Visualization of Software Assurance Information

Martin S. Feather   Joel M. Wilf
Jet Propulsion Laboratory, California Institute of Technology
Martin.S.Feather@jpl.nasa.gov   Joel.M.Wilf@jpl.nasa.gov

Abstract

During the conduct of Software Assurance on a software development project, data is gathered on both the software being developed, and the development processes being followed. It is from this information that Software Assurance derives insights into the quality of the software itself and the efficacy of the development process.

For large software developments such data can be voluminous, making deriving and conveying insights challenging. This motivates our ongoing efforts to apply information visualization techniques to software assurance data. While visualization techniques have long been applied to software itself, the application to software development processes and the data they yield is relatively novel.

We report on several such applications and the insights they revealed. We offer some suggestions for the further investigation of information visualization techniques applied to assurance data.

1. Introduction

Information visualization is a dynamic field of study, with entire conferences devoted to its advancement. Information visualization has often been applied to software artifacts themselves (e.g., files and lines of code; software architecture; software change histories; software execution behaviors) as a means to support the understanding of software systems.

Our focus is on the use of information visualization of software assurance data, a relatively unexplored application. Our work takes place at NASA’s Jet Propulsion Laboratory, where software for many of NASA’s deep space missions is developed. These developments follow NASA’s Software Assurance Standard [1], which defines Software Assurance as “The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. [IEEE 610.12, IEEE Standard Glossary of Software Engineering Terminology] For NASA this includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, Software Quality Control), Software Safety, Software Reliability, Software Verification and Validation, and IV&V.”

Herein we show two of the kinds of visualizations that we have applied to such data, and recount viewers’ reactions to these visualizations. The first kind of visualization organizes the information with respect to a timeline, specifically the timeline over which software anomaly data is reported. The second kind organizes the information with respect to a hierarchical structure of software development processes. These approaches to organizing visualizations, referred to as “anchors” in [2], match the styles of data that we see occurring frequently in software assurance.

2. Trends over time in anomaly data

JPL’s spacecraft projects track software and hardware anomalies during the testing and operation of their spacecraft. Software Assurance is often involved in assessing the projects’ responses to these (e.g., to assure that the cause of the anomaly has been ascertained, and addressed as need be). Software Assurance also tracks the trend of anomalies over time (e.g., to see whether a spacecraft’s anomalies are being dealt with in a timely manner). Since Software Assurance is institutional, it can also compare anomaly trends between spacecraft, and over the long timeframe as software development practices evolve.

Our first foray into information visualization of software assurance information was in this context. Our colleagues Allen Nikora and Nelson Green obtained anomaly data from 11 of JPL’s planetary missions, spanning a decade [3]. Their data included the date at which each anomaly was reported, the spacecraft to which it applied, and an indication of whether the anomaly was in the flight software (executing on the spacecraft) or the ground software (executing on computers back on Earth). The temporal nature of this information led us to devise a visualization anchored on a timeline, described next.

2.1. A timeline-anchored visualization
The features of the timeline-anchored visualization we devised are shown in figure 1. Each anomaly is displayed as a colored circle, located with respect to a timeline running left (earlier in time) to right (later in time). When multiple anomalies are reported on the same date, they are displaced vertically to avoid overlap. Vertical displacement is also done for anomalies on different but nearby dates to avoid overlaps. The net result is the ability to concisely display points representing a large number of anomalies.

Figure 1 - Timeline-anchored visualization

2.2. Timeline visualization of anomaly data

Our visualization of Nikora and Green’s complete dataset of 11 missions’ software anomalies, approximately 2,700 in all, is seen in figure 2 (filling an entire page at the end of the paper).

The timeline, spanning 1996 until early 2007, is indicated along the top. The 11 missions are portrayed as faint rectangles, the left edge of which represents the launch data, the right edge the end of mission, or anomaly collection date for missions that were still ongoing at that time. The tiny colored dots denote anomalies – blue dots for anomalies in the ground software, red dots for anomalies in the flight software. For some missions, anomaly reporting began during testing prior to launch – the dots seen to the left of the mission’s rectangle in those cases.

