NASA JOHNSON SPACE CENTER ORAL HISTORY PROJECT
EDITED ORAL HISTORY TRANSCRIPT

DAVID C. MCGILL
INTERVIEWED BY REBECCA WRIGHT
HOUSTON, TEXAS – MAY 22, 2015

WRIGHT: Today is May 22, 2015. This oral history session is being conducted with David McGill in Houston, Texas, as part of the JSC Oral History Project and for JSC's Knowledge Management Office. Interviewer is Rebecca Wright, assisted by Jennifer Ross-Nazzal. Thank you for coming in and spending your Friday morning with us.

MCGILL: Thank you.

WRIGHT: I'd like for you to start. If you would just briefly describe your background about how you became a part of NASA Johnson Space Center.

MCGILL: I graduated from Lamar University over in Beaumont [Texas] in '68. There's an interesting story that got me here. I'd been interested in the space program since we had one, and in electronics since I was four I think. That's as far back as I can remember. But at the time I was interviewing I had several interviews with people like Western Electric up on the east coast. The head of the EE department—I have an electrical engineering degree—really felt like when companies came in for interviews their roster should be full. I saw that Philco was coming in, and I said, "I don't really want to build refrigerators, air conditioners, it's not really what I'm interested in." But he believed that they ought to fill up, and so just because he asked me to, I went over and signed up.

It wasn't till then I found out that they actually had the contract out here, and I said, "Whoa, that's quite different than I thought." So I certainly accepted an opportunity to come over and interview with them. When I found out how deeply rooted they were in the US manned space program I just couldn't resist.

WRIGHT: It was a good idea then, wasn't it?

MCGILL: So that's what got me over here in June of '68. Now actually the first project that I worked on people probably don't remember. But at that time the United States really had two manned space programs, the one everybody knows about where we were heading to the Moon, but the one they've forgotten about that was called Manned Orbiting Lab [MOL]. It was an Air Force program.

The first thing I got assigned to—mainly because I was the junior guy on the team, so the junior guy always gets to do all the travel—we were integrating together the systems that were to upgrade the Air Force remote tracking stations and also generally building the first version of what we now call the Blue Cube or the Satellite Test Center [Onizuka Air Force Station, California]. Those were certainly going to be used for a variety of things the Air Force wanted to do in space. But the real driver to upgrade those was the Manned Orbiting Lab.

That caused me to spend most of my time on the west coast. Did make it to a few of the tracking stations. Didn't get to go to the Seychelles Islands one, which I always regret not having put a bug in that system before it got shipped. But it was a very interesting time. I look back on it, and if you got to pick a career path, I think that would be a great starting place for

somebody who wants to design this stuff, because it was largely designed when I joined the team, and my job was to sit out there and try to get it all integrated together so it would work.

It certainly leaves indelible impressions on you about the implications of your design decisions. It really did start me out thinking you better design this stuff to integrate or you're going to have huge problems when you try to put it back together and make it play. So I look back on that, it certainly was not by design, but I feel like I was very fortunate to have started out my career in aerospace by trying to integrate a system that somebody else mostly designed.

That lasted for about a year. The country then decided that the smart strategic position was to declare space civilian. So they canceled the Manned Orbiting Lab, the Air Force project, and put all the focus back on the civilian side and going to the Moon, and that's what put me back on the Houston side, working on the NASA things. That's a long answer to your question about how I got started in all of this.

Fortunately back in Houston I was given the opportunity to work on a variety of projects. To set the stage a little more I think of what ought to be our key topical thread today is looking at the architecture of the [Mission] Control Center, how it's evolved. Probably there are not many of us around anymore that remember what the first one really looked like. Remember, this was a thing that was built 50 years ago, mid '60s timeframe, no such thing as personal computers, no such thing as smartphones. A computer was a very big thing and it didn't do very much.

The initial Control Center, which was in place when I got here, by that time it was about '69. In fact I actually watched the Moon landing from Kodiak, Alaska. I was up there at the tracking station at the time. Got fussed at because we decided we'd swing the big antenna around and see if we could hear the LM [Lunar Module] on the Moon. The station commander didn't think much of our plan. Almost had it too.

The Control Center, initial one, was driven by several interesting things. First, I think some of the best innovative engineering I've seen in my career was done by the people that built it. Unfortunately I can't take credit for any of it. But we had nothing. There was virtually nothing in the commercial marketplace to help us get there. So it was all very clever custom hardware design. Even things today that you can't imagine as being some hybrid of electronics and mechanical things. Like one of my favorite stories is the initial display system that drove those small monitors in the workstations. The first one was a thing we called the converter slide file. The way the thing worked is there was a box about 2 feet long and maybe 10 inches wide. In it was a linear tray that held 35-millimeter slides. Each of those slides was mounted in a little metallic frame that had little dents in it that coded the address of that slide.

On those slides was the background for the displays. Background meaning it might say pump A or bus B or whatever the name of that field was going to be. The way the thing worked is the computer sent an address to this box that told it which of those 1,024, I think it was, 1,000 slides in there it should pull up. This little robotic thing would run down there and grab the right one and pull it up out of the tray and move it up to the front and insert it in an optical gate at the end of the box, backlit optical gate. So the backgrounds on those displays were coming off of prebuilt 35-millimeter monochrome slides.

Then the dynamic information in there where you had to say pump A was either on or off or some bus voltage was at a particular value was filled in by a device. I believe it was Stromberg-Carlson that built them. But it was called a Charactron tube. It was basically a double-ended vacuum tube that from one end of it you would fire an electron beam, and inside the envelope of the tube was a metallic plate that actually had the alphanumeric character set etched in it. So you defocused the beam slightly and hit the character you were interested in and

then you took that resulting set of electrons that are making it through the hole that outlines that character, and repositioned it on the other end of the tube to position it in the right place on the screen.

There were two 945-line cameras, 945 lines was a big deal in those days. The standard broadcast system in the US was 525 lines, so we called it high resolution. Today it's very low res. There were two cameras. One of them was looking at the converter slide file, the background, the other one was looking at the output of this Charactron tube. Then you mixed the video between the two and the end result is you get the display with the static part and the dynamic part. I don't know who actually developed that, but I thought that was one of the more clever things I had ever seen.

WRIGHT: What's the timeframe for all this, to make this work?

MCGILL: We were using it certainly in the '68, '69 timeframe. About the time I came over to the NASA side was when we were recognizing that raster scan alphanumeric display things were starting to come along. Certainly not like they are today, but there were fully pure electronic ways to produce those displays. Here in Houston Philco was involved in trying to do some of that, mainly because the Houston operation here was actually a branch office from the Western Development Labs out on the west coast.

The Western Development Labs was heavily involved in DoD [Department of Defense] contracts, including the MOL one I mentioned earlier. They were under DoD contract seeking ways to produce better display technology for a variety of contracts, including the updates to these ground stations that I mentioned earlier. There was already some pure electronic raster

scan alphanumeric display technology starting to be developed here, and it was considered prudent to get away from this very difficult to manage electromechanical display thing. There was only one guy on the face of the Earth that could actually make that thing work, with all the electronic and mechanical alignments that you had to do to get everything to fit just right on the screen. It was very labor-intensive. Not to mention the fact that there was a whole team of people that built all these 35-millimeter slides. Obviously you had to think about everything you want on the display well in advance, and they had to go design it. It had to be very carefully laid out so everything was going to fit back together. Then go through the photographic process to produce the slide itself. It was very labor-intensive.

WRIGHT: Are any of those slides still around?

MCGILL: I don't know. I wish I'd captured one of those converter slide file boxes.

WRIGHT: I have to find out.

MCGILL: There might be one hidden under the floor over there in 30 someplace. There's all sorts of stuff over there. But they were very interesting. If you look around, they're about two feet long, look like they're made out of aluminum. I believe Aeronutronic on the west coast built them, and since they did predominantly DoD contracts, it looks like they'd take a direct hit. They're very rugged boxes.

In fact when I came over to interview they were tinkering with trying to build full electronic displays. One of the things I recall seeing when I came over here was the prototyping

on that. That led to them awarding a contract. Actually Philco bid on it themselves, but they were very careful about organizational conflict of interest, and since Philco was the integrating contractor for the Control Center, they walled off the people who were bidding the display.

I think mostly for fear of appearing to be a little biased, they actually picked the bid from Hazeltine [Corporation], who was getting into the display business at that time, as opposed to their own. Actually I think Philco probably had a better design. But nonetheless that device was called the DTE, the Digital Television Equipment. It was a really big deal. It replaced all the mechanical parts of the thing. It allowed us to generate those displays fully electronically.

It was a little bit challenging. In fact I think it was a little too challenging. Those displays, again going back and thinking of technology, in that timeframe we didn't have solid-state memory like we have today. The only way you really could store things was in core memory. To use core memory to store the elements to produce displays really caused them to have to go extremely fast with the core memory. Probably too fast. One of the issues we fought for years with the Hazeltine DTEs was we would burn up the core drivers periodically, because we were really trying to push them too hard.

But it worked very well. Each one of those devices produced eight channels. We call them a display cluster. The way that mapped between the mainframe computers and the workstation display devices is the mainframe computers would send—we retained the old converter slide file communication format because we didn't want to rewrite the software in the mainframe. It would tell the Hazeltine device what display it was seeking to turn on in the old converter slide file format. So it didn't actually pull up a slide anymore, the backgrounds were stored on a local disk.

Then the dynamic stuff of course was periodically changing, was driven out of the mainframe to fill it all in. The composite video then was sent eventually to the workstations. What we had was the eight-channel clusters, the display generators, a big video switch matrix, and then a bunch of workstations.

We had a TV guide. There was one channel that you could pull up and see what is already up and running. If you wanted to view a display that's already up and running, then when you selected it the video switch matrix simply routed the display that's already up and populated to your workstation. If you requested a display that was not running, assuming there was an available channel in the system, then the system would activate that display and connect your workstation monitor to the output of that display.

