

May 12, 2009

An Approach for Documenting Requirements in a Software Visualization

by **Chris Staufer**

Executive Summary

A software visualization captures an incredible amount of detail describing what the business user desires from the system end state. But how does a Business Analyst ensure that this detail is something that is functionally complete, i.e. that a developer can immediately take, and code to? This article recommends several approaches to audit and validate the requirements content present in a software visualization.

Process Flows

Viewed at a high level, the links that tie individual pages together in a software visualization actually represent a series of process flows mapping out the users path from some starting point to the end-state of the system. An important step in validating the final requirements deliverable package is capturing any process-level requirements that may exist.

One exercise that can be useful is to map out a visualization's process flows using a modeling tool such as Visio, or by leveraging existing navigation/modeling capabilities in the visualization platform used, such as the Scenarios feature in iRise. When modeling the process flow, decision points are a great area to focus on first, as a visualization doesn't always amply demonstrate at first glance the logic that is required to make a decision of when and how pages are displayed. It is critical for a Business Analyst to closely examine the various paths a user can follow, and to document any requirements that define the logic that the system will use to decide where to send the user next in the process flow.

If the visualization platform supports it, then the ideal place to document these process-level requirements is directly where the process flow is displayed in the visualization. Otherwise, the analyst should consider assigning a reference number to the modeled process flow, and capture the requirements in an external resource. Either way, the due diligence must be done to validate the process flows that are created as part of the visualization exercise, and to identify any gaps where they exist.

Business Rules and Non-Functional Requirements

Business rules and non-functional requirements are two types of requirements that are often difficult to "see" in a traditional software visualization. For this reason they can sometimes be passed over by the business due to a focus on what is happening on screen. It's important to keep this in mind when conducting modeling sessions with the business stakeholder, and to ask probing questions when potential Business Rules come up.

To get an adequate focus on non-functional requirements, it is recommended that a separate session be held to discuss and capture the non-functional requirements present for the application, outside of the fast-paced modeling session. If the visualization platform supports it, these two requirement types should be captured in a notes or other supplemental section that applies to the overall visualization.

Otherwise, it is strongly recommended that the Business Analyst leverage existing documentation templates, and consider creating a supplemental requirements package to include with the final product that captures these requirement types. This supplemental requirements package should document the items not readily apparent in the visualization, and be a focused, lightweight, deliverable. Often a visualization allows a user to create a page that links to external resources, in such cases, consider creating a global menu or navigation page that provides a link to this supplemental content.

Functional Requirements

Functional requirements explain to the development team the intended behavior of the software system to be developed, they focus on the "what" more than the "how." It is important when using a software visualization tool to not fall into the trap of assuming that a page in a visualization represents the complete picture of the page's functionality.

Some page-level functional requirements that can be overlooked in a software visualization include the following items:

- Alternate page states
- Validation rules

- Back-end system interaction
- Accessibility requirements

While alternate page states are often captured, in complex systems that have numerous user types, re-creating the same screen numerous times is sometimes avoided as a time-saving measure. In these cases authoring a set of functional requirements that explain in detail which fields are visible, editable, etc, can save modeling time and help the development team see what the end state system should display to the end user.

Validation rules are often difficult to appropriately model, due to the numerous error messages that must be displayed. Writing a requirement to supplement the displayed content can explain the gaps that may be raised by the development team. A series of “on submit requirements” can explain the order in which content on the page is considered, and what messages may be displayed depending on the data provided.

Which back-end systems a field pulls from is not something that is readily seen in a software visualization. Supporting requirements must be captured to explain where the information is going to be retrieved from.

Accessibility is an often-overlooked design element in a software visualization, but for a growing number of businesses seeking ADA compliance, it is imperative to represent what a user would see under a number of different circumstances. Alternate text, screen reader support, and color-blind behavior are all important requirements to document and associate for a given page’s elements. As previous sections recommend, it is strongly encouraged to capture these requirements at the page-level in the software visualization, so as the developer is reviewing it, they can immediately see these supplemental requirements and answer questions that might occur.

It is critical for the Business Analyst to closely scrutinize each element present in the final software visualization at the page level, and to seek requirements from the business stakeholder on the features that are not adequately represented. This sort of audit exercise fits very well into an iterative approach to requirements gathering, where the business is engaged multiple times over the course of several weeks, instead of in a one-shot setting.

Other Recommendations

Some other recommended practices to utilize include the following:

- Utilize a numbering system for each business rule, process-level requirement, functional, and non-functional requirement present in the software visualization. Doing so helps ensure a degree of traceability, and allows you to reference the requirement from external sources.
- Number the software visualization elements. Much as how we want to number requirements elements, it is imperative that the software visualization pages and flows be numbered as well to allow for external and internal referencing.
- Perform a traceability exercise frequently during the software visualization creation process. Often the only way we discover a missed requirement is by confirming that each business stakeholder requirement has a corresponding functional requirement. This practice helps the Development team in authoring technical requirements, to ensure traceability from the Business Stakeholder all the way through to the Test team.

Conclusion

In closing, a software visualization is a powerful tool that a Business Analyst can use when soliciting system requirements from key stakeholders. When using this technology, it’s important not to fall into the trap of assuming that the visualization represents a complete and total picture of the end system in its entirety. It is the job of the Business Analyst to audit the final visualization product, and to capture any supplemental requirements necessary to ensure a complete requirements package is created.