular flow provides a continuous product improvement cycle and keeps everyone involved focused on meeting the user's needs.

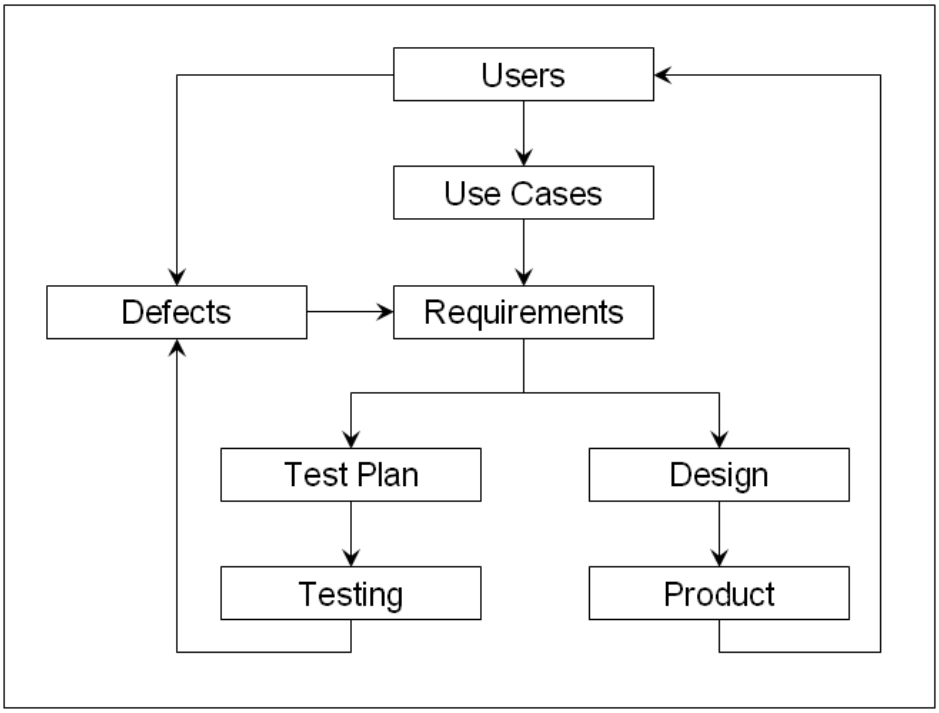

Figure 1. Simplified software lifecycle flow diagram.

References

Appendix A – Minimal Software Lifecycle

Below is a minimal software lifecycle. The most critical items to insure product quality are the Requirements Specification, Testing, and Customer Support that provides feedback into the design and implementation phases of future releases.

1. Investigation
- **Use cases** developed from user interviews
- **Requirements specification** reviewed by users and stakeholders
- **Feasibility studies** to assure robustness and performance can be met
- **Alternatives analysis** to select the best implementation technologies

2. Design
- **Product design** detailed enough to begin implementation
- **Test plan** describing test tools, test cases, metrics, and defect tracking
- **Maintenance plan** for customer support and upgrades

3. Implementation
- **Product**

- **Documentation** describing how to use the product
- **Test tools** for in-house testing of the product

4. Testing
- **Defects** at release for support to prepare responses to questions
- **Test metrics** tracking the number of defects over time

5. Delivery
- **Training** including existing defects

6. Maintenance
- **Defects** including defects from users
- **Test metrics**
- **Upgrades** to repair defects and resolve compatibility issues