Strengthening Development Workflows by Graphically Communicating Elements of Software Design

Date: 12 June 2024

Presented by: Rafael Mudafort (National Renewable Energy Laboratory)

(The slides are available via the link in the sidebar of the page linked above.)

A related book recommendation from the audience (Juan Espinoza):

> The best reference for model-based software engineering is Steve Tockey's textbook How to Engineer Software: A Model-Based Approach, in Amazon: https://www.amazon.com/How-Engineer-Software-Model-Based-Approach/dp/1119546621

Q: Can UML be used to model new features of C++ like concepts or modules?

A: Possibly, but I'm not aware of any specific UML features for language features specific to C++. I say "possibly" because there is a mechanism for extending the UML, and this is where SysML that I mentioned fits it. It's an extension to the standard UML models to support models specific to systems engineering. I also heard of another UML extension for cyber security. However, I couldn't find anything for C++ language features in a quick web search.

Q: Are there any code metrics that can be computed from UML?

A: UML itself is a set of language constructs, so UML can't compute code metrics directly. However, there are a number of tools available to create UML diagrams through a static code analysis, and this process could provide some code metrics. Something like the depth of a particular object hierarchy (to go back to an example in the talk) could be determined by a static code analysis. My guess is most of the tools that produce UML diagrams from static code analysis do so with an abstract syntax tree. Of course, any kind of run-time metrics will require profiling tools.

Q: How does documentation of requirements fit into the document-driven process? The emphasis is on design, but at least rough requirements should proceed this. How could traceability between the requirements and design be handled? Likely changes should be documented before starting the design, since they govern the design.

A: In my opinion, the identification and *communication* of requirements is intrinsic to documentation driven design or documentation driven development. It's usually important to know what you need to do before you start doing something. The type of diagramming and documentation that I mentioned here is really directed at software architecture, so defining

requirements should probably be done prior to thinking about the architecture. In fact, I shared an anecdote where my lack of defining the requirements led to an architecture that was exactly wrong for the way the software was ultimately used. And I completely agree that changes should be documented before starting the design especially because they'll nearly always influence the design.
The question on traceability between requirements and design is a good one. I wonder if somehow this could be captured through the sort of diagrams I mentioned today. As in, maybe careful selection of perspective in diagrams would allow for visually communicating requirements.

Q: What advice does Rafael have on design principles to use? Composition over inheritance? Recommended design patterns?

A: I'm hesitant to recommend any specific design principles to follow because I think these tend to go in and out of style. Even in my own mind I tend to vacillate on these types of questions like composition vs inheritance (though I've been on the side of composition for a few years now). Instead, I recommend to create a design *process*. Think about how you're moving through the design of your system: How are you incorporating requirements? Do you have a *parti*? How will you verify that your design has done what you intended? In my experience, building a strong design process enables building a strong design, and then questions of one pattern over another are less relevant than questions of achieving the objectives of your specific project.

Q. What tool did you use to draw the diagrams (especially the sequence diagrams)?

A. I used Mermaid in combination with pyreverse for nearly all of the class diagrams in this talk. I created the sequence diagrams manually using Mermaid.

Q. Did you have to configure/tell your GitHub environment to use Mermaid to render the diagrams (in screen space)?

A. No, any GitHub product that uses GitHub flavored Markdown supports rendering Mermaid diagrams. See this GitHub blog post.

Q. Will these tools work with code that uses mixed language programming - so a mix of C++/C/Fortran with calls to external libraries?

A. Doxygen does work with a combination of C / C++ / Fortran. Pyreverse is Python-specific, and I'm not sure how it would respond if you somehow integrate another language such as through an extension. That being said, most language ecosystems seem to have their analogous tools for this.

Q. What is the incentive for researchers to communicate their research well to non-experts?

A. This is a big question, and there's probably a lot to be said here. Instead of answering as a researcher, in general, I'll take it specifically as a research software engineer working at a National Lab with many colleagues who are closer to the domain in which our software is used. From that perspective, the viability of my career path is dependent on my communication about my work. Since most people in science and engineering are at least somewhat familiar with programming, I think it's easy to trivialize software design. However, writing a simple script to do some algebra and make some plots is an entirely different thing than creating an elegant software system that meets a set of requirements. It's important for us to communicate the things we've done that add value to the research environment so that our colleagues and managers understand it and can incentivize it. Without this, the RSE career path has limited growth potential and talented, experienced software engineers move on to other industries. I see this as a fundamental issue in research, at the moment, and one that can be solved by simply communicating more. I can't expect my colleagues to understand the design of a system that I created unless I tell them. Similarly, I can't expect my colleagues to understand the value of my work unless I tell them. This is the same thing we do with other products of research such as citation metrics for academic publications.
   This doesn't address the incentive to get your work to people who can make use of it. For me, that's another very strong incentive. Personally, I like to produce software as a means to an end - the end is wind energy production. I'm motivated to make my work more accessible so that companies creating wind turbines and developing wind energy projects will use my work to improve their products and ultimately help in the transition to renewable energy.

Q. Can you elaborate on how receptive your management and your "customers"/colleagues have been to your use of these tools?

A. My manager also has a strong computational background and has done his fair share of software development, so he sees the value in whatever method of communication supports our software development. My manager's manager probably has not seen my use of diagrams :) I think most of my colleagues, though, are generally appreciative of visual communication about software design since it really eases the burden on them when they need to understand my work.

Q. To what extent do you use these tools when writing up your research? Do you see them as only useful for "software" papers or are they also useful for domain science publications?

A. I primarily use these tools in software related products such as documentation site, GitHub, even within the source code. For scientific publications, I've historically not used these tools because I don't find their outputs very pretty. However, I've been making an effort to understand the styling and configurations available in Mermaid, and I coincidentally did include Mermaid diagrams in a recent paper and conference poster.

A suggestion from the audience (Jason Gates):

For generating these types of diagrams for the sake of publication, use LaTeX + tikZ.