The Evolution of CTTS Development and Data Structure in Three User Cases

By Ruben Safir
Justin Lau
Trevor Ashley

Structured System Analysis & Design CS633

The goal of project management in the enterprise, when handled properly, is the efficient reduction of organizational needs, as perceived by organizational end users into working digital systems to promote business goals. Along those lines, analysis of how data is construed within an organization, is a complex task, that as until recently, minimal thought has gone into the data flow and analysis of its usage, except as byproducts of other business goals, like the production of physical products which are brought to market. It has not been until recently that we have come to understand that while the traditional widget based view of business is valuable, as we have come to learn from traditional economics dating to the seminal works of Adam Smith, that it is often helpful to turn this view upside down, as see that marketable widgets are actually a byproduct of effective data usage and analysis, driven by the data, and that the widgets that we stick in our supply and demand charts are actually a byproduct of business data analysis. Perhaps, it is helpful to view the business activities as a byproduct of the analysis of business data, instead of the reverse. And innovation within business is increasingly dependent on accurate analysis of such data, and most importantly on the flow of such data, to produce new product and services that can be profit drivers for our capital enterprises, and even non-business enterprises.

As such, when we reviewed the standard texts and writings in our class work, we have had some objections to some of the concepts. And in that regard, we have evolved our CTTS project substantially differently than the analysis and charting that the vast majority of individuals who do this exercise, both as an academic exercise, and in the real world. Much of the text emphasizes, the need to democratize the process of data system design. To maximize the participation of all levels of work, to chart all the operations and the potential “stakeholders” (a despicable term that itself should be exercised from the political and business lexicon) and to create products which conform to those needs. This form of analysis is doomed to fail and we see examples of this every day, whether it is overly expensive and failing railroad systems that can't handle simple switches in the rain, to banking systems that crash as one waits for the computer to come on line, or the failure to integrate healthcare records to patient interactions in clinics. We see rampant exploitation of our systems by criminals who have real innovation, and slice through the security of our digital services, to steal billions of dollars, to affect nuclear reactors and power stations, to hack into pace makers and kill patients, and so on.

In order to service CTTS properly, we have to do more than simply transform the same activities that our organization does on paper into an electronic format. We need to analyze the processes that CTTS currently does, and change the core working methodology of the people who do these functions past their current understanding of their jobs, to give them more information for independent critical thinking, and to promote responsiveness and forge sales

opportunities that have been previously overlooked. In this regard, the technicians that we send out are not just mechanics who change the brake fluid, but can be retooled as sales representatives of our services, technical advisers of our customers, and most importantly information gathers for our management team so that we can deliver innovative services.

Additionally, we are the experts in information technology and computational sciences. As such, we have a professional responsibility to assure that what we build, and present, is secure. While the initial analysis in conversations of CTTS were scuttled because of a lack of desire to allow confidential information onto the internet, once the insightful decision was made to allow for such access on the world wide infowebs, never again is there a substantial conversation about security of data that is NOT OURS, but for which we have fiduciary responsibility in our acquiring of such data since it is ultimately our clients data. If we mess up our reputation with our clients as to the security and privacy of their data, we will not only destroy our business, but we will also be rightfully so sued for damages, and be open to broader exploitation by hostile actors. We will have failed our basic moral objective.

This in mind, we are now going to present who we designed and responded to the CTTS project, interviews, and analysis and how we reached our objectives and the hopefully enhanced CTTS's enterprise. We have not succeeded, in my opinion, in all the goals stated above, but I hope that our work substantially reaches these goals and more importantly, lays down a foundation for future work, and a change in the corporate culture of CTTS, which in my mind suffers from a little bit of staleness in vision.

We started by adding a 72 hour aging report to the system, which was not discussed in the interviews. This is an example of our IT professionals developing tools that transform work flow and data usage to provide new services and improvement that are marketable and give clients better deliverable. In the initial invocation of our project, the 72 hour report was added to the usage tables had the follow activity diagram, and sequence diagrams.

