CS-425 Software Engineering

Individual Assignment | #1

A possible project idea, CSE related job targets,
and analysis of software quality and use.

Contact:
Stephen Foster
stephenfoster@nevada.unr.edu

1. Possible Project Idea:

An idea for a software intensive product that as an entrepreneur or manager would want to be in development is a tool that manages and processes non-trivial, domain-specific requests from potential business clients for an arbitrary type of business. To gain an understanding of the tool, an anecdote can be used to demonstrate the gap in which such a tool could aid in filling. For example, if as a potential client I need to obtain a quote for custom commissioned work, for something like printing, or product acquisition, or any number of other domains. Current workflows between many current businesses and clients are inherently bottlenecked as far as time, as certain business initiations or events are inherently locked behind the necessity of human-human interfacing. If for example, it was possible to submit explicit sets of input, submit asynchronous communication, and authorize transactions, these bottlenecks would be reduced to their theoretical minimum between both clients and businesses. The software to be built could provide templates for certain kinds of product work or commissions, that all derive from our product base that serves as a layer of security and management. On the user end, it's a web application that abstracts and simplifies workflows for both client and business, and on the service end a business may engineer their own connections to the software's API endpoints. If certain aspects of some kind of work rely on current manufacturing division metrics, then workflow conditionals in the software would have the ability to access the derived result to inform both users and businesses, without the requirement for explicit inter-division or inter-organization communication and results in an optimized workflow, while maintaining the ignorance of our software's knowledge of proprietary information.

A group of 3 or 4 software engineers, along with security specialists and marketing specialists would work on the web application that encapsulates the total service. Two engineers would focus on frontend work, and consult with marketing and user specialists that could obtain domain research that would influence interface design. The other two engineers could work on the backend service that encapsulates the management of arbitrary business information that triggers conditionals in the workflow of commissioning. They would work with security specialists to ensure that the backend remains ignorant of a business' proprietary data, and implement a DMZ that provides a business' engineering team the ability to simply extend and fill out small bits of logic functionality.

The idea had its genesis in the numerous occasions that I have required custom artisanal or manufacturing work, and the answers to my questions remained behind numerous time-consuming phone interactions and requests for information. I so wished for a way to just put in work upfront, and then immediately understand the intrinsics of my commission in relation to the specific business in its current state.

In relation to current solutions, this proposed solution actually remains quite small and domain specific. The solution can actually be thought of as a typified subset of generalized commissioning. Current macro solution softwares in existence are CaptivateIQ and Everstage to name a few. Their core tenets are automation of commissioning processes relating to calculations, payouts, tracking, and quota. In essence, they exist as a partial solution to the fiscal side of vendor management. The proposed solution would exist in the gap these softwares leave open, and that's the optimization of the required human-human interfacing in the initiation of commissioning cycles between businesses and vendors, and satisfies a direct to consumer element as well.

2. CSE Job Target:

My dream job upon graduation would be the title of Software Systems and Applications Engineer at the National Aeronautics and Space Administration. I'm in love with the thought of being a critical component in the advancement of human knowledge and fulfillment of the wonder intrinsic to the human experience. Since I was a child, my mind has remained in the sky both figuratively and consciously. Some of my fondest memories include the times I spent late summer evenings with my primary school teacher out on our school's blacktop, along with some peers and likewise-interested members of the community, all oscillating between our faces buried in telescopes and our chins pointed to the sky. The opportunity to partake in such-related endeavors through the direct application of a skill-set I've been in the process of developing my entire life would be of the utmost meaning.