You can see there were some fairly significant limitations. You only got eight channels per cluster. I think we got up to 15 clusters, but we never used all 15 of them for one flight activity. We eventually grew into two systems and we would allocate some clusters on one side and some on the other.

There was a limited number of concurrent displays that could be up in the system. But it was still an incredible maintenance and operation advantage over the original mechanical system.

That system existed over there into probably the early '80s. I'm thinking around '82 or '83. Probably '82.

WRIGHT: About 20 years, give or take.

MCGILL: About 20 years, yes. So it served us well. At NASA we try to milk everything we can out of anything we spend the taxpayers' money on. They don't believe that but we really do. Interestingly enough, when it came time that they were clearly worn out, we had to replace them—

WRIGHT: Let me ask you. How do you know they were worn out? Were there some indications that you were starting to have problems with them?

MCGILL: We use the term wearout, but it really has a bunch of things underneath it. If it's a commercial product, which we use predominantly today, wearout probably means the vendor doesn't want to support it anymore. They've gone three versions later, and they don't want to mess with that anymore, so we can't get any vendor support on it.

Back in those days when things were much more hardware-oriented and mechanical in some cases in nature, wearout could mean actually the motors were giving up. The reliability maintainability group would monitor all the equipment, still does. We could see by where we were seeing failures occurring on a lognormal curve that they used that we would declare it in wearout. It's becoming such a maintenance problem that it's more cost-effective to replace it.

The other type of wearout on things that are more pure electronic in nature is you can't get the parts anymore. So if parts fail you may not be able to replace them. That's basically where we were with the Hazeltine DTE stuff. The electronics that it was built with were becoming very very difficult to procure and parts give up and you have to replace them. The real driver was the growing maintenance cost associated with it. Interestingly enough, when it came time to do that I got asked to do the study to figure out how we ought to replace it.

By that time there was lots of display technology floating around. There were numerous vendors building a variety of displays, certainly things had gone from monochrome to color, much higher resolution, full graphics, which the original DTE could not do. It could do very limited graphics.

But interestingly enough, when I walked into doing that study I was predispositioned to think obviously we're going to go buy something. It wasn't until I dug into the details of that that I realized that was the wrong recommendation to make because the interface to that display system was so custom. Remember I said earlier the way it was communicated with went all the way back to the mechanical converter slide files. Of course there was no such thing out there in any vendor's product, absolutely not.

When I looked at the end result of taking some commercial product and modifying it so it could fit into the system, compared to the cost of designing from scratch a functional replacement for the Hazeltine DTE—and by the way, by that time with all the solid-state technology we had, it was trivial, absolutely trivial design—the recommendation I brought forward was let's just functionally replace it. Let's just build another functional equivalent to the Hazeltine DTEs. We called it DGE, Display Generation Equipment. It took two racks of equipment in a cluster for the DTE, and we put it in about five circuit boards, the DGE, to give you some comparison of how much technology had reduced that, it turned into an absolutely trivial problem. There was certainly no component that was under any stress at all in the DGE.

Let me step back and say another couple things about the original Control Center architecture. It was certainly dominated by having to custom-build everything. Almost all the functions done in the Control Center originally were done in custom-built hardware. We had computers, but they were so slow by today's standards, most of the high speed work, like the

telemetry processing, they just couldn't keep up. So we had very custom-designed frame synchronizers and multiplexers and decommutation equipment, and a whole variety of things were all done with actually a fairly substantial group of engineers, myself included, that did custom hardware design.

When I did the study for the replacement—the other thing that dominated that Control Center I should mention, and it may have just been me because obviously I was a real neophyte at that time, but there was no vision of life after Apollo. We were commissioned to go to the Moon, that's what we were fully focused on. We weren't thinking about any kind of a follow-up project. Obviously there were a few things said in public about doing other things. But our mission was to get to the Moon, and so I guess all of us assumed that once we pulled it off they were going to dissolve NASA and we were all going to go find something else to do.

But we didn't think about it very much. Certainly a plan beyond Apollo was not a driver for the design of the original Control Center. It was rather tactical in focus. Not that I think the people who designed could have come up with anything much different even if they had been trying to build a system that had a 20-year life expectancy. But nonetheless that really was not a consideration.

By the time we got into the early Shuttle timeframe and I was replacing the display system—

WRIGHT: What do you consider to be the early Shuttle timeframe?

MCGILL: Probably about '82. We had started flying, but barely, and I was asked to go do the study. That triggered a couple of interesting things. As I said, I wound up recommending a

custom-built replacement. Although by that time moving away from custom hardware seemed to be the appropriate thing to do, the circumstances around that display system were such that it was more cost-effective for the government to go ahead and replace it functionally with a custom design.

When I looked at the world around I said, "Everybody else is doing things a lot different than we are." The original architecture for the facility was a computer-centric architecture. I'll explain myself. I see big systems as falling into three categories. There's those that are computer-centric, there are those that are data-centric, the heart of the thing may be a large database, and those that are by today's terminology network-centric. Networks didn't exist back when the Apollo Control Center was built.

When I did the study to replace the display system I looked. I said, "The world seems to be going to network-oriented distributed processing systems. Displays are no longer generated in clustered display generators. They're generated in workstations." Even though the task I was asked to look at was only to replace the display generation stuff, it became clear to me that we really didn't have the right architecture on the floor.

Understand that from the original Control Center up until the early Shuttle timeframe the architecture did not change, but we'd swapped out every piece in the thing. It wasn't that we had mid '60s vintage equipment in it. There might have been a little of that lying around, but mostly all the key elements had been upgraded at least once, maybe twice. Mainframe computers were probably two models later by that time. As I mentioned, we replaced the display system twice, because it was originally this electromechanical converter file thing with the DTE and now we were going to put DGE in there.

Even though the pieces were changing, and certainly we had a large effort to swap almost all the pieces out for early Shuttle to make it easier to maintain, it became very clear to me we really had the wrong architecture. It wasn't just a matter of modernizing the elements of the system. The system structure was not correct. I'll explain why I came to that conclusion, because this is really I think the important topic today, about architectures and why they get picked.

Some of what goes on there actually is outside of what [NASA Procedural Requirements] 7123 talks about and some of the documented [NASA] systems engineering processes [and Requirements], which I think is important for people to understand. The thing that was wrong with that facility, we had gotten rid of a lot of the labor-intensive hard configuration items. We no longer were making 35-millimeter slides for the displays. Originally all the event lights on those consoles were hardwired to drivers, computers turned the lights on and off.

Every time we tried to change the configuration of the facility you had to move all those wires around, so we had cross-connect cabinets, and we had a herd of people then that had to go in there and check to make sure the right light lit when the particular bit was set. It took thousands of hours to configure the system, which really wasn't a consideration in Apollo at all.

We had started moving away from that. We'd gotten rid of the 35-millimeter slides. We had done a project we called CONIS, the Console Interface System, which provided a bus-oriented communication out into the workstation to turn lights on and off. So it eliminated a lot of moving wires around and checking that. We still had to do a lot of testing to make sure everything was hooked up right, but it made it more of a configuration message that was sent to the console to turn a light on or turn a light off.

We were making real progress on chiseling away at the labor-intensive nature of configuring the facility. But we were sitting there and I looked at our architecture and I said, "Nobody does business that way anymore. Everybody's going to network-oriented distributed processing." At least that's what I thought at that time. It should be remembered that the Shuttle program early on was trying to get to flying about 24 flights a year. They envisioned being on orbit all the time with a Shuttle and sometimes having two up there. This was all before [Space Shuttle] *Challenger* [accident, STS-51L] and we decided maybe that was too aggressive.

But when we looked at that facility, even where we were with the things we had done to make it easier to configure, the best we could do was nine flights a year. The system, the way it was architected, was still hard-configured, was still not very flexible. It took weeks to months to instantiate a different mission configuration in the building.

Here we had a program that wanted to fly two dozen times or so a year. We couldn't do it, absolutely could not do it. So we really couldn't meet the program's requirements. I concluded the basic way we had it structured left over from Apollo was not the way systems were going. So our ability to reach out and touch commercial products would be seriously limited because we had a structure that wasn't consistent with the way the rest of the world was going.

There were a whole lot of factors that got me to thinking that we really ought to go do something about it. Interestingly enough, Philco, probably Philco-Ford by that time, agreed with me. They set aside some of the company money to launch a study in 1983 to go look at that, which we did totally over on our side of the house. We certainly weren't hiding it from NASA, but I don't know that I advertised that we were looking at replacing it, mainly because the paint wasn't even dry on all those pieces that we had swapped out in early Shuttle, and it didn't seem

like a real good idea to go over and tell them, "You know all that stuff we just spent a lot of your money on, we think it's wrong."

Interestingly enough, in '83 NASA came to the same conclusion independent of us. It was one of those phenomena that's hard to explain. But they realized that we had this brand-new set of stuff on the floor and we really weren't structured right. That eventually led into what should we do, and there was a lot of years that went by. Some of it technical, much of it concern from a programmatic standpoint. I'll whiz through some of that. But the conclusion there was that we really wanted a net-centric architecture, not a compute-centric architecture. We wanted to distribute the computing so that we had some unknown number of workstations, each of which was a computer, and all the application layer would run out there. So we had no limitation in the system design then as to how many workstations we could have. Eventually we'd run out of room in the building I guess. But the system didn't really care.

The distribution on the local network was somewhat predigested data but not carried all the way to where it was analyzed. That was also very appealing to the flight control community. One of the things they disliked about the system early on was to get any analytical application produced, it had to get coded and placed in the mainframe computer, competing with everything else that was going on in the mainframe computer. It was not uncommon for it to take a year and a half to two years to get some new application in there, because it had to get fully defined in the requirements. Had to go off and code it carefully and test it to the nth degree because it was going to run on a machine that all the critical functions were running in.

They didn't like that, the operation community tended to have a little shorter horizon they were looking at then than what it was typically taking to get new functions into the system.

Giving them each their own workstation they could play with was quite appealing to them. Eliminating their dependency on the mainframe computer.