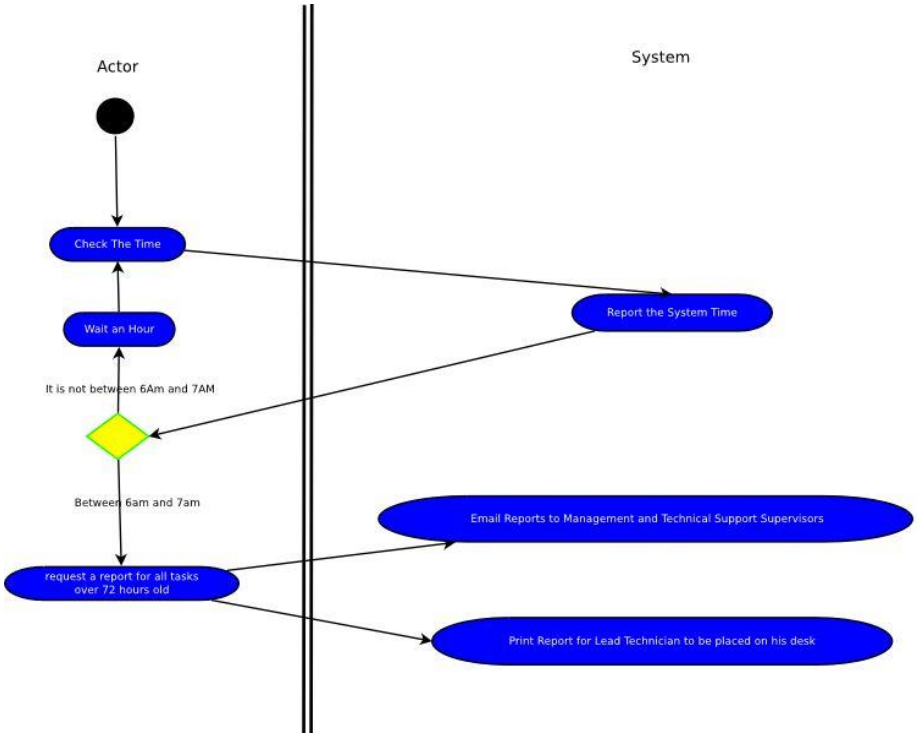

Illustration 1: 72 hour Activity Diagram

The only actors in this area are the system and the program. The management is to receive reports by email and in print. The sequence diagram for the reports is as follows:

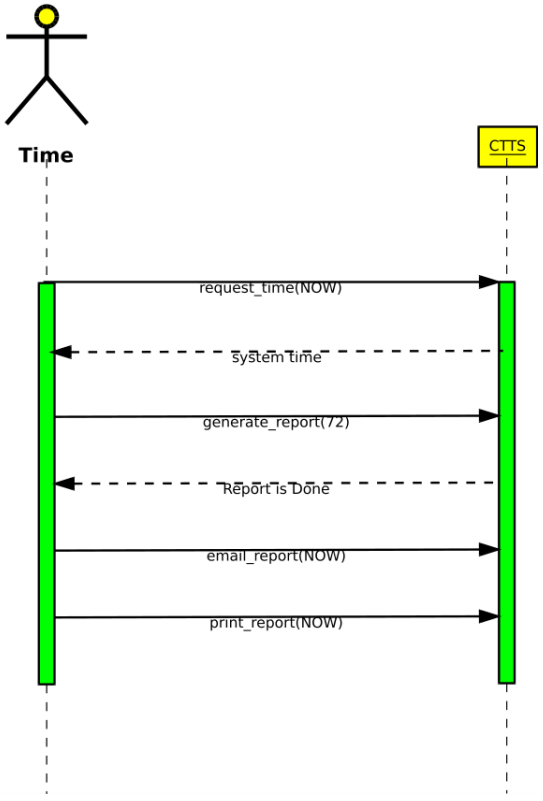

Illustration 2: Sequence Diagram for 72 hour report

Likewise, it was disturbing to hear so much discussion on access times and replication and almost nothing on security. This is an example of where proper leadership by the IT department is simply not being exercised. This project is not rogue. It is to be fully integrated with our overall infrastructure. It is not to be a slapped together, unaudited, application built be the secretarial pool. As such, resources like network connectivity (which is guaranteed to be better that 99.8% uptime by contract on our dual T3 lines by Verizon, and database access and design, which is running on our fully live replicated postgres database engine with parallel back up to our remote site in Edison, New Jersey, is guaranteed to give us nearly 24/7 access in all events short of a nuclear assault. Additionally, our system administrators assure us that our advanced clustering services with 3 remote clustering heads guarantee us protection again all local service outages and out laptops come with G3 wireless backup services assuring us access even when clients local networking is problematic. All of this is way beyond the scope of our project and the discussions of this among non-technical staff should and needs to be ignored.

More important is the failure for our stakeholders to identify a need for program security and system administration. After the initial hesitation to put our proprietary information on the public infowebs, the department heads completely drop the ball in understanding the needs for security for the project. As such, we have a fiduciary responsibility to build such security and system admin interface into the system in order to provide secure and LIMITED access. Our security development started looking as follows:

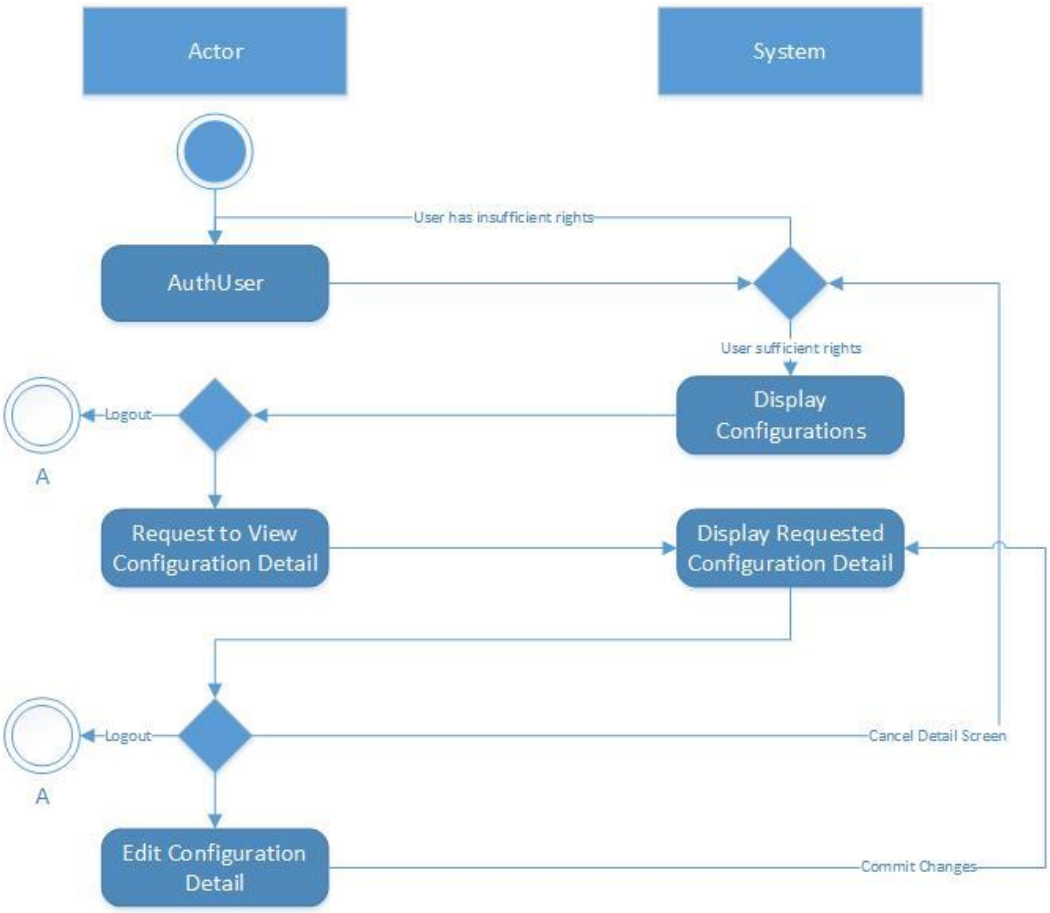

Illustration 3: System Administrator Activity Diagram

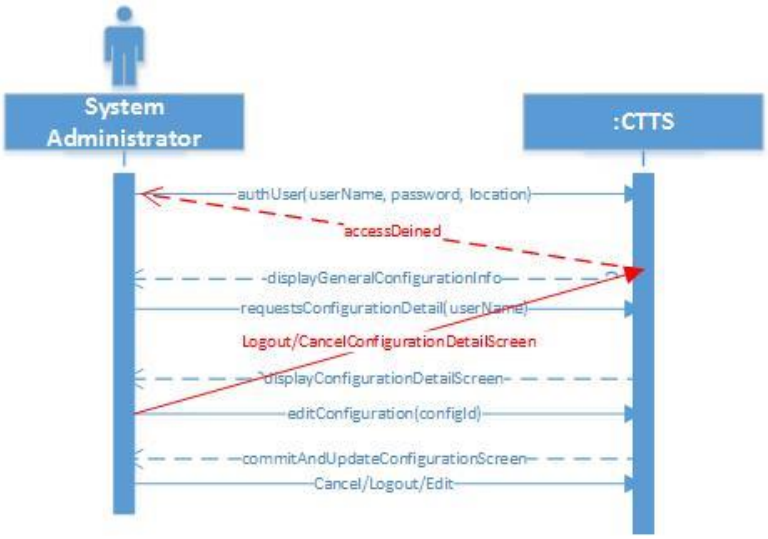

Illustration 4: System Administrator Sequence Diagram

Initial design for the system administration includes the ability to see system settings, and details like user permissions, and then permits us to add and change settings. Users of the software are not users of the operating system that it runs on. They are specific to the application and have no direct access to anything outside of the application, with the exception being this screen which has prescribed access to resources that control our application software.

Finally, in the last use case that we examine we looked at the requests for equipment and component installations. This is the core purpose of the system, and it was pointed out that equipment and were separate kinds of data. On examining the data details, our staff determined this to be INCORRECT. Equipment and Components are exactly the same kind of data. And our application is designed to prevent a confusing and unnecessarily complication of trying to separate the two items. Components and Equipment are interchangeable terms in our design. The initial activity diagram and sequence are shown below, but they had undergone a significant change in the last phase of development as needs became clearer.

