Apollo Guidance Computer History Project

First conference

July 27, 2001

Margaret Hamilton's introduction

MARGARET HAMILTON: When I first got involved in Apollo, I was coming off the SAGE project for the Lincoln Labs. I was a young kid, and I was hired by Dan Lickly over here (pointing to Dan). And I was assigned, this was during the unmanned missions, and I was hired to do what they called programming, they also call it software engineering today, but not back then.

One of my first assignments was to work on algorithms having to do with lunar landmark tables. So I wrote programs having to do with that area. Then there was an unmanned mission that was taking off, and people came running in to tell me that the lunar landmark tables were in upside down and backwards. This was just when the mission had just taken

off. And I believed, until it landed, and heard on the news everything was okay, that I was in real trouble because I was involved with these algorithms.

Then, because I was still a beginner, I was assigned responsibility for what was thought to be the least important software to be developed for the next mission. I was the most of the beginners; I mean, I was the first junior person, on this next unmanned mission. And it was developed for what would happen only if the mission aborted. So nobody really paid much attention to what I was doing, because it was "never going to happen." And I called, I still remember the name of the program was called "Forget it." I don't know that many people really had a chance to even see what was in there since I was pretty much left to my own devices, but when the mission was actually aborting, then I became the expert of the "entire mission" because control in the software had gone to "Forget it". So I had to come in for the emergency. I was called in, and I was the one who had all the answers to all of the questions in "Forget it."

Well, as time went on, the group began to grow and I had people working for me in the systems area, things to do with operating systems, man machine interface, error detection and recovery, and working with and interfacing with all parts of the flight software -- this is the software side nof the mission now that I am referring to, the programming side.

And I learned an awful lot from Dan, who was a real guru in all of these areas. I was trying very hard to learn from him all of the things that he knew that I needed to use in order to be more successful at doing my job. And all of these people kept running around calling everything by their special names. I wanted to know more about the techniques that they were using. For example, they were using the Auge Kugel method, and I didn't know what that meant. But someday I wanted to know what this Auge Kugel method was . I found out much later on that it meant eyeballing in German. That's what they kept using, and that's how they discovered certain errors. So I got involved in operating systems, error detection recovery, putting together everything with what we back then called the glue - how everything interfaced and how everything was integrated.

We began to grow, and eventually Dan put me in charge of the command module software. He had the courage to put me over that whole area, and I got very interested in management of software; again, integrating all of the glue. And when Dan left, Fred then even had more courage and gave me the responsibility for the LM too, in addition to the command module flight software and now I was in charge of all of the on board flight software. Again, I became even more interested in management of software techniques and how we could automate what was at that time manual.

Then I got interested in commonality, for example, the kinds of things that the LM and the command module shared as software, and also how we kept track of all of the things that happened - traceability

Many of the things I was intrigued by had to do with how to make the mission software safe and reliable. And one of the things I remember trying very hard to do was to get permission to be able to put more error detection and recovery into the software. So that if the astronaut made a mistake, the software would come back and say "You can't do that." But we were forbidden to put that software in because it was more software to debug, to work with. So one of the things that we were really worried about is what if the astronaut made a mistake -- We were also told that the astronauts would never make any mistakes, because they were trained never to make mistakes. (Laughter)

So we were very worried that what if the astronaut, during mid-course, would select pre-launch, for example? Never would happen, they said. Never would happen. (Laughter) It happened.

In fact, they went back to read the program notes and we had a program note saying "Do not select PO1 mid course." I was so happy that that it was in there. That was Apollo 8, Jim Lovell's mission. We had the program note "do not select PO1 during flight." I still remember the program note.

So as time went on, I got very interested, even more and more interested in error detection and recovery, because of the errors that took place and how we could avoid them in the first place. We were doing simulation, much simulation, but of course we couldn't test the flight in real time. We had to simulate it, and I got very interested in static analysis.

Especially since we had somebody involved who was a superstar at static analysis - manual static analysis. Was it Norton? He was from TRW. Many of the errors that were found in the flight software were because John Norton, who I guess was responsible for another type of mission, was it Venus? -- [this mission John Norton was responsible for failed because of a documentation error. As a result Norton decided to devote his life to finding errors and not being responsible for any more projects per se] But anyway, he used to eyeball the listings, and he found more errors than were found by any other means, by the Auge Kugel method.

So as time went on, I really had, I guess a sense of history about this entire thing, so I totally understand this mission that you're on. We began to analyze all of the errors that had taken place on the flight software when we were in actually Validation & Verification mode. When each of many of the error reports came in asking for "reason for error," the engineers would fill in a response and they would just say "bug" and that wasn't enough.

So we got very interested in how we wrote errors up, so that if we understood the error, then we could maybe prevent it on the next mission. We did a thorough analysis of the on board flight software, including the errors themselves, and began to categorized those errors. For example, one category is if you took certain steps, it would have been eliminated. Another category, if you took certain steps, it would have been eliminated. That evolved into a theory, if you will, with six axioms, that have to do with defining software in such a way as to avoid interface errors; actually defining any kind of system, whether it be a people system, a software system or a hardware system.

We named this theory Higher Order Software. From Draper I formed a company called Higher Order Software that was named after the theory, and we worked for a long time to build a complete tool suite environment based on this mathematical theory, that if one used the theory correctly, one could avoid a certain class of errors called interface errors. Seventy-three percent of the errors, by the way, were interface errors when we did that empirical analysis. Forty-four percent of the errors were found by the Auge Kugel method. So this gave us the idea of coming up with a means of not just running software or simulating it to find errors, but analyzing the software automatically.