WRIGHT: A true own domain.

MCGILL: Yes. It not only had some strong technical basis to go that way but there was a pretty significant emotional driver in the community to get away from that kind of a restriction.

WRIGHT: I was going to ask you earlier about the transitions from one to the other, because users had to make those transitions. So this was going to be a transition that they were probably going to embrace.

MCGILL: Yes, it was. The transition itself is probably a fascinating topic all by itself, because as you might imagine just from what I've said so far, generation two, Apollo being the first generation architecture, the one we're talking about going to here is the second generation architecture, we're on the third one now, I'll describe it in a little while.

It was a very radical change. There were many many things that were fundamental in the new architecture that didn't exist in any form in the previous one. It was right on the leading edge of applying local area networking technology to something as demanding as real-time telemetry processing. In fact I think I jumped the gun a little bit. A little too ahead of the curve. We had a history of very centralized computing going to fully distributed computing. So, it was a very radical change.

I do recall some of the senior NASA management telling me they weren't at all sure I could figure out how to pull off a transition from one to the other, because Shuttle was already flying, and the Shuttle Program wasn't going to stand down for six months or a year while we played with the system.

In fact one of the difficulties in trying to figure out how to do that was the lack of real estate, [Building] 30M [Mission Operations Wing] was all we had initially, and it was full. Trying to figure out how to come up with enough floor space to put basically another system alongside of it to transition to didn't exist. Looked at options about trying to stage it over at Building 17. I looked at trying to get trailers and move everybody out of 30 into trailers in the parking lot to free up as much space. Nothing was going to work.

Fortunately in preparation for Space Station, when we got to thinking about it, we went and built 30S [Station Operations Wing]. So all of a sudden we had some real estate to work with. Otherwise we'd have never pulled it off.

Knowing we had to transition rather seamlessly was a major system level design driver. I'm going to go back and recap some of the things that are really important considerations that you can't find written in Level A requirements, that you don't really find in the classically documented engineering process.

I'll explain exactly what I mean by that. We were looking for a system that could be configured quickly. The closest I could come to that is I did write a Level A that said we had to be able to transform the system from one activity to another in an hour or less, which seemed incredible at the time. Today we do it all the time. That one could be a real requirement. Many of the things we were after are rooted more in what I call needs, goals, and objectives that in engineering terminology can't be requirements, because you can't test them.

If I tell you I want a system that can remain viable for 20 years, you're going to tell me that's not a good requirement. I'm not going to wait around for 20 years to try to sell this thing off to you. Those kinds of desirements are difficult to deal with in the classical systems engineering process.

We had a lot of those. Many of the things that we wanted. For example, we wanted to have fully portable software. How many times do you have to port it before you can demonstrate you complied with that requirement? How many years will that take? It's not a testable requirement. Very difficult to test if at all.

Most of the real targets, the behavior that we wanted out of the new system, really weren't deeply rooted in standard engineering processes.

WRIGHT: So you were rocking that boat, weren't you?

MCGILL: Yes. I still am. Which is one of the things that I hope comes across in this material, because I'd like for people to understand there's a lot more to consider when you're building these systems than simply meeting the Level A functional requirements.

The other things that were important to us at that time is I was convinced that we were on the brink of being able to eliminate all the custom hardware in the facility—we almost did it, didn't quite—but that there was enough out there in the marketplace for us to go buy, not software but hardware. Which eliminated an awful lot of the cost of ownership. As you can imagine when it was locally designed and built hardware, then we had to do all our own maintenance, board level maintenance. You couldn't call a tech to come in from the factory and do it, we were the factory. There was a huge labor force just to keep all that custom hardware

working, not to mention the labor force it took to design it in the first place. I thought that the technology had gotten to the point that we could put the new system together largely with commercial hardware.

Even on the software side, I felt like that we could go with commercial operating systems. In particular I wanted to pick one that had fairly wide industry support, because recognize, hardware does wear out, and you have to replace it, and you don't want to rewrite all your software every time you do it, because remember, software portability was one of my targets as well.

We went with POSIX [Portable Operating System Interface] as the standard that we were going to use—today we would call it UNIX probably—because many vendors were supporting it. I knew that whatever hardware models I went and bought, within seven years or so, I was going to replace them. They were going to be in wearout, the vendor didn't want to support them anymore.

I wanted to buy some insulation against the hardware, because the downside to going and buying COTS [commercial off-the-shelf] hardware is you can't fully control your own destiny. If you design and build it here, you can maintain it as long as you want to, as long as you're willing to pay the annual price to do it, because you're sustaining it yourself. You're not at the mercy of the vendor to do that.

Knowing that when you go COTS, you got to be prepared to swap that stuff out on a fairly regular basis, the right thing to do is to make sure you put all your custom intelligence of the system in in a way that you've bought as much insulation against the underlying hardware platforms that you can, so you can replace them whenever it's prudent to do so, and you don't have to go rebuild all the software.

We picked a POSIX kind of an operating system structure. Which I was told by many people there's no way you can do real-time processing on POSIX, but that's turned out to not be true. Although they did scare the hell out of me, I'll tell you, they really did. Because some of the people who were telling me that I had a lot of respect for.

WRIGHT: Was that in house or industry people telling you that it wouldn't be able to do real-time processing?

MCGILL: Really both. There wasn't very many places that that kind of real-time processing was being done at all. Even going back, the original, the real-time operating system we ran in the IBM mainframe computers, IBM built it for us here. Eventually they marketed it and called it RTX. But most of the time computers were checkwriters. They were being used in the financial marketplace but not flying spacecraft. They weren't rich with the kinds of resources that you really needed.

The UNIX kind of OSs were really designed to predominantly support a software development environment. It did things, for example if you had 50 users logged on the machine trying to do software development and compilations, it sliced up the machine evenly across everybody trying to use it. Really didn't make any provision for prioritization of tasks. It was in the basic personality of those operating systems to not give you a rich set of features to handle the fact that some things are more time-critical than others.

There was an element of truth in what they were saying. I still felt like we could put enough horsepower behind it and make it work anyway. The benefit of getting something that

was pretty much industrywide supported outweighed the difficulty that I might have in getting it all to play right.

We were trying to accomplish a lot of things there. There was a fairly long list of targets, many of which you could not represent in the documented requirement set, because there was no reasonable way to go test the system at delivery time to prove that you met that. So, most of the drivers were really hung out over here on the side.

But we did produce it. As I mentioned earlier, and I'll give you a couple of examples, I knew from the beginning that I had to put something together that we could transition. We were not going to be between programs. We were going to be in the middle of a program. We couldn't stop supporting the program. There were features designed into the system that would allow us to incrementally morph from the old architecture to the new one. Maybe that's a slight overstatement.

I'll give you some examples. In the old original architecture we had some frontend processors. We called them TPCs in those days, telemetry preprocessing computers. They did the first level of decommutation on the downlinks for example. They decommutated the downlinks for us and ordered all the parameters up in a buffer. The buffer then was sent through a very custom set of communication stuff that we called the MBI, the multibus interface, that provided the communication path between the frontend processors and the IBM mainframe computers. Then the IBM mainframe computers did all the magic to the data. They were responsible for doing all of the computations for analytical purposes, generating all the displays, and driving the display system.

The first step in the transition was to replace that very custom communications thing between the telemetry processing computers in the front end and the IBM mainframe computers

with a local area network. Fortunately we could go get Series/1s, which was an IBM product that we could hook up to the mainframe computers, and with a little magic in them we could make them learn how to talk on a network. We went and built a device called the NDD, the network data driver, very cleverly named, that functioned like the TPC did, but it knew how to talk to a network.

We switched first so we started trafficking the data between the frontend processors and the central processor across a local area network. That did two things. One, it put the network in place so now we could start talking about hanging workstations off of that network. But it also eliminated one of the very custom pieces of equipment that was hidden in the middle of it. Since it was transitional, the functional allocations between the processing on the front end and the mainframe didn't change. The frontend processor produced what I called an NDM, a network data message, that was raw in that it was not calibrated—it was the actual bits that came down from the spacecraft—but it was decommutated. We pack those things in there as tight as we can, because we're always very bandwidth-limited on the RF [radio frequency] links. There's actually a bitfield in there. You got to sort it all back out to identify the individual parameters.

The NDM was a raw form of the message. It had been decommutated and the parameters identified, but they were not calibrated. Trafficked them over to the mainframe. It did the same job it had done before where it handled all of the polynomial calibration on the parameters as well as did all the computations with them.

It was expected at that time by me that eventually that message would go away. That was an interim step because eventually I was going to move all the calibration functions over to the frontend processors. Remember, the mainframes were going to go away, at least in my plan.

It turned out by the time we got to that point in the transition where it was going to pull the plug on the NDMs, I had already replaced them with what I called a PTM, a processed telemetry message. And so it carried the fully calibrated version of it, calibrated in the frontend computers. When I got ready to get rid of the NDMs, the flight control community said, "No, no, no, no, no." They wanted the raw values too, so I said, "Okay, we'll just let them run in there." But that was supposed to be a transition-only element in the design. It turned out it stayed around all the way through Shuttle. But who cares? We already had it in there.

But, we did things in a particular way so we could incrementally shift things over. The actual transition from old to new was incremental as well. We had both systems up and running in the building. We would first switch over and go do some routine Shuttle on-orbit ops with the new system, but we flew all the critical phase things like ascents and entries with the old system, until finally we had a high enough comfort factor that the new system actually worked that we were willing to fly an ascent with the new system and declare transition complete at that point.

WRIGHT: Do you remember what mission that was?

MCGILL: No, I do not. It's probably well documented someplace and probably remembered by other people. But it was probably in the '95 timeframe by the time we actually pulled off the transition.