2.3. Reactions and Observations

When this chart was included in presentations of the work, many of the audiences’ comments were expressed with reference to this presentation. They were not commenting on the pros and cons of the visualization itself, but rather on the insights they were able to draw from the information it conveyed. Most notably, members of the audience who were knowledgeable of the context behind some of the missions’ data offered numerous key observations that helped explain the data. Examples of these are as follows:

The Deep Impact (next to the bottom on the chart) mission’s flight software personnel recognized that its software anomaly data was vastly over reported – it actually included numerous hardware anomalies (the database field in which anomalies’ types were recorded defaulted to “software”, and during data entry people rarely bothered to change it to “hardware”). This motivated Nikora to propose and conduct a subsequent research effort to automate the classification of anomaly data.

The Mars Pathfinder (2nd from the top on the chart) mission’s flight software personnel explained its relative paucity of reported anomalies as due to the fact that the mission benefited from use of previously developed and tested ground-to-spacecraft software. This reveals that certain kinds of software reuse can be particularly effective, contrary to widespread mixed feelings about software reuse in general.

The 4th and 5th missions on the chart were actually lost because of software problems. During a presentation to the NASA Software Working Group, John Kelly observed that the data showed very few software anomalies leading up to their loss. Was this because during that “Faster Better Cheaper” era there was less recognition or reporting of anomalies? He also observed that the visualization made obviously made no distinction between the “severity” of the anomalies on this or other missions (which was because the dataset from which the visualization was crafted didn’t include this information – if it is available it’s easy to incorporate such information in the visualization, as will be seen in the next section of this paper). This raised some key follow-on questions, such as:

If different severity levels were portrayed, would it support the notion that a flurry of less critical anomalies can serve as a precursor (e.g., [4], [5]) of dangers ahead?

What consistency is there between different missions’ reporting of anomalies?

Over this decade JPL had progressed through CMMI level 2 to level 3; has this resulted in more diligent and consistent reporting?

In general, what can be concluded from the statistical analysis that comingles data from such uneven reporting and disparate severity of consequences?

Since information of this nature is used to establish a software quality baseline against which to measure future progress, guide investment decisions, and focus software research & development, correct interpretation of such data is particularly important.

The reactions recounted above indicate that our audiences had no apparent difficulty comprehending the visualization. Those who were knowledgeable of the individual sources of the data were readily able to convey their additional understanding of the situations from which the data was derived. Those who were interested in interpreting the data set as a whole were
readily able to offer insights, questions and suggestions of further work to perform.

3. Another timeline-anchored visualization

We have recently developed a variant of our timeline visualization for inclusion in a “dashboard” like display of project anomaly information.

In this case the anomaly information includes the “severity” level of each anomaly (on a four-valued scale), so our visualization uses color to indicate this:

- Black = highest severity
- Red = next highest
- Orange = medium severity
- Green = low severity
- Gray = severity not assessed

This visualization also indicates both the dates at which anomalies are “opened” (i.e., reported), and the dates at which anomalies are “closed” (i.e., the anomaly has been investigated and, if corrective action has been deemed necessary, such action has been completed and confirmed).

Figure 3 shows this for one of JPL’s ongoing missions. As before, the timeline runs from left to right. The opening of an anomaly is portrayed as a circle in the upper white area, located with respect to the horizontal timeline according to the opening date. If it has since been closed, the circle is drawn filled in; if it is still open (as of the time of the data extraction), it is drawn as an open circle. The closing of an anomaly is portrayed as a circle in the lower left white area. Anomalies can be assigned dates by which they are due to be closed, so currently open anomalies due to be closed in the upcoming month are portrayed as circles in the lower right white area, and anomalies that are overdue to be closed are portrayed as open squares in the upper white area.

The jagged blue and yellow trend lines visible in the upper white area indicate running totals of anomalies opened but not yet closed. The yellow trend line is the running total of yellow-severity anomalies, the blue trend line the running total of all anomalies. On this figure the green, red, black and gray trend lines are not readily discerned, as they lie close to the timeline where the circles swamp the view.