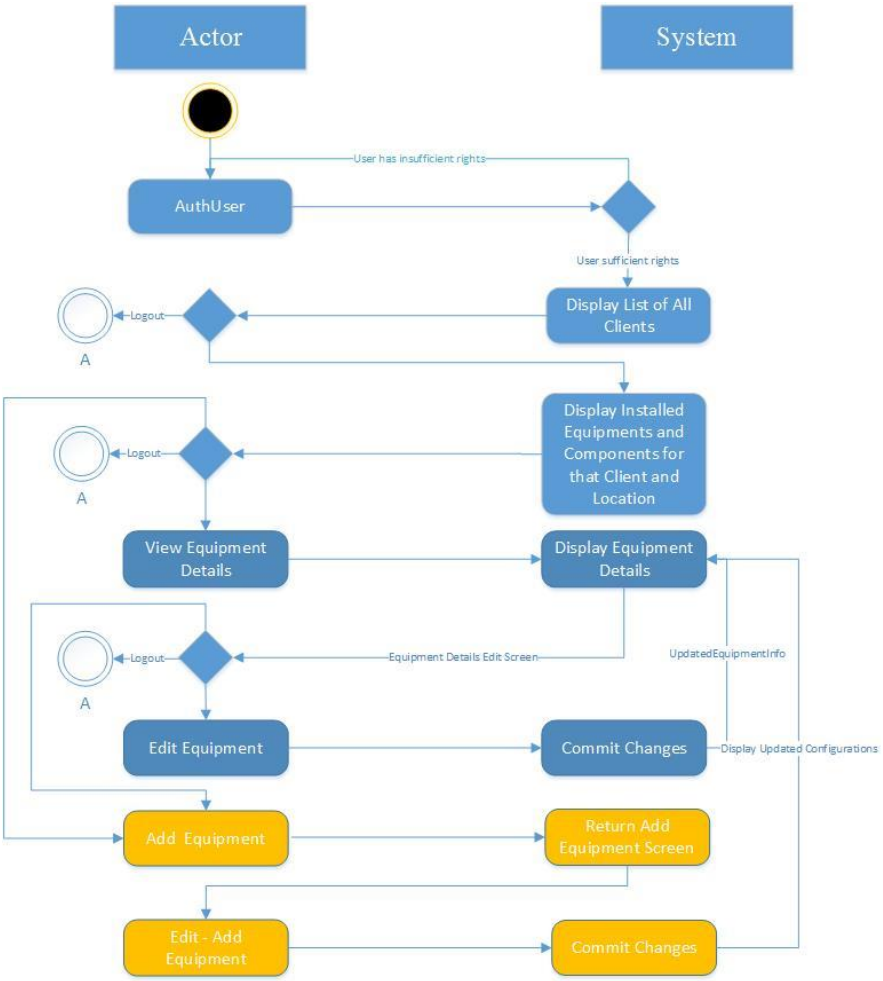

Illustration 5: Equipment/Component Activity Diagram

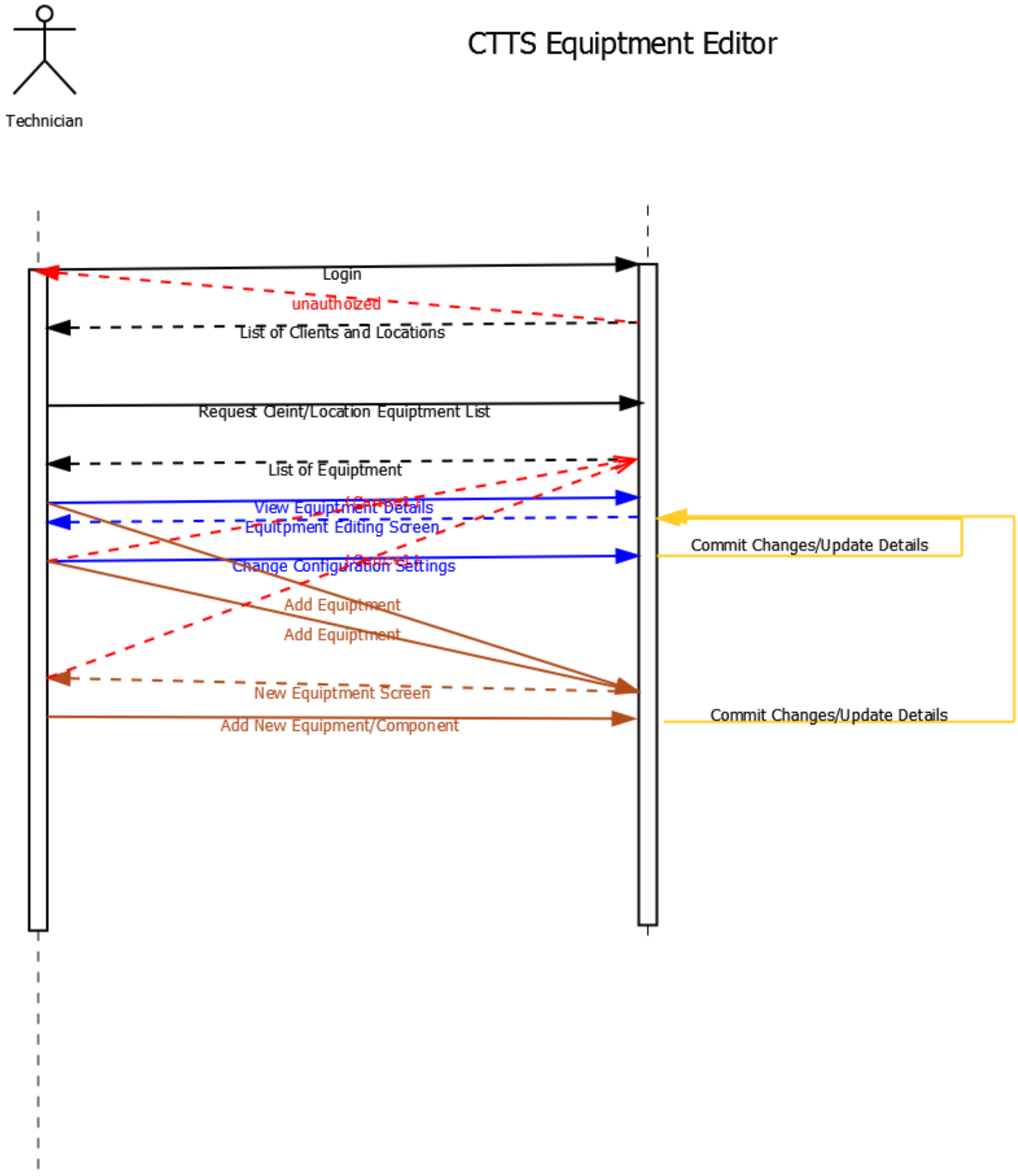

Illustration 6: Equipment/Component Sequence Diagram

Combined, our applications first overall activity diagram was constructed over the developed user activity tables as follows:

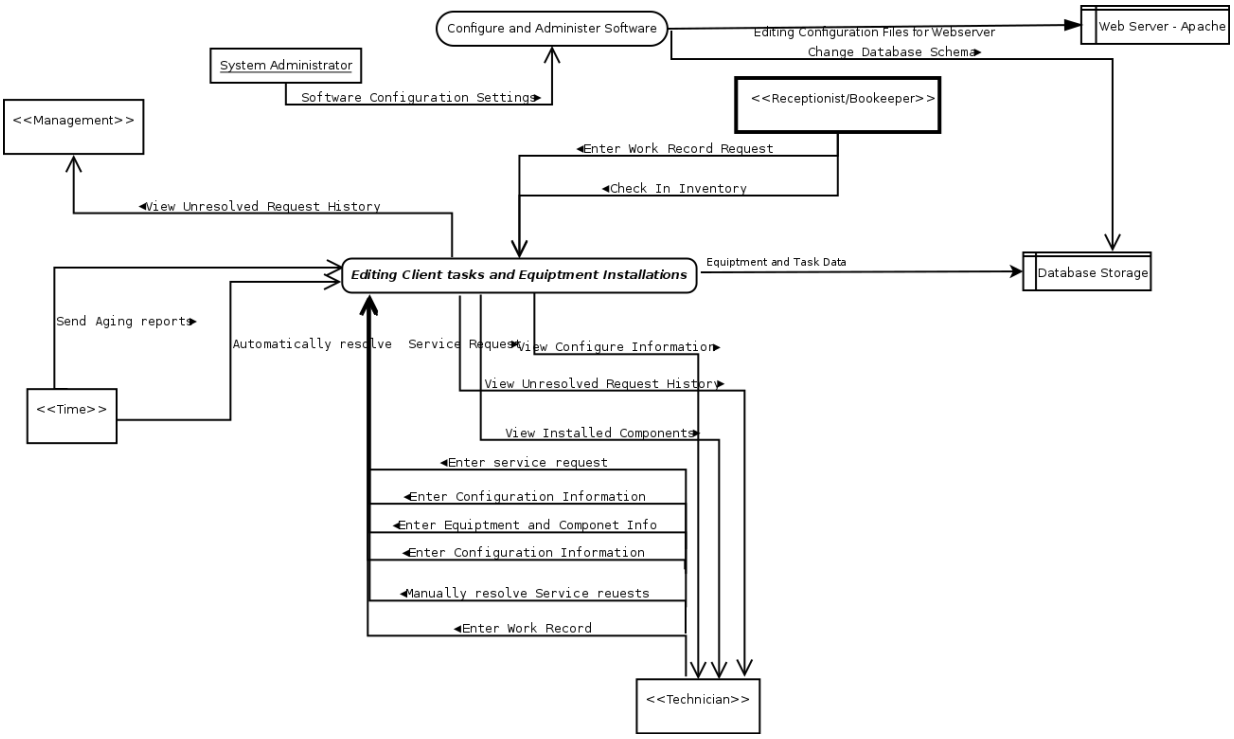

Illustration 7: Overall Activity Diagram

Once we filled out the user case narratives we were able to sharpen the focus and get a class diagram and DFD up and running. For security and business purposes, we had decided it necessary to allow access to request record ONLY through the professional sales staff/receptionist. Furthermore, inventory was beyond the scope of this program, but only inventory control personnel were allowed to put inventory in or remove inventory from our listed stock. This is not, however, to be confused with the kind of inventory control that is done for accounting purposes and fiscal auditing.