It was incremental in two ways. One, we didn't ask the flight control team to show up in a different place one day and just pull the plug on the old one. But it was also incremental in the way we designed the system so we could morph functions over from one to the other so that things appeared to be under control all the way. But it did influence the design. Certainly as I

mentioned the NDMs stayed around for a very long time. They're not there now because we're not flying Shuttle. But still even on the Station side we have RIMs [Raw Information Messages] and PIMs [Processed Information Messages] which mirror the old NDMs and PTMs, the raw telemetry messages and the processed ones. There are artifacts of having to do that transition in what we now call the legacy system because we're fixing to replace it again.

Again you can't write those in a requirements document. One of the key things in really understanding this architectural engineering is figuring out how you're going to get there from here. Obviously if you can't get there from here you've got a failed project. But also to understand what it's going to mean to try to put an architecture in place that you think you can live with for 20 years. I'm going to say some more about some of the elements of that that I think are very important.

You're not going to get a chance to replace the architecture very often. That takes a lot of money. Seldom do the planets align and congressional—

WRIGHT: Benevolence?

MCGILL: Yes. I was looking for a word. Allow you to do that sort of thing. Longevity is a very very important element in producing a design at the architectural level. Let me define what I think architecture means. That's an often used and mostly abused word I think.

To me the systems architecture is talking about its behavior, especially its behavior in the face of change traffic. A system level design, you can hand me a stack of functional requirements, Level A requirements, things the system has to do. I could go draw you up probably half a dozen different structures that can meet every one of those requirements. In fact

some of them would be compute-centric, some of them might be data-centric, some of them might be network-centric. But they functionally will do everything you've asked me to do. But over time they're not going to behave the same way. One structure is going to be able to capitalize on things like network bandwidth or speeds just becoming better and better and better without us spending any money on them, the technology is growing, computers are becoming faster, they have more memory in them.

Some of those structures would lend themselves to that kind of plug-and-play swapout better than others would. To me when you talk about the architecture of a system, you're really talking about how well is it going to behave itself for the next 15 or 20 years. Some of that change traffic you probably can see. Some of it you probably can see. Some of it you cannot. A key element in a long-living architecture is building in some very carefully thought out modularity and flexibility. We like to call them pinch points.

What you really want to do is decouple things in clever ways. As I mentioned before, we have what we call ER [equipment replacement] program. We typically try to swap out the hardware stuff about every seven years. Some things live a little longer than others. You know over the course of architecture that's going to last you 20 years you're going to change all those pieces out several times. How do you do that without disrupting the rest of the system? Because you don't do them all at the same time. The money is not there. Each fiscal year you've got budget to change out a few things. You have to have a certain modularity so that you can actually sustain the system, you can swap those things out without disrupting operations, without forcing a massive redesign on other parts of the system.

The modularity is also driven by things like if you look out there you can see that certain technologies are going to advance because the commercial marketplace is pouring money into

them. A common mistake, an example of it I think was in early Station. There was thinking initially that they would go ahead and run a 300-megabit downlink on the Ku band side of Station early on. Remember, that would get downlinked to White Sands, New Mexico. The question in my mind was how are you planning on getting that from White Sands, New Mexico, to Houston, Texas? Remember, back in those days, communication satellites was all you had.

The people I was quizzing on that said, "Well, there are 50-megabit-per-second transponders on communication satellites today. By the time we need them they'll be 300-megabit."

I said, "No. The commercial marketplace is spending their money trying to figure out how to slice up one 50-megabit transponder to service a half a dozen different customers. They're not spending any money trying to build 300-megabit transponders. They have no market for it. If you take that position you're probably going to wind up having to build and deploy your own wideband communication satellite, which I don't think you've got in the budget right now."

It turned out the problem got solved eventually because everybody starts sticking fiber in the ground. Now we have lots of bandwidth from coast to coast in the US with all of the fiber optics that's in the ground which is what allowed us to step the Ku band rate up recently to 300-meg and we're fixing to go to 600-meg. Satellite communications is not the way we get it in; we get it in with fiber.

One of the key things is when you look out there you can't just assume everything's going to get better. The speed of light is not going to change. If you're going to count on something improving you better have identified the non-NASA commercial drivers that are going to make that happen, because we don't want to have to go invent all the stuff ourselves.

We want to be able to buy it. Some things, you can see those drivers, and you can count on them, they will happen naturally, and by the time you get to the next ER cycle you'll have better things available you can go buy. Some things will not. Some things are not going to get better with time because there's nobody wants it but you.

WRIGHT: Can you give me some idea? How were you able to keep up with what all the industry was doing to give you those pieces of information to be used? Because you didn't have a lot of the access to information like we do in 2015 back then.

MCGILL: We didn't have Google. You couldn't Google things like we can now. But still there were enough periodicals floating around. You could pretty well tell where the networking industry was going, where—I won't call them personal computers but much more personal than a mainframe certainly—workstation class machines were going. You could look at the trends and the projections in the semiconductor industry about what they thought would happen periodically in terms of the speed of processors. There was plenty of indicators out there to give you clues about which things were going to improve for you.

It also gives you some insight into—when you're putting the thing together, and certainly the second generation architecture is an example of it, the things that you would be willing to be right on the hairy edge because they're going to be self-improving, they'll fix themselves for you. The initial instantiation of the network-based second architecture system barely had enough bandwidth on the local area network. In fact it really didn't. I compromised the design some to make it work, but it was right on the hairy edge. I knew from what was going on in the industry that you wait a few years and that's going to get really fast. Then by the time we get ready to

replace it, that won't be an issue anymore. You're willing to take the chance there because you know the situation is going to improve.

There are other categories of things where you might say, "I'm willing to be wasteful in this category because there's a benefit. The category I'm being wasteful in will become less and less of an issue because the marketplace will fix it." For example when we initially put together workstations the amount of memory that was in those things was very limited. Amount of processing power was very limited. And I was willing to get on the hairy edge on the processing power to do a whole lot of graphics in there because I knew that was going to fix itself with time. Without us spending any money those processors were going to get really fast.

You can make decisions where some things you can project out with a fair amount of accuracy. You may not get the timeline exactly right, but you can tell where things are going to go just because the industry is driving that way. Not that you're foolish enough to think you're going to drive them. So, you can make decisions that say that we can be a little at risk here because it's going to fix itself with time. I can be wasteful over here because there's a benefit to being able to display data more graphically, for example, even though I'm really taxing the processor a lot. But that'll fix itself with time.

Some of those things are fairly easy to see. But there's certainly a category of them that are not so easy to see. We're sitting here today looking at exactly that kind of a future. As I mentioned, we're on the brink—I say on the brink. We actually started doing simulations for certification Monday this week on the new system. It's rolling into operations. Within a matter of a few months it will be flying the Space Station for us.

It's the same MCC. We only have one. Now we call it MCCS because we've integrated some of the peripheral functions in with it too, so it's Mission Control Center Systems. But

MCC-21 is a project. It's the latest one where we're pushing the thing up to the third generation architecture.

Why do we do that? What's different about the third generation architecture compared to the second generation one? What were the reasons why we thought a change was necessary? I'm going to throw a term in I think nobody in the world uses but me but I like it. I firmly believe in the phenomenon that I'll call architectural wearout or architectural obsolescence.

Obviously you can go use lognormal curves or some of the other things that are used to predict when equipment is going to wear out. But what that really means is when you look at the state, the configuration of the system you have on the floor, and the amount of money you've got invested in it, and what it would take to replace it, and the operational requirements that you're seeing now, you're trapped behind the power curve. The world has shifted enough on you that all of a sudden you can't really support it well. I mentioned even going to the second generation architecture some of those things.

Shuttle wanted to fly 24 times a year. We couldn't do it. The architecture wouldn't support it. So I would contend that coming out of the Apollo era going into Shuttle we were in architectural wearout. The one we had on the floor was dead. We could not get it to go where we needed it to.

Today we're seeing a variety of things that are shifting. First off, we believe when we look out there with commercial players involved we look at much more diversity in the kinds of missions we think we're going to fly. We were used to having one or maybe two single big programs that went on for 20 years. Now we think we're going to do one-offs. In fact we already have. EFT-1 [Exploration Flight Test-1] was a good example of it.

We see the need for a different kind of flexibility. We also see the need for the capability to be a viable participant in geographically distributed operations. Where today if you look at our history MOD [Mission Operations Directorate] was it. We did the whole mission start to finish. The Control Center that's on the floor was designed as a closed shop Control Center. Now we did stick Band-Aids on the side of it to support international partners, but that really wasn't a driver when they designed that one. In fact Larry [Lawrence S.] Bourgeois was our MOD rep to the ISS [International Space Station] Program early on. He was quite emphatic that we were not going to do element ops. If the Japanese [Japan Aerospace Exploration Agency, JAXA] wanted to play they could come sit in our building. If ESA [European Space Agency] wanted to play they could come sit in our building. We all know that's not the way it came down. But the system was not designed to support geographically distributed operations, and so it's been a little bit of a stretch to make it do it. But we see our future as necessitating that all the way around. If we go off and do some sort of a Mars mission, you know that's going to be an international event. It's not all going to fly out of [Building] 30. Other geographic locations are going to have certain responsibilities in that.

We know in the interplay we already are participating in with commercial vendors that that's the nature of the beast. The new MCC-21 architecture is explicitly designed to support that, which was not a factor in the second generation. In fact the Agency did us a huge favor—coincidental but I accept a gift anywhere. They had decided about the time we were looking at needing to do this to collapse out all of the Center level network domains and go to NDC, which is the NASA Data Center domain.

Prior to that Marshall [Space Flight Center, Huntsville, Alabama] had their little network and JSC had their little network. There were cross-connects across the Agency, but it was not

one domain. By going to NDC, which is NASA-wide, one domain, that gave us a very convenient mechanism to exchange data between here and other Centers.

Most of the participants that interact with NASA, even if they're not a NASA Center, have NDC accounts and have access to NDC. That gave us a very convenient mechanism to build on. We changed our strategy not only from MOD flying the whole mission to being willing to work with other operational participants, but also even on the JSC campus of the role of MOD versus IRD [Information Resources Directorate].