3.1. Interactivity

As mentioned above, this visualization is part of a larger dashboard. The dashboard is interactive, offering the user the ability to change the timespan over which anomaly information is portrayed. For example, the user can focus on just the last month and upcoming month. The user can also filter the anomalies portrayed based on their severities. Figure 4 shows the same timespan and dataset, but where filtering has removed the numerous yellow and green (lesser severity) anomalies, to leave visible only the much less numerous black and red (higher severity) and grey (severity not rated) anomalies. (We have since modified the visualization software to automatically plot the trend lines for the infrequent high severity anomalies on the same chart but with respect to a different vertical scale as an alternate to such filtering.)

Also on the dashboard are lists and tables of additional information (e.g., summary statistics) using standard lists and text boxes. Useful as these are, since these are not particularly interesting from an information perspective, they are not presented in this paper. We added two-way interactivity to this – the user can click on an icon in the visualization and see the corresponding item in the lists of such, and vice-versa.

3.2. Reactions and Observations

Our visualization was crafted with help from the software assurance personnel whose responsibilities include regularly reporting on the status of ongoing software developments. Their interests include understanding both the current status of a development (e.g., how many anomalies are currently open), and the trends in that development (e.g., are anomalies piling up faster than they are being closed?). It was this latter need that drove us to including the trend lines to this visualization.

We found that the audience of this visualization could readily comprehend it following a brief verbal description of the visualizations’ elements and their meaning. For example, a mission assurance manager commented that the dominance of the yellow colored circles (indicating medium severity) suggested the need for a subdivision of that severity category. As
before we found that they were readily able to offer insights into key events in the development schedule as explanations of, for example, the notable “spikes” in the anomaly opening and closing activities. This led us to recently add the capability for users to provide their own list of key events and their dates, which are then portrayed as annotations on the timeline visualization.

The positive reactions to the trend lines on this chart prompted us to add a similar feature to our visualization of results of software failure data analysis recently performed by Nikora and colleagues [6]. Figure 5 shows two extracts from the entire visualization we generated from them. Here, the blue and red circles denote two kinds of faults, as before, placed horizontally with respect to a timeline. The red and blue lines indicate scaled running averages of Times To Failure (TTF) (time intervals between successive failures). The diverging red and blue trend lines on the upper portion indicate reliability growth (time between failures is increasing), whereas the red and blue trend lines on the lower portion clearly don’t have such a simple interpretation.

4. Adherence to institutional standard software development processes

JPL, like many organizations that develop significant amounts of software, has codified the software development process that its software development projects must follow as they progress through the software lifecycle. One of the roles of Software Assurance at JPL is to assess adherence to these standard processes. For a given software project, Software Assurance’s confirmation of the project’s adherence to these standards gives confidence that the software development is on track. Conversely, Software Assurance’s identification of discrepancies against the standards points to areas of concern. For example, omitting a review activity could potentially lead to a software artifact advancing to the next phase of development before it has achieved sufficient quality.

JPL’s software development process standards are organized hierarchically, e.g., “Software Product Delivery” decomposes into “Prepare for software product delivery”, “Review and approve delivery” and “Deliver product to customer, install, and verify”. JPL’s Software Assurance personnel record their assurance activities in a common database. When their activities are assessments of projects against the process standards, the activities themselves are recorded, and any discrepancies against the standards identified during those assessments are also recorded.

This hierarchical structure prompted our selection of the Treemap visualization, described next

4.1. A hierarchy-anchored visualization – Treemap

“Treemap” is a well-known information visualization technique invented by Ben Shneiderman [7]. It uses rectangles to subdivide into smaller rectangles to represent the data’s hierarchy, setting the leaf node rectangles’ areas proportional to an attribute of the data, and (typically) setting their color to indicate another data attribute.

Different variants of Treemap take different
approaches to control the placement and sizing (height to width ratio) of rectangles, achieving different balances between the aesthetics of avoiding very narrow rectangles, and the closeness by which layouts that mirror the order of elements in levels within the hierarchical data.