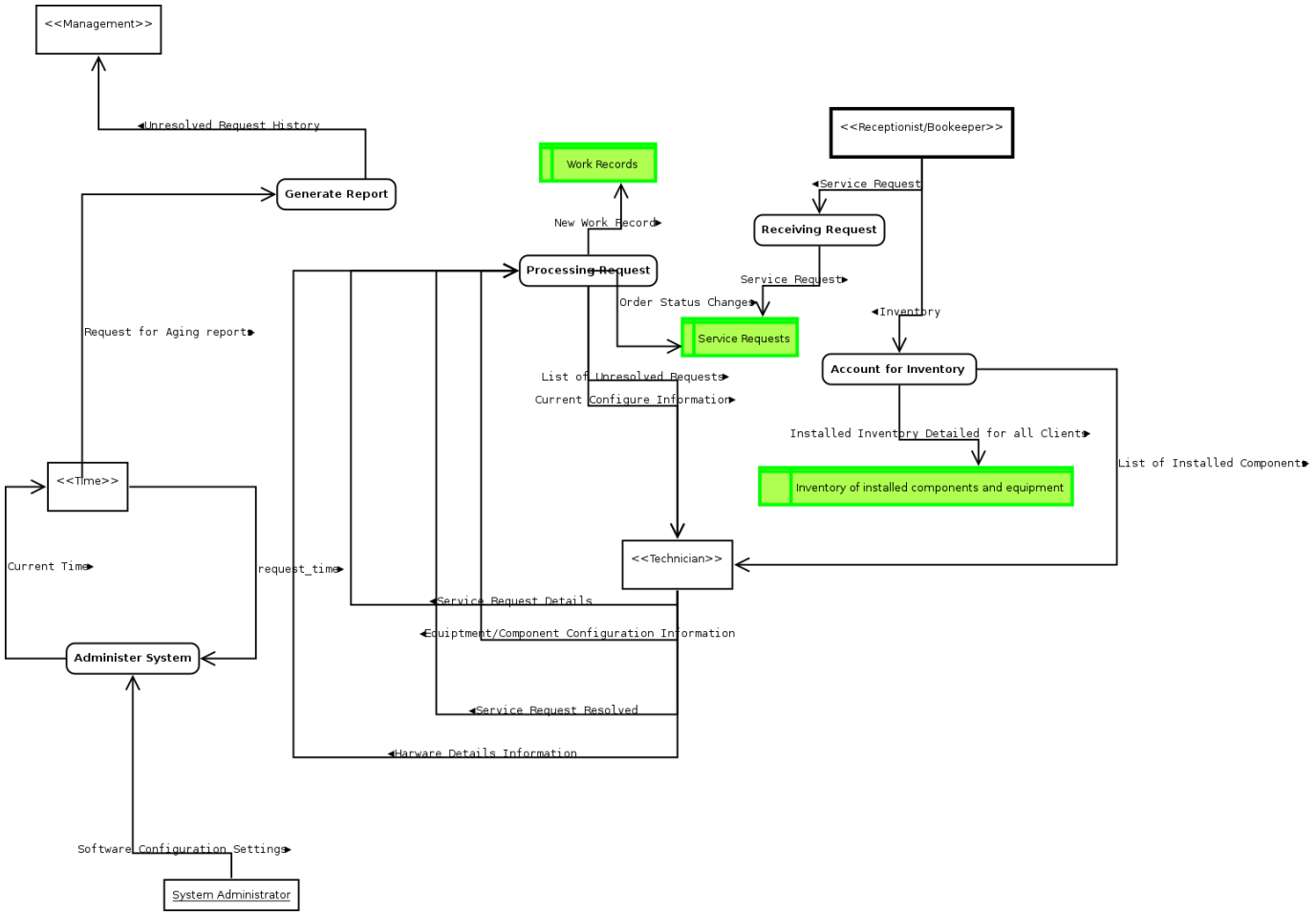

Illustration 8: Data Flow Diagram

Finally, we completed the project design with the resulting Sequence Diagrams, activity Diagrams and Overall Class Diagram. Additional details are added to the interfaces and controllers of the data modules including specific interfaces to the aging report, and the addition of parent and child relationships to the equipment. Interfaces without attributes are reflected in the control, but should be specified in the next and last revision before production.