When I looked out there I saw several interesting things. The Agency had invested significantly in what we call our corporate backbone, e-mail and WebCAD and all the things that go on to make the organization work, where certainly not very many years ago the capability of those systems was not suitable for flying manned spacecraft. The criticality was too high. Reliability was not there.

But today with the investments the Agency had made, they were very very close to the kinds of performance that we were used to for flying spacecraft. Certainly for the less critical side of our operation they were there. So, it seemed economically prudent to jump on their bandwagon. Plus the fact that they had provided us the very convenient mechanism for exchanging data with other NASA Centers.

We split our architecture. At the very top level of the new architecture we have a side that we call the high side and a side we call the moderate side. The terminology used in there, we have MCE, which is the Mission-Critical Environment, and the MSE, which is the Mission Support Environment. We split it that way probably for two reasons. There's a balancing act, which there seems to always be. One is I said the MSE side is sitting on the NDC domain. So it

exposes those services in ways that it's very easy for us to give accounts on the system to people at other Centers. It's there.

But the downside to NDC is, remember, it's hooked up to the Internet, and people need that to do business. So it's a little bit public. So to try to get the best balance between making our services available outside of our little world and yet providing the kind of security mechanisms that we really need to protect ourselves, we split the system.

MCE, the Mission-Critical Environment, can be isolated away from the MSE, so that if we're having a very bad day and NASA goes under attack, that does not compromise the vehicle or the crew. We have a safe haven. We have the right switches that we can throw, and we will not be corrupted by whatever might be going on on the more public network.

Of course at the expense of some of that intercenter traffic, but if you were in hunker-down mode, that's probably not the number one consideration. So we split it that way to get the best balance we could with guaranteeing that we provide the level of security that the country expects from us to manage these very expensive spacecraft, but at the same time build that element into the architecture that was missing from the second architecture to conveniently support interactions with people that don't live here.

The other thing that we changed in the system that's a bit subtle, but very important, is we decided since the first one I mentioned was a compute-centric architecture, the second one was a network-centric architecture, we decided to go with a data-centric architecture. The reason is again even back when we did the second one we were flying Shuttle. We did real-time operations. The system is really designed to be a time now real-time system. That's what we did mostly. Now the engineering evaluator guys worked in a different timeframe but they'd come get stored data and go off and analyze it, but it was a time now system. When we looked out in

the future with the desire to go play with asteroids and go to Mars and flight time delays were going to get very long through the laws of physics, I can't fix that. All of a sudden it didn't look like time now was the right answer. It looked more like a TiVo kind of a thing where you needed to be able to slide a little slider up and down and say, "Well, here's what it was doing yesterday, here's what we think it's doing about right now, here's what we think it'll be doing tomorrow based on our spacecraft models." All of that obviously implies that I'm interacting with storage, with a database.

We believe the best way to posture for the kinds of missions we think we want to fly at least—again this is speculative, it's not really written in the Level As, but planning for the future—is to switch to a data-centric system. So, we have a thing we call Ops History. We gave it that name because that solves a real problem that we looked at. I probably should have said some of the changes always are not because the world has changed around you. Sometimes you're changing things because you say, "Boy, I wish I could have done that the first time but the technology wouldn't support it but now it does, so I want to change that." Some of that are just flat lessons learned saying, "Boy, that wasn't a good idea, was it?"

One of the things that I felt like though we missed the mark on on the second generation architecture was a lot of data got generated in the system that the system common services did not manage for the user. They were just left on their own. They could write over their files, and they did occasionally. That was a real mistake. There really was no complete auditable data set that was kept by the system for them. When we put the data-centric element in it we called it Ops History. We think we have largely fixed that where it's a very complete set of everything that was available while the mission was going. The raw data, all of the computations that were

done, all the analytical results are stored in there, and configuration-managed for users automatically by the system.

Part of that was lessons learned. Maybe a little technology in there, although file systems were pretty good by the time we built the second one. Really we didn't go to the third of the three categories just so we could play with all three of them eventually. We went there because we think that is the structure that best suits our future. Some of it is speculative. But we wanted a posture so that if a new administration might suggest that we could go play with a Mars moon then we were positioned to be able to do that. Our system can support that quite conveniently.

Other changes that we made architecturally. As I slightly mentioned the downside of opening it up is security. As long as you're air-gapped the way we were basically, you don't worry about security too much. Nobody could get in, the door is locked. When you start making the services available outside of your controlled access areas, however, you better worry about security.

We made security in the system a number one priority in the new architecture. Security is not a Band-Aid that we stuck on the side of it after the fact, which is what it was largely in the second generation architecture. It is a primary part of the way the new architecture is put together. And it goes beyond just making sure bad guys don't get into our system. It deals with again another artifact of where we think we're going. There are different levels of sensitivity of the data. For example on ISS there's some elements in the downlink that are considered ITAR [International Traffic in Arms Regulations]-sensitive that we don't just give away to a foreign national unless there's been some agreement made.

There are other elements in the downlink that are medical private. For example any spacesuit data. There's legislation that we have to guarantee we don't distribute that

inappropriately. But in general in the second generation architecture all of that was dealt with procedurally. You're not supposed to look at that, so don't. But there was nothing in the system that really prohibited somebody from looking at something they weren't supposed to.

With the new system we've built all of those data protection mechanisms into the heart of the system. Not only that, really I don't think we could get away with being grandfathered like we were anymore anyhow. But even more importantly, if we're going to start dealing with commercial vendors' data, they consider it proprietary. They're not going to want to play with us if we can't protect their data.

The new system not only has new security to protect us from the outside, it has security on the inside to make sure that we are fully capable of handling a variety of different data types, and even though we have multiple concurrent things going on in the building they don't get shared with people who are not supposed to see them. There are a variety of new features in the new system. Also some of them that are more subtle. In the second generation system I mentioned the process we call recon, reconfiguration. Every time we get a new vehicle, every time we fly another mission, we wind up having to reconfigure for it. We don't like to do anything twice.

It's a significant process even with a soft-configured system like we have now to regenerate all of the file products necessary to understand how to interpret the downlinks and calibrate things right. Unfortunately in the second generation system part of that process was all the way over on the client side of the network, simply because the network bandwidth was not enough. That's the compromise I mentioned earlier. I could not fully process the data and make the client recon-insensitive because the end of it was we fixed that.

The new scheme that we're using to traffic messaging inside of MCC-21 totally removes the client side of those messages from the recon process. They don't require any recon products. All fully self-interpreted messages. There's also another element in this that has yet to happen, although I think we're on the brink. I have this vision that I've had for a while that's difficult to make happen, but to use the cliche OneNASA, a much more integrated ability across the Agency of the Centers to interact with each other.

We designed the mechanisms that we're using internally where they work equally well externally. The one I was just talking about, our internal messaging scheme, is very well suited to share data with another Center. It's very efficient. Like I said the recipient Center does not have to know anything about recon for the vehicle. It also has security mechanisms in it.

Also even the second generation system is technically what's called a service-oriented architecture. To explain that a little bit, obviously it's network-oriented, so things actually happen based on IP [Internet Protocol] addresses for source and client, but you don't have to know the numerical IP address. It's a little more like the way you surf the Internet where I know this URL [Uniform Resource Locator] for a Web site and I stick it in there. Actually that gets resolved in a domain name server somewhere where that name is matched up with an IP address which connects you to port 80 on that server out there someplace.

You don't know that, but that's what's really going on. We do that inside even our second generation system, but on a much larger scale than what a DNS does, a Domain Name Server does, because we actually rendezvous not only just with IP addresses but port numbers on the machine. So one physical machine under one IP address could be vending a lot of different services.

Our whole internal architecture on the second generation system was really service-oriented. It did not connect with things by knowing IP addresses. It connected with things by knowing the name of a service or the name of a parameter, and it got automatically connected.

Those mechanisms were largely built in the second generation system to work inside a system. They were not well suited for a wide area network. They were okay for a local area network. When we generated those functions in MCC-21 they were built where they can work quite well on a wide area network. It sets the stage for interactions between us and Marshall or us and KSC [Kennedy Space Center, Florida] to where we can take some of the techniques that we have proven over the now decades here to work well for us and push them out in the communications between Centers.

We're expecting some of that to occur going into EM-1 [Exploration Mission-1]. Some of the work we're doing with Marshall folks and figuring out how we're going to interact with them. They're building the rocket, the SLS [Space Launch System]. It looks like they're going to accept our TIM format, Tagged Information Message is the name we gave to this new way we traffic data, as an input. We've agreed to provide them software to read it. There's an opportunity to save some money there but also to start a process of standardizing the way the centers will interact with each other.

WRIGHT: Yay!

MCGILL: Yes. Maybe my dream of pulling the Agency together, which I think the Agency would be incredibly powerful if the Centers could work together more. I can't solve the politics

but I can certainly work the systems engineering side of the thing to try to put mechanisms in place that allow that to happen if the desire is there.

WRIGHT: You could build the road for them to go down.

MCGILL: That's right. We're doing that. There's a lot of it built in. Some of it is a natural artifact of opening the thing up to the outside. But some of it is making very clever choices about how things are done so that we already have the solutions in hand for the problems that have not been placed on the table quite yet but we think are there. You can see there's a list of categories of things that need to be thought about when you're fixing to rearchitect one of these systems. Which leads me to one of the questions I saw that you had on your list that I think is an interesting question. What are the big challenges?

To me the biggest challenge, certainly if you're working at the system architecture level, is communications. Let me explain what I mean by that. There's probably not a large percentage of the community that's running around figuring out where you want to be 20 years from now. It's just the way life is. The majority of the people don't think past Thursday next week. Building large systems is very much a team sport. It takes a lot of people to do it that range all the way from the architects at the top to the software developers and procurement organizations. There's a large number of people involved, and there's decisions being made all up and down this hierarchy.

As I mentioned, there's a fairly well documented process in systems engineering, 7123 for NASA, but there are many many descriptions of the engineering process out there. It really talks about how you have the set of requirements somebody wrote, you take those requirements,

and a smart engineer does some designs and produces some sort of a design spec, and then that design spec becomes the requirements to the next guy. All of a sudden this thing decomposes all the way down to things that can actually be implemented.