We also found out that 60% of the errors were still in the releases that were ready to fly, but yet they were still lurking in releases way before. They were subtle errors. This is why we keyed in on interface errors because they were of most interest.

So since that time, the theory has evolved and now I actually lost the first company to venture capital people, and started a second company called Hamilton Technologies. Our theory has evolved to something which we call Development Before the Fact, where we have system-oriented objects each of which integrates the functional, the timing and the data side of a system.

And we have a set of products for the life cycle of a system called the 001 tool suite, which is used for defining systems, and developing software. Among other things, it does

things like automatically generate 100% of the code for any kind of software system. The main thing that -- it has been actually all based on our systems language 001AXES which is based on this theory Development Before the Fact -- which goes back to understanding things like interface errors, which goes back to the empirical studies we made back in the Apollo days.

We're writing a book on this theory for Cambridge University Press, and there is an entire chapter devoted to defining what is an error. We chose to make this whole chapter be about errors on Apollo, and what we did in order to understand what an error really was, and how we differentiated between types of errors, and how that drove us to coming up with a way of defining systems in such a way as to avoid these kinds of errors.

So that's sort of, that's what we're doing right now. I must say it's a mission almost as exciting as Apollo, and it's one that I can't stop until it's completed. Even if I wanted to, I can't.

DAVID MINDELL: Could you just way back up for a moment, and say a little bit more about SAGE and your background of how you got into that from the beginning?

MARGARET HAMILTON: Well, on the SAGE system, I was working on a radar system, - in programming systems. What they used to do when you came into this organization as a beginner, was to assign you this program which nobody was able to ever figure out or get to run. When I was the beginner they gave it to me as well.

And what had happened was it was tricky programming, and the person who wrote it took delight in the fact that all of his comments were in Greek and Latin. So I was assigned this program and I actually got it to work. It even printed out its answers in Latin and Greek. I was the first one to get it to work. You can see the overriding theme here; I was and still am very interested in what causes errors and how to avoid them throughout. That was one of my very first experiences in this regard.

When we worked on the Sage system we were over at the AN - FSQ 7. I think it was the very first SAGE computer, the Army Navy computer.

DAVID MINDELL: Who were you working for?

MARGARET HAMILTON: This was for Lincoln Labs. And the thing that comes to mind is the computers that you might have right on your desk here took rooms, and rooms of space. The way we would debug is there would be like a message on the computer which you would take down by hand - it was just the computer's stop location. Polaroid had just come up with the Polaroid camera, and we used to pose in front of it to show that that was our error. But I think from day one, it's been a fascination - the subject of errors that is.

And then Apollo gave us the opportunity to make every kind of error that one could ever imagine. (Laughter) So it was a wealth of experience.

We've written many things from a historical point of view, because we didn't want to let it go in this respect . So we'll be glad to share some of what we did at Draper and later in this regard.

DAVID MINDELL: Was your education in mathematics?

MARGARET HAMILTON: Yes. I started off at a Quaker school, Earlham College in Richmond, Indiana.

JOHN TYLKO: Just after you were asked that, I hope you don't mind the question, how many other women were involved in the Apollo Guidance Computer?

MARGARET HAMILTON: It's interesting that you asked that, because I looked around and it was familiar. (Laughter) But many more came on later on, towards the end.

DAN LICKLY: That's a point that has never been written up, but according to my records and looking back, before 1960 I believe more than half the programmers in this country were females. They were often retrained from other things because the computers were in the big aircraft industries and the other big number crunches, Dolgrans (?) and so on. They often took people who ran Monroe calculators and all of those, and retrained them for the scientists, and they were the programmers. They can't get females to go into programming now.

DAVID MINDELL: I've heard a theory, and that's why I'm interested that you worked on SAGE. I heard a theory from Gwenn Bell, who founded the Computer Museum here, she said she thinks it was the SAGE Project which pushed women out of programming. Because when Systems Development Corporation got the contract for programming and they re-advertised all the slots -- she said she has the advertisements in the archives -- they reformulated all of the slots as men only.

MARGARET HAMILTON: But there were women in the SAGE programming group. I remember there were women, like Dan said, maybe almost as many as there were men. But on the Apollo project I started off with mostly engineers, and I don't remember there, well, there weren't, for a long time, any other women, as I recall.

JOHN TYLKO: Can I ask one more? Did you stay with it through the end of the operation phase?

MARGARET HAMILTON: Yes. We were doing a lot of studies for the space shuttle. We came up with the theory at Draper, the first phase of it. But we were working on space shuttle studies for distributed processing, redundant management, things of that nature, in the sky lab as well, before I left.

JOHN TYLKO: Were you there through like, I guess '75 when the Apollo --

MARGARET HAMILTON: Actually, in '76. We got involved. Actually, we were doing a lot of studies for the space shuttle. This work that we were doing began on Apollo. The theory itself was actually, we came up with the theory at Draper, the first phase of it. But we were working on space shuttle studies for operating systems, error detection and recovery, distributed processing, redundant management, things of that nature, in the Skylab as well, before we left. Before I left.

Fred Martin's introduction

---

site last updated 12-08-2002 by Alexander Brown