4.2. Treemap visualization of process adherence findings

An example of our application of the Treemap visualization to software assurance data on projects’ adherence to JPL’s software development process standards is seen in figure 6 (at end of paper).

The Treemap features of layout, size (area), and color are used to convey the following:

- **Layout** – the topmost level of subdivision of the rectangular display area, indicated by the black-bordered interior rectangles, corresponds to the major sub-divisions of JPL’s institutional standard software development processes. For example, the black-bordered rectangle in the upper left corresponds to the “Software Product Delivery” process area. Decomposition continues into: process sub-areas (grey-bordered rectangles), software development projects, and, at the lowest level, individual findings (of discrepancies against the standards). The version of Treemap used here is “Squarified Treemap” [8], which favors keeping the areas generally square-like (i.e., avoiding where possible skinny rectangles).

- **Size** (area) – the priority assigned the finding by Software Assurance (“critical”, “major”, “neutral” or “minor”) is used to determine the area of the innermost rectangles (individual findings). Here we use areas in the ratio of 8:4:2:1 to represent those four priority values.

- **Color** – the risk rating assigned the finding by Software Assurance (“high”, “moderate”, “not rated” or “low”) is used to determine the color of the innermost rectangles. Here we use colors red, pink, white and green to represent those four risk rating levels.

- **In figure 7 (at end of paper)** the Treemap display has been hand-annotated to point out several of the observations offered by this visualization.

There are noticeably plentiful large and/or red findings’ rectangles present in the highlighted top-left and top-middle process areas; (the top-right process area is a close third).

Conversely, the highlighted area in the bottom-right of the display contains ten process areas whose total findings occupy about the same area as the single process area in the top-right.

The highlighted bottom-left area contains a project with numerous (close to 40) unrated neutral priority findings – the white rectangles obscured by the legend. In that same process area some other projects (to the right of the legend) also have several unrated neutral priority findings, plus some green rectangles indicating low risk rating findings.

As with the timeline visualization, interpreting these observations requires care. For example, numerous discrepancies within a process area could have many explanations: possibly the projects are performing poorly at adhering to the standard (which might be due to it covering a particularly challenging area of software development, due to poor project practices, or due to poorly crafted standards); possibly the scrutiny of adherence to that standard is particularly thorough as compared to the scrutiny given to other standards.

4.3. Reactions and Observations

We have used this visualization to present data from the last year’s activities of JPL’s Software Assurance Group, both to the Assurance Group itself, and to the Software Quality Improvement organization within JPL (distinct from Software Assurance within JPL). It was necessary to provide a brief explanation of the Treemap visualization, and how it was being used on the assurance processes and data. After this, each audience seemed readily able to grasp its contents, as exemplified by the following reactions:

- **Within the Software Assurance group**, there was interest in distinguishing between the projects contributing to the overall pool of data. In particular, there was the suggestion to divide the data into that pertaining to “flight software” (software that operates on board spacecraft), and “ground software” (for example, software used by the Deep Space Network), and present a separate Treemap visualization for each. Doing so revealed the differing emphasis on the process areas between this split of activities. Dividing the data further, to look at individual missions (and thus see how a mission compares to others) was also suggested. Overall, the visualization added credence to the long-standing impression within the group that there was a need to ensure better consistency among the details of assessments (risk ratings, and priority). There was also discussion of the way the visualized information reflects the different ways that JPL’s software assurance personnel interact with software development efforts and the value that assurance provides [9]. Those called upon to perform “intervention” assurance (proactive involvement while artifacts are being developed) yield a different pattern of findings than those called upon to perform “investigative” assurance (e.g., assessments of artifacts at key decision points in the development lifecycle).
future work we would like to incorporate information on the effort expended on activities within the visualizations.