That's all true. But what's not in there are the things that defy being represented in those requirements. That's the reason I mentioned that several times. There is a set of drivers that need to go into the engineering process at every one of those levels that is cognizant of where we're trying to go with this. These are not things that we can consider requirements because you can't test to them. But they're things that cause me—of the five different ways I could satisfy these requirements—cause me to pick a particular one over the other four. Those decisions are made across the entire team.

As you decompose one of these things it grows geometrically, and you can quickly wind up with 1,000 of these things going on, and one person can't possibly be cognizant of all the decisions being made. So, the challenge is how you take those difficult to represent things that can't really be requirements and communicate them across the team so this whole array of decisions that are being made on a daily basis really favor where you want to go with it.

I don't have a really great answer for that. I would love to challenge someone who might read this someday to think about a great way to blend that into documented engineering methodology. It may have to do with a set of categorical questions that might get written down and evolved with time, where as design decisions are made you go back and challenge them against these questions.

Well, how will your design react if suddenly we have a mission that is going to involve three countries to go fly it? How are you going to tolerate that? How is your system going to

respond to all of a sudden wide area networking is twice as fast and half as much money as it is today? Can you take advantage of that?

Those kinds of questions possibly could be documented and used as a test and some guidance that when you get down into the ranks of the people building it to help them steer their decision process a little bit to favor the things that you think are important out there. The actual set of questions varies depending on the perception, but it may be a better way to try to communicate that across the entire team so that all those decisions they're making that definitely affect the behavior of this thing, the architecture, are made smartly.

That's a challenge I guess to anybody that's reading this. Think about how you think you ought to do that. If you believe what I said about architecture as a layer above a system design, if you want to see it that way, an architecture is all wrapped up in behavior and how well this thing may or may not behave itself over a period of time in the future, then the real question is how do you get the whole team to understand your vision of what that looks like, and the decisions that they're going to have to make to make them consistent with helping to make that a reality.

To me that's the biggest challenge. Now of course there's plenty of challenges in this business all over the place. If you're designing hardware how you get it to go fast enough. But I think the one that really is the opportunity to break some new ground, do some good things for the Agency and maybe aerospace in general, is to figure out how to better address those somewhat elusive characteristics that a good system versus a bad system has to exhibit. To me a good system is one that lasts a long time. Replaced all the pieces in it, but the architecture is still there.

I'm sure there will be a fourth generation out there at some point. Hopefully it'll be 20 years from now or so. Maybe less. Maybe by that time someone will have a very clever way to have written an annex to 7123 to put all these factors into the consideration in a more formal and organized way than simply using the missionary process that we do today where you're running around telling people and hoping that they understood what you said and go back to their office and think that way.

WRIGHT: Let me ask you too, when you're talking about challenges, through the course of our conversation you mentioned about for MOD it was pretty much all their domain.

MCGILL: Closed shop.

WRIGHT: Then was there a lot of cultural challenges when you started broadening that? Because all of a sudden now it wasn't a controlled factor as much as it was a communicated factor.

MCGILL: There is some.

WRIGHT: Or is some of that going away now?

MCGILL: It was surprisingly little actually. I think because it unfolded incrementally. There was some shock like that. Certainly when we started Station and it became an International Space Station. I mentioned that the MOD position early on was no element ops. If they want to play they come here. That changed gradually where we wound up with an ESA Control Center,

a JAXA Control Center, and the Russians sitting in Russia. They won't come here. We're interacting with those people. That happened over time. I think it was not as much of a culture shock because it happened over time. The more recent element, when we canceled the Constellation Program and the administration's decision was to put more emphasis on commercial vendors, which I personally think is a great idea, the reality I think of you can't own it all yourself anymore started settling in.

You really had two choices. Either you got with the new program or you went and found yourself something else to do. I think it added another layer on modifying the culture to recognize that our future was dependent on our willingness to be a bit part player rather than to own the whole show ourselves.

WRIGHT: Interesting way of looking at that.

MCGILL: The other side of it, I don't think MOD was ever in a land grab expedition. It was never that way. This is probably self-centered because I've always lived in MOD. But we stepped up to do whatever it took to go fly the mission. Many times we thought it was somebody else's charter but they didn't seem to want to do it. It wasn't that we were out trying to assimilate the world or take things over. It simply evolved that way that we wound up doing it end to end to get it done.

We sit here today for example. We, FOD [Flight Operations Directorate], have equipment positioned at KSC and positioned at White Sands. We don't really want to manage equipment at those places. We think they should have put it out there for us but they didn't do it. I don't think we got there because we were being self-centered. I think it was a natural evolution

process that it's the nature of the culture of the organization is we're going to get it done. You can do your job, or we'll do your job for you if you don't want to do it. But we're going to get it done. That's always been the way MOD saw things. I think our success over the years demonstrates that we really do mean it.

WRIGHT: You took that full responsibility and continued the accountability.

MCGILL: It wasn't ever that there was a decision made that we were unwilling to share. It was more one focused on the mission objective and the importance of pulling it off in a very effective way that caused us to do the things that we had done. I think that it was less of a culture shock in the strict sense of it but a realization okay, maybe the world out there really does want to play now. We certainly see viable commercial vendors coming along. That hasn't been true for very long.

It's really more of an adaptation I think to the situation as we see it. I don't think there was an awful lot of culture shock. There's still a few people that are territorial. But that's always going to be the case. People are very proud of what their organization knows how to do and don't believe anybody else can possibly know how to do it.

WRIGHT: When you're responsible for the safety and maintenance of not only the Station but the people that are on it, it makes you feel a little—

MCGILL: We take that job very seriously. Yes. One of the things I've discovered over the years. I'll say a couple things about it. You walk into things thinking, "Oh, everybody does

business the same way we do." We found out that's not really true. Give you some examples. Even in the US space program, we had been responsible for these vehicles from the time they left the pad until they landed, splashed down, from end to end. Things like understanding how to do an ascent design, which is a very complex problem that the flight dynamics people take on, to tune all those factors that have to be tuned to make sure you insert this thing exactly in the orbital window you want to put it in, most other parts of NASA have no idea how to do.

The reason is the science community out at JPL [Jet Propulsion Laboratory, Pasadena, California] or Goddard [Space Flight Center, Greenbelt, Maryland], they go buy launch services from somebody. That launch provider figures out how to do that and delivers their payload to them already on orbit. You don't find knowledge in the Agency on that side of the thing except here, because that was not part of their charter to do that.

Interestingly enough, the way that came down, which people think I'm crazy when I say this, but it's actually true. Remember when we started this whole thing out it was Mercury. We didn't fly them out of here. Mercury was flown out of KSC. The start of Gemini was flown out of KSC. The decision to move mission ops here put us into Gemini before we were going to start flying them out of here. We'd already built the Control Center. It was online but wasn't primed for the flight for Gemini III I believe.

It just so happens at the time that we lifted off Gemini III and of course television was pretty primitive back in those days, the cameras were very insensitive and you had to have huge lighting to see anything, when they fired all the theatrical lights up down there, they brought the power down on the LPS, the Launch Processing System, in KSC. Right at first motion, we were already online. KSC went down. We said, "That's okay, we got it." That set the stage for why we took control of the vehicle at first motion. There was no other science in it, nothing else

manned or unmanned launched anywhere in the world hands it off to the operations center at first motion. Everyplace else hands it off on orbit.

But people should also remember back in those days from one mission to another things were very dynamic. We were told to get to the Moon and get there quick, and each mission had specific new things that we weren't real sure we knew how to deal with. So, there was an attitude well, if it works, we'll just keep doing that. When we took it over from KSC to here on first motion, that worked just fine. That just set the stage, we've been doing it that way ever since.

But an artifact of that is we find that the knowledge that exists within MOD is unique compared to other space missions that are flown by NASA. Certainly we find more subtle indications of that. We have crews. The other missions flown by NASA don't. Obviously the Russians do, Chinese do, some of them. But most of those missions don't try to sustain what I'll call a synchronous forward link, a hard forward link between the ground and the spacecraft, all the time. They don't have anybody to talk to and we do.

We find very subtle differences in how we want to do business operating through TDRSS [Tracking and Data Relay Satellite System] for example or the Deep Space Network or a ground station compared to what anybody else wants to do, because we have people on board to talk to, and they want to be able to talk to us. There are some very subtle things in there that are unique simply because it's a manned program. I think there's reason for MOD as an organization to be extremely proud of the breadth of missions that they know how to deal with. You won't find that anywhere else.

It just so happens that normally the way the functions are allocated they don't all exist in one place like they do here. Self-serving but it's true. We are I think a great ally for a

commercial vendor that's trying to get into the business or another country that wants to participate in some great adventure, because we really know a lot of things, and we've got a lot of years of learning where the landmines are and where not to step. There's a great deal of experienced knowledge that exists within the organization that is relevant to pulling these things off.

WRIGHT: Although spacecraft may change and missions may change, that basic knowledge of what—

MCGILL: The laws of physics do not change. They are the same here as they are in Goddard, as they are in Moscow. There's certain common factors underneath all this. The nature of the business makes it very dangerous. You have to deal with very large amounts of power that you try to keep under control, but can get away from you very easily. If you're going to take something fairly massive and accelerate it up to 17,500 miles an hour, it's going to take a lot of power.

Until somebody can come up with antigravity paint or something like that, this is just a very risky business. You're dealing with huge amounts of power that you've got to control all the way down to the fifteenth decimal place. It's a funny combination of precision and power that has to always be balanced out. It will get away from you very very quickly.

WRIGHT: Thinking about some of the words that you shared of making sure that the communication of the systems on the ground can tell you what's going on so far away so that you can prevent and/or anticipate any issues.