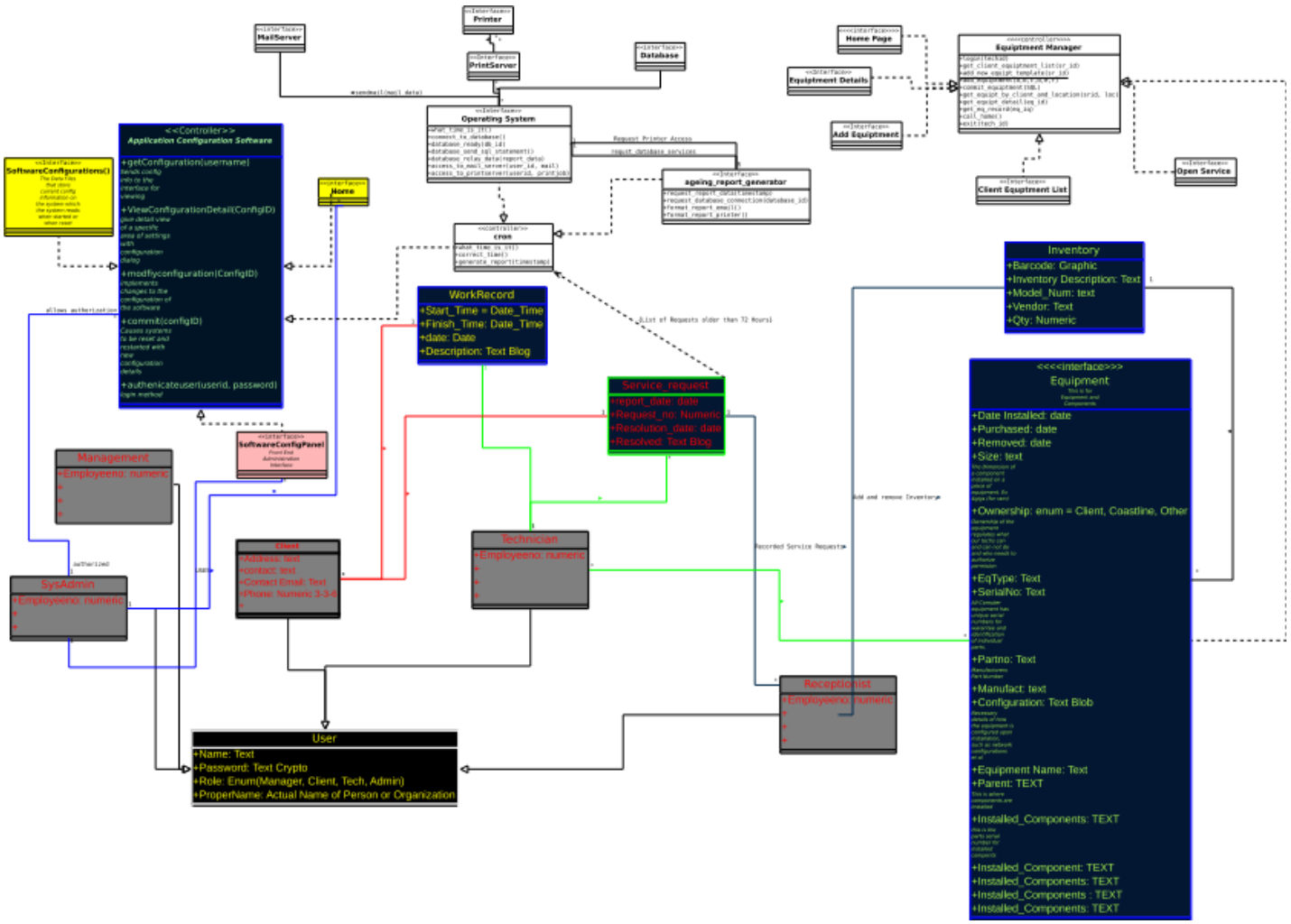

Illustration 9: Class Diagram

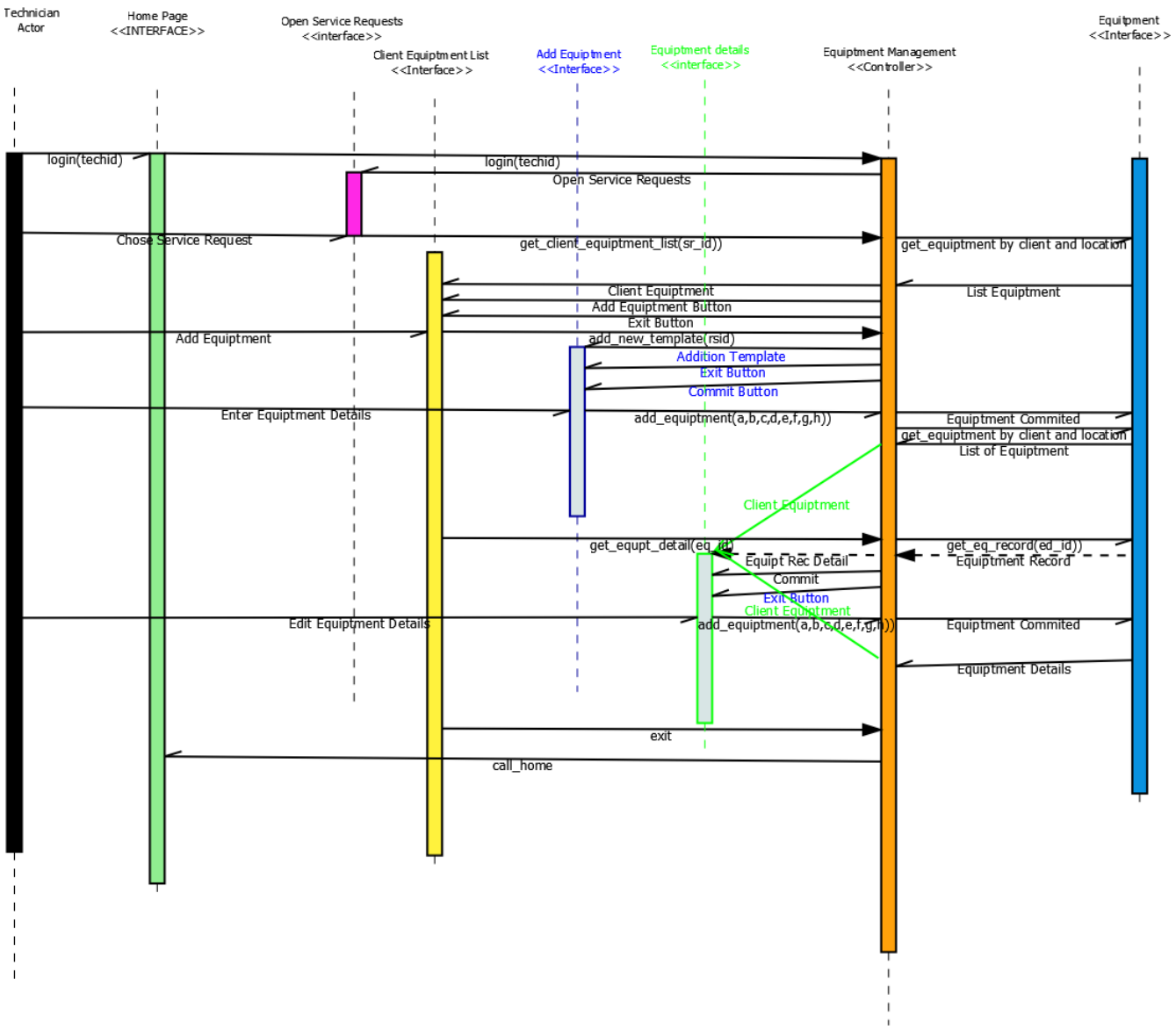

Illustration 10: Equipment/Component Sequence Diagram