There's quite a bit to expect with such a role, as the systems I'd participate in designing and implementing would be incredibly lateral across a relatively non-discrete number of domains, while also simultaneously requiring longitudinal knowledge of varying depths in each. As a Systems and Applications Engineer, I'd be part of designing and implementing the cyclical systems required to both maintain life and conduct research. I imagine applications could range between the development of embedded systems that manage anything from life support systems, space suit functionality, or terrestrial rovers. Potentially even auxiliary systems for space shuttles or space stations that wholly incorporate both the highest of efficiency design theories alongside the highest of safety and reliability standards. The concept of mission-critical stems from the groundbreaking to the mundane, and as such the job is essentially the role of hyper-focusing on distinct domains to find their discrete set of relavancies, then prioritize order of system responsibility based on a repeatedly iterated set of findings.

Although a comprehensive plan of skill-set preparation is most likely only possible with direct experience, I am sufficiently familiar with at least some relatively main computer science-related areas of study. Due to mission criticality, concepts such as Discrete Mathematics, Algorithm Analysis, and Embedded Systems Design. The ability to mathematically isolate sets of code paths–and their most fundamental components–in order to prove program safety and security is the bare minimum for the job-scope. When tens of billions of dollars' worth of equipment and man hours rely on relatively hot and small code paths, perfection here is a must, and is exponentially more so when considering that lives may rely on them too. Additionally, knowledge of Materials Science, Mechanical Engineering, and Electrical Engineering would be valuable co-requisites, as understanding the basis of these domains would allow me to design around the constraints any System or Application is inherently bounded by.

3. Software Quality:

A software ecosystem I'm familiar with is the IoT ecosystem belonging to *Cync*, an outwardly seamless application with a client entry-point into the ecosystem of both C by GE and Cync branded IoT smart devices. The list of devices range from digital light bulbs, security cameras, digital circuit switches, motion sensors, and wireless arbitrarily targeted dimmers and fan controllers, as well as both thermostats and their mesh sensors. Its core purpose is to service the authorization and authentication of the API communications that elicit functionality. This is achieved through wrapping access through a proprietary server, and associating such access to user accounts that act as collections that manage the settings of a client's local mesh network of devices.

The first example of a feature that demonstrates good quality is the mobile application's user interface that wraps the whole update, authorization, and authentication loop. GUI elements and workflow further abstract implementation knowledge from the user, while only exposing endpoints that a user would be both familiar with and would want to interact with such as color, timing, and device relationships. This results in easy-to-use software for both technical and non-technical users. The second example of a feature that demonstrates good quality is the seamless and wide range of interaction devices within the ecosystem share with one another. Devices doubly serve as mesh network nodes that perform IoT responsibilities such as message sharing. Disregarding the requirement of wireless internet connectivity for submitting behavior updates to the mesh network, the mesh network can perform its responsibilities with just access to electricity, as each device maintains bluetooth transceivers that synchronize timing and behavior in a standalone fashion. The third example of a feature that demonstrates good quality is the authorization, authentication servicing of device API requests. Along with its implementing two-factor authentication, this software process dually insulates mesh network interaction from security threats and client-sensitive information in the cloud. This effectively manages user responsibility in regards to data management and workflow.

The first example of a feature that could be improved upon is the interoperability of the ecosystem with down-the-line softwares and platforms such as voice-assistant integration and home automation servicing solutions. Current interoperability remains flawed, where many likely user flows result in invalidating the invariants used by software up-the-line, and results in the necessity in manually resetting states that are only exposed indirectly to a user through the collection of their installed interop applications. It's self-evident the disruption, however native interfacing is the proper solution to this interoperability problem, as the ability to invalidate invariants down-the-line indicates non-native interoperability solutions being used. The second example of a feature that could be improved is the feature-set and platform availability of the GUI client. As of writing, there is no first-party desktop application. Third-party solutions rely on vulnerable invariants, as well as the fact that the API interactions are either round-a-about or reverse-engineered with limited ability. This leads into the third feature that could be improved, and that's first-party documentation of ecosystem and outward facing structures and endpoints. For example, when two-factor authentication first went live for the ecosystem, third-party solutions had to reverse-engineer communications to find the new server address to send authentication requests to, as well as deduce the intrinsics of a two-factor authentication solution. As a consumer, this kind of scenario is unacceptable.