MCGILL: MOD in our classic role I've always thought of as being schizophrenic. Our primary job is to go fly a mission. So the programs have defined a set of detailed objectives that they want for this mission and the job is to go fly a mission and produce those results. However, experience will tell you that the spacecraft don't always behave right. Things break. You wind up in a troubleshooting mode. You wind up doing onboard system management and taking corrective actions mainly to try to protect your ability to meet all those mission objectives.

Because things don't go to plan—they never do, I don't think they ever have—you have to be very very fleet of foot, very agile in your ability to replan to try to accomplish everything the program asks you to accomplish, even though it didn't happen the way it was supposed to happen. It's always been amazing to me. We spend an awful lot of effort planning these things out. We spend a lot of time running simulations, practicing not only the nominal but even more so the off-nominal. So often the off-nominal case that occurs is not one we practiced. There's so many more things that can go wrong than go right that I guess just statistically that's the way it is.

WRIGHT: Is there some times that you remember back through all these years that maybe not just through the transition but the systems that you thought were going to work so well glitched for some reason that you had no idea? You were talking about the incremental changes and how you saved the ascent and descent for the Shuttle as the end, knowing everything was as good as it could go before you went to those critical steps. But were there troubleshooting along the way during those? Or did much of what you planned fall in the place the way that you had perceived it and planned it?

MCGILL: No system is ever perfect. Certainly if it's new it's very imperfect. We will go into operations with MCC-21 with a lot of flaws, many of them known and some of them unknown. It's always a calculated risk. We've never in the history of the Control Center had a system on the floor that had no known anomalies documented. Our DR [Discrepancy Reporting] database, DRTS [Discrepancy Reporting and Tracking System], or now we use a different system, but whatever it is has always got thousands of things in it, most of which are known to us, we have operational work-arounds. You're used to when you fire up your PC, [Microsoft] Word is not flawless, PowerPoint is not flawless. We learn how to work around them and get the job done anyway.

Obviously the newer the system is, the more likely there are unknowns. Our best protection, the way we try to get enough of a warm fuzzy to sign a CoFR [Certification of Flight Readiness] that we're willing to stand up to the program and say we're ready to go fly, is we try to log as many hours as we can testing and running simulations so the probability of having encountered those anomalies is as high as we can make it. Doesn't keep us from discovering something for the first time when we're on orbit, but the only way you can mitigate that I think is we try to to the extent possible have more than one way to do things.

You try to make sure that you put in place operational work-arounds that are inherent in the way the thing works. If it can't get it done the way I wanted it to, it may not be real convenient but there's another way to do it. To the extent that you can anticipate those kinds of things and have the money to build alternate paths in the system, that's about the best you can do. But I think we've had a history of deploying fairly well by systems.

It was established very early in the program that we were going to operate to a level of "two nines five" reliability. That's the probability of success or reliability number that says, "Well, what's the probability of you being able to do your critical functions?" The number we established decades ago was 99.5. So there's a half percent chance we'll fail by the way, which we haven't yet, but we're due.

It turns out that if you do that then soon as you say two nines five reliability, that means that for every critical function you do you have to be one fault-tolerant. You can't get there any other way. For all of those things like being able to command the spacecraft, being able to process trajectory and know where it is, being able to view the telemetry, being able to communicate with the crew, all those critical functions, whenever we're in critical or complex phase, depending on whether you want to use a Shuttle or Station term, we guarantee that we're two-stringed. We have two separate independent processing strings up. If one of them fails, we switch to the other one. Of course that doesn't protect you any for common cause problems where both of them have the same flaw in them. But typically even if it is a design flaw, coding flaw in the software, those things that are just flat logic mistakes get discovered very early in the testing process.

The ones that'll tend to propagate long enough down the timeline to get you past all the simulations and into operations are things that require a very particular time-ordered sequence of events before they can trigger. It's so unlikely, that's why you haven't seen it before. Generally if you switch over and you do it again, the time order is not the same, and the problem probably won't surface, even if it really is an implementation problem, a design problem, because you reset the roll of the dice.

It's amazing how often we have seen cases like that where we're sure after the fact, go back and look at it, we actually have got a code problem in there we got to go find. Sometimes they're very difficult to find because trying to reconstruct them, get them to happen again, when they're improbable and very conditional. We do a lot of logging in the system to give us as much data to go back and look to see exactly what was going on when this happened, to try to reconstruct it.

Generally another interesting design I think consideration today is the systems don't have to be perfect. They have to get the job done. I'm a firm believer in what I call fault-tolerant software. You can implement it in a particular way where yes, there's a bug in the code, but it'll still get the job done. Maybe it forces a restart, the computer comes back up again and gets the job done, or maybe you switch over to an alternate path and it gets the job done. At the bottom line, you get the job done, and that's good enough. That may mean you still want to go look and see if you can figure out why the heck it did that, or maybe not. Maybe if it does it so rarely you just don't want to spend the money even trying to fix it. Nobody asked us to build a perfect system. They asked us to build a tool to go fly missions successfully.

WRIGHT: On a personal/professional reflection, you've spent a good part of your life in creating systems that are beginning as a vision but yet turn into something very practical and reliable. Any point during that time period you thought about going to do something else? I know you started our conversation, before we started recording, that you had retired. But what happened there?

MCGILL: Yes. I was very happily retired. I retired in 2003. I'd done 35 years and three days. It turns out you only retire on a Friday. Did you know that?

WRIGHT: No, I didn't.

MCGILL: I didn't either. But apparently it's true. There were a lot of things at the time. There wasn't much dynamics going here. The second generation system was supporting perfectly. No one was suggesting we needed to do anything about it. I had been off actually doing commercial projects. One of the other guys who was at Lockheed and I teamed up and we started a little company within the company. We were doing things for a variety of commercial companies. I'll keep them nameless. Doing pretty well at it, but the corporation didn't much care for us out doing commercial work. They thought themselves in the aerospace business, and I think they were a little afraid of us.

We started backing out of that somewhat because it was not smiled upon by the corporation. It was a lull. There was going to be a change in the retirement policy that the company had about that time. It seemed like the right time. I went ahead and pulled the trigger and retired. I did occasionally go do a little consulting work either back for Lockheed or other places like DARPA [Defense Advanced Research Projects Agency], but really wasn't looking for anything like a full-time job at all.

When we got into 2006 the Constellation was flaring up big time. Lockheed Martin out here was obviously a major bidder on Orion. They were using a number of the local people to try to work on that proposal effort. Because of the way NASA chose to run that competition that required them to firewall them away. There was a fair amount of work that needed to be covered

over on the NASA side that there really wasn't any resources to cover. That resulted in me getting a phone call saying was I at all interested in doing anything. I got to thinking about it. I said, "Well, engineers always demand symmetry." That's a flaw in the engineering mentality. Going beyond low Earth orbit when I started matched up with going beyond low Earth orbit at the end appealed to me. I think once you get the stuff in your blood you can't get rid of it anyway. So I said, "Well, sure."

This was in February 2006 and I agreed to work the balance of that fiscal year over at NASA to help them work the Constellation issues so the rest of the Lockheed guys could go off and design Orions. Of course that was 2006 and I'm still here. They haven't chased me off yet. But the fascination is still there. The opportunities are still there for us to go do great things.

WRIGHT: The technology keeps changing for you to adapt to.

MCGILL: It does. You can't get bored. You're just not watching if you get bored. The tools that we have to work with today are constantly in a state of flux. Many of them very relevant to what we're trying to do. Also it's been beneficial that the world at large is much more space-oriented than certainly they were in the '60s. You have things like the CCSDS [Consultative Committee for Space Data Systems]. It's an international organization that develops standards for major functions that people need in space. Their objective is if you create a standard that's internationally accepted then commercial companies will go design products to those standards because they have a marketplace. That's worked, and that's provided some tremendous capabilities for us that are not good for anybody that isn't doing space but they're commercial products.

What's available as a tool set to go build these things with is changing a lot. Just the basic technologies, how fast networks can run, compute power that we have in these workstations is phenomenal. There's such a change in what we can do, the sophistication of the spacecraft. Orion is almost capable of taking care of itself. It has a very powerful set of fault detection and isolation capability on it. I can't imagine how someone could get bored. They're just not paying attention. I'll stay on as long as they'll let me probably.

WRIGHT: I think one of the questions that I think Blair had sent over was if you were leaving or you were sharing those things for new people coming up, for the next generation coming up. What are some of those aspects or those—I don't want to use the word lessons learned?

MCGILL: He's keying on a comment I made to him when I was standing outside, so you know what triggered this in the first place. Another factor in 2006 when I got the call is I felt like I was fortunate that the taxpayers invested a huge amount of money in my education. Obviously not everything worked exactly great the first time. You learn from that, but it's very expensive. I looked at it, and I said, "I think it's payback time. The country invested a lot in me learning what I learned over the years. I think they got something for their money but also it's time to give back." Toward the end of last fiscal year it looked like because of the budget pressures for example that probably I was not going to be able to continue to do this. Lockheed was really getting squeezed.

To that end one of the areas that is very important to us and not well covered with the resource base is the communication subsystems, which I happen to know a bit about. I went off and decided I'd teach communication school, because I wanted to make sure if I left I left as

much behind as I could. I really feel that way. I'm obviously not going to be able to do this forever, and the next generation needs to pick it up and run with the ball.

One of the guys in the session who was an absolute master with a video recorder and an editor, he videoed three of the four sessions. They're huge files, about two hours each. It teaches you how communication systems work. We stuck those out there on servers and a lot of people have watched them. I hope they're getting something out of them. But one of the things that triggered Blair is I feel very strongly that what I know was paid for by American taxpayers, and as much of it as possible needs to get left behind for the next group to take and run with. I really feel that way. This I see as an opportunity to do some of that. There are other topics probably that are like that. But the important thing is that the space program continue to flourish.

WRIGHT: When you do finally leave, what do you feel like is going to be, if you had to pick one or two, the best contributions that you felt like you've been able to make?

MCGILL: I certainly do think that the evolution of the architecture of the Control Center is one. I'm not going to take credit for the MCC-21 but I've had my finger in the pie on that one big time. That's the next generation that owns that one. They need to get credit for that one.