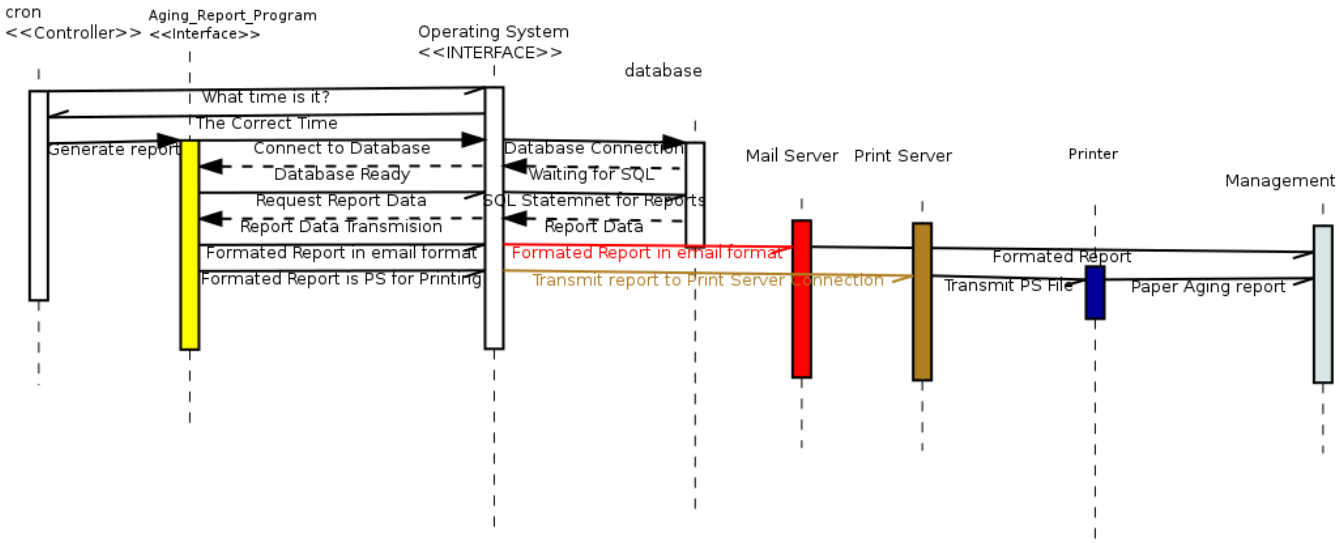

Illustration 11: 72 Hour Aging Report Sequence Diagram

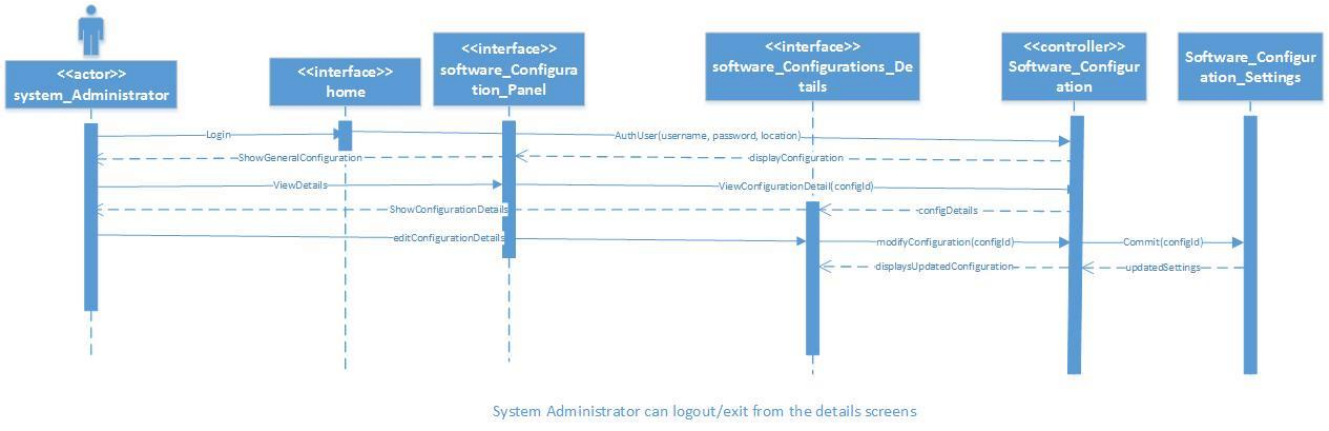

Illustration 12: System Administrator Sequence Diagram