When presented to the Software Quality Improvement Organization, they requested some modifications, notably to forgo (for the time being) the rendering of risk ratings and priority so as to focus primarily on the frequencies of activities within the areas of the JPL development processes and the distribution of findings they yield. We modified the visualization accordingly, and they then used that in their regularly scheduled reporting to JPL’s mission assurance management. Their focus on the development processes (as compared to the software assurance personnel who each have an interest in specifics of the development effort(s) they are involved with) prompted us to switch from the “Squarified Treemap” to the original Treemap layout algorithm, because the latter holds constant the ordering the hierarchy’s elements on the page (e.g., the same area always appears in the top left corner), even though this has the drawback of typically resulting in more overly skinny rectangles.

The interactions with the Software Quality Improvement Organization also triggered an interesting discussion on activities that yielded zero findings – an overly narrow interpretation of the value of software assurance would dismiss these activities as wasted time and effort, but in fact they have yielded value: the information of an activity having been performed and yielding zero findings increases confidence, which may be of great value to a decision maker faced with a key decision.

5. Conclusions

We have shown several examples of our application of relatively straightforward information visualization techniques to cogently present software assurance data.

In the software realm, it is commonplace to apply information visualization to the artifacts of software development. For example, [10] adapted Treemap and Sunburst (another approach to organizing visualization with respect to hierarchy) to visualizing several quality attributes of hierarchically structured object-oriented software. The visualization techniques we apply are in widespread use for many other purposes. However the application of information visualization to software assurance appears novel and, we think, underutilized.

We have not performed experiments to assess the effectiveness of these visualizations, or compare with alternate presentations (e.g., tables). It would clearly be desirable to do so, especially should we move to use of more sophisticated visualizations. Instead, we rely on the reactions of our audiences. We are able to infer from their reactions whether they correctly understanding the visualized information. When they offer suggestions of additional ways to use the visualizations, we are able to infer that they see value to continuing in the pursuit of information visualization for those purposes. When they choose to use the visualization to in turn present information to others, we infer that they perceive the visualization as improving their communications.

Some of the specific insights we have found to date in pursuing this approach are:

It takes a little bit of coaching to understand these visualizations – as in this paper, they have to be explained to first-time viewers, but such explanations can be provided quickly.

Once understood, our repeated experiences with showing them to various audiences has repeatedly shown that they effectively convey “the big picture”, more so than (say) scrutinizing the same data presented in lists or tables (although we have not performed any usability studies to confirm this intuition).

Useful insights from the visualizations come from the people who understand the details behind the data. For example, in the timeline visualization of the 11 missions’ software anomaly data, it was the people familiar with the individual missions who could offer explanations of the evident patterns.

We find it useful to prototype our visualizations, trying them out on real software assurance datasets to explore how effectively they perform. The construction of the visualizations themselves is more time consuming than we would like, in part because the data is not always conveniently assembled in one place, and in part because have not yet populated a “toolkit” of visualization widgets to draw from. Almost inevitably when we present one of our visualizations to others, they offer suggestions of alterations and/or features to add. Ideally we would like to produce a toolkit that others could adeptly use to craft visualizations to their liking.

Overall we are encouraged by our experiences to date. We think that software assurance can derive benefit from the application of information visualization techniques. The visualizations that result can potentially serve several different information functions: to discern the status of individual software development efforts; to discern status with respect to institutional concerns (e.g., development standards); to discern long-term trends across multiple projects. We also believe that the utility of this will continue to grow as the scale of software development efforts at JPL increase, and as the collection of assurance data...
improves, both trends leading to the collection of more data.

6. Acknowledgements

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration, and at NASA Ames Research Center. We thank Allen Nikora and Nelson Green for the opportunity to work with anomaly datasets they assembled, and to JPL’s Software Assurance group for the opportunity to work with data they too have been amassing.

The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance and Technology Program Office.

7. References


Figure 2 - 11 missions' flight and ground software anomalies
Figure 6 - Treemap visualization of findings against software development standards

Figure 7 - Treemap visualization, annotated

These two process areas have numerous critical/major priority, high/moderate risk findings

A project with numerous unrated neutral priority findings

Below: 10 areas with very few findings