I think that has benefited certainly the Agency at large and the manned program in particular. I feel like it's been very worthwhile for both sides. Certainly it's been very good for me. I've been able to practice the thing I'm passionate about. Life gets no better than that.

WRIGHT: Rewarding.

MCGILL: If you get to do what you want to do. I think probably the next wave above that is a topic that I mentioned several times here. It's not finished. There's lots to be learned and figured out about how to go do these big systems. There's opportunities for people to think about it, come up with clever ideas and document them. We're nowhere near finished in any one of these threads, including the methodologies.

I like to leave that challenge behind for some of these bright people. One of the huge advantages of working out here is this is the smartest bunch of people you're going to find anywhere on the face of the Earth. I know that to be true because I've been a lot of places. You're not going to find a talent base anywhere close to this anywhere. They're really really sharp people out here that are capable of great innovation.

The other thing I'd like to leave behind is the challenge to pick that ball up and run with it. There's plenty of field left ahead of you.

WRIGHT: I know. I think sometimes the generation feels like there's nothing new to explore or to do.

MCGILL: It is not true. It is definitely not true. In fact if anything the rate at which things are evolving is accelerating. The opportunities are growing. It's not simply working with new products. It's even challenging old techniques.

I had one opportunity in my career that occurred in '91 when it actually happened I guess, rolling into 1992. The situation we were in was early Space Station. Space Station needed a Control Center. We were using a very classical engineering methodology they call waterfall to go create one. The program was quite unstable. I think twice we came within one

vote in Congress of canceling the whole damn program. About the time we'd get through the front end and get a system design spec put together, the whole thing would flip over and you'd go back and start over again. We were executing completely in compliance with the contract, doing what I thought was pretty good work per the contract, but producing nothing of value that was going to go on the floor and go fly missions, because you couldn't get that far.

I think the reality of that was we were spending a lot of money. The cost to complete was basically not changing from one year to the next, and the schedule was just incrementally sliding out. I had a history of running around complaining about bad processes. I guess that was the reason why they said, "Okay, go fix it."

They gave me the opportunity in '91 to go overhaul what was at that time the MSC contract. They didn't put any bounds on me about changing the statement of work and the DRL [Data Requirements List] and anything else I thought needed to be worked on. I did that and changed a lot of things. In fact some people said I changed everything but the carpet on the floor, which might have been pretty close to the truth. But went to what today we would call an agile process. If I'd thought of that term back then I'd have used it. It's a perfect term. I called in incremental because I wasn't smart enough to think of agile.

What it really said is hey, the world is too dynamic. The reason we can't deliver on this is things don't hold still long enough to use the long duration waterfall process. We got to build this thing incrementally. We got to eat this big cookie one little bite at a time, even though we've never seen the whole cookie at once. It's all hidden behind a curtain somewhere. I reshuffled everything around so we could have a systems engineering level that was very heads up, keeping track of all the change traffic, and an implementing organization that were very heads down, that were taking each one of these delivery increments defined and going off and

doing that without paying any attention to what's happening to the budgets or the schedules or the politics or anything else going on.

So we shifted to that. It was interesting, because when we made the change we gave the program back $70 million, I think was the number, and we trimmed two years off the schedule. But in reality that's an inaccurate statement, because like I said, the cost to complete was staying the same, and the schedules were slipping. But the one that was baseline—and we delivered by the way. We met those budgetary commitments and schedule commitments.

The reason I bring that up is it is useful and sometimes very appropriate to be willing to challenge the well rooted processes that are in place. If you can build a compelling case and it makes good sense for why you really ought to do it a different way, then people will listen to you. Obviously NASA contracts accepted all that before we were allowed to go do it. There's plenty of ways to cause progress to occur. Some of them are deep-rooted technically. Some are more just how it is you go about doing it. But the important thing I think is always not just to say, "Well, it's been done this way for 20 years, that's obviously got to be the right way to do it." The right thing to do is look at the situation around you right then because that's what's changing constantly.

The operational requirements are changing. The technologies are changing. The budget situations are changing. The national priorities are changing. You're really pressed to find something that's not changing. If you're not willing to be very agile, very fleet of foot, you're probably not going to succeed. It's all about being able to adapt. That means adapt the processes, adapt the systems, adapt all the things that we're about to succeed.

I got asked soon after we built the thing over there, after I redid the whole contract, the Air Force was on some kind of a summit thing in Colorado Springs. They were trying to figure

out what to do with their systems in Cheyenne Mountain was what they were about. I guess somebody had mentioned what I'd been doing here in Houston, so they asked me to come up there and tell them about how it was we built the new Control Center here. I had a chart [in my presentation that showed the context the system engineering process must operate in]. Basically it was showing that the operations guys, are in a state of flux too. One day to the next they're trying to figure out what the heck they're going to do.

Even made the comment that I could remember when I would have been the first in the line to go say, "Well, if those ops guys can write a decent ops concept I can sure build a system that would meet up with it." Now the recognition that they would love to be able to do that but their world won't hold still either. This whole chart was oriented around if you're going to do good engineering these days you're going to have to be constantly aware of all that change traffic out there, and modifying your plan to match up with it.

It was interesting to look around the room, because I didn't know anybody in the room, I don't think. You could easily see where the ops guys were sitting in the back, they're all standing up and cheering. You could see the systems integrator bunch that the Air Force had hired over there because they're saying, "Well, doesn't that cause requirements creep?" I remember answering that question saying, "Well, I guess. But what I deliver the ops community likes. So I don't guess it matters, does it? It satisfies their need. I don't know whether the requirements crept or not." Because we started doing it before they were all written down. The requirements were evolving too. I don't know what requirements creep means anymore.

There are a lot of comfortable things that an engineering community can hide behind like a well formed set of Level As. It's not in my requirements. I'm not going to do that. Those kinds of things are inappropriate today. The world is not that stable.

WRIGHT: I'm going to ask you one more question and then we'll let you go. Don't want to take all your day. You've seen a lot of human spaceflight. Is there a time during all these years that you've worked around this community that you feel that's the one you were the gladdest you were around? Special mission or that transition or the new Center opening? Whatever. Is there something that just really sticks out in your mind of all the things that you put up?

MCGILL: There are probably a lot of those. But interestingly enough, today is probably the one. The reason is I had the advantage of working with some incredibly bright people. I see them taking the reins. I see them doing things that I can't do, that I don't know how to do. They're more contemporary in their knowledge of the technologies, and very talented people. To me there's nothing more rewarding than that, seeing what a bright future our program has with those kind of people who have a lot of years ahead of them to be able to contribute like that.

Today is probably as good as it could possibly be. Of course you look back and you see some of the things that have happened over time. Certainly the day that we really started transitioning the second generation architecture in place was quite fulfilling to me since I was the thing's daddy. You always like to see your baby succeed. Certainly if you look at some of those events back in Apollo, every one of those missions was very risky, very challenging. The organization and the tools that were available for them to work with helped us pull off some rather remarkable things I think.

You have trouble finding a year where you can't find something to be very proud of. I don't think any exceeds today. Just watching what the team is capable of doing is incredibly rewarding.

WRIGHT: What a great time for you to learn even more.

MCGILL: It is. One of the things that—I guess you can teach an old dog new tricks. When I came out of retirement, there's been a lot of things learned since then that I've learned. I like to learn. That's another major pleasure that I get out of this thing. There's so much in this business, so much complexity in it. Every one of those rocks has got a gem underneath it. You turn it over and find, "Wow, I think I'm going to learn more about that." There's some really really interesting things out there to dig into.

WRIGHT: Maybe when MCC-21 is up and running you can come back and visit with us and tell us about the end of that saga.

MCGILL: I will. We should be totally flying by the end of the year with it. So it's not far down.

WRIGHT: I look forward to talking with you again then.

MCGILL: So far we've got enough testing with the simulations. The basic behavior that we're looking for I think is actually there. It's not finished. These things never are. We've been working on the Control Center for 50 years, never have finished it. It's not so much a point, it's a vector. You're really pointing it in a direction, putting it on a vector. Now you continue to build on that off in that direction. There are a lot of things in there. There's some features that I would have liked to have seen get in there probably day one, but that's okay.

One of the fascinating things about building it in the first place is with all the change traffic the front office recognized that maybe it was time to do something. They told a few of us that, "Okay, if you had a clean sheet of paper how would you do the Control Center?" Clean sheet exercise.

We all head out for a week. I say all. It was four or five of us head out for a week in an office up there. I remember telling them. I said, "Well, we'll ask for everything. They'll give us about 50 percent. We got to be willing to accept about half of what we're asking for." That's the way it usually is. So, we put everything in it that we wanted, and they gave it to us all. They told us, "Just go for it." The front office very much supported our desire to make wholesale changes to the way the thing was put together and move it in a significantly different direction than where it was set to go.

You really can't ask for better than that. Most of it got in there, even though there were certainly some things that absolutely had to happen, other things could be put off and it wouldn't hurt. But everything that had to happen is in there, and a lot of the things that didn't have to happen right now are also in there. It's difficult to figure out exactly how you measure the success or failure of one of these things. But I'd declare it a success.

WRIGHT: The future will be fun seeing how it all comes about.

MCGILL: Yes. I'm excited about like I said the work we're doing so far for EM-1 and 2 because it is more intercenter. Seeing the great spirit of cooperation and working with the other Centers, I think moving down that path of OneNASA, if you want to use a cliché, can actually happen,

where we can really pool the different resources at the Center into one enterprise objective and make it all fit together rather seamlessly.

We can get that started. That's probably not going to finish while I'm still around. But that's another nice challenge for the next wave to go take on to pull all the Centers together.

WRIGHT: Exciting times in the future.

MCGILL: It is. Absolutely. Absolutely.

WRIGHT: Thank you for giving up your whole morning. I appreciate it.

MCGILL: You're very welcome. Thank you for the opportunity.

WRIGHT: I've enjoyed it. Thank you.

[End of interview]