This interview was carried out by the Stanford Oral History Program, sponsored by the Stanford Historical Society and the Stanford University Archives. No rearrangements were made in the transcript of the recording; the transcript was lightly edited by program staff, the interviewer and the interviewees, but few substantive changes were made.

The transcript and recording are part of a collection of oral history interviews documenting the history of the University and the experiences, accomplishments, problems and viewpoints of members of the Stanford community. The Stanford Oral History Program is under the direction of Susan W. Schofield, Academic Secretary, Emerita and the Oral History Committee of the Stanford Historical Society.

All uses of the transcript are covered by a legal agreement between the interviewees and the Stanford Oral History Program. The transcript is thereby made available for scholarly purposes in the Stanford University Archives. The copyright to the transcript including the right to publish is reserved to Stanford University. No part of the transcript may be quoted for publication without the written permission of the Stanford University Archivist or his/her representative.

Requests for permission to quote for publication should be addressed to the University Archivist at the Stanford University Archives, Green Library, Stanford University, Stanford, California, 94305, and should include specific passages to be quoted, anticipated use of the passages and identification of the user.
STANFORD HISTORICAL SOCIETY ORAL HISTORY PROGRAM

Purpose Statement

The Stanford Historical Society's Oral History Program explores the institutional history of the University, with an emphasis on the transformative post-WWII period, through interviews with leading faculty, staff, alumni, trustees, and others. The project furthers the Society's mission "to foster and support the documentation, study, publication, and preservation of the history of the Leland Stanford Junior University."

The interview recordings and transcripts provide valuable additions to the existing collection of written and photographic materials in the Stanford University Archives. In addition to scholarly use, information from these interviews may be shared more broadly through print articles, campus lectures, or electronic media.

Like any primary source material, oral history is not intended to present the final, verified, or complete narrative of events. It is a unique, reflective, spoken account, offered by the interviewee in response to questioning, and as such it may be deeply personal. By capturing the flavor of incidents, events, and personalities, the oral history approach provides details and viewpoints that are not often found in traditional records.
J. Kiefer: This is Joyce Kiefer interviewing Bill Kiefer and Dick Guertin on June 4, 2010 about SPIRES, the Stanford database management system designed by the University's information technology group. SPIRES stands for Stanford Public Information Retrieval System. I want to mention that a video was made in 1990 about the development of SPIRES from 1970 up to that time. Nine people were interviewed. This interview that we’re doing today will cover the story of SPIRES from 1990 to this year, 2010. So before we get started, will each of you state your name and your affiliation with Stanford?

Kiefer: [00:00:41] I am Bill Kiefer. And I’m retired from Stanford. I retired in 2004, so I’ve been out for a while. I worked for the ITSS and its preceding groups, whatever they were called, for 34 years, starting in May of 1970.

J. Kiefer: Dick?

Guertin: [00:01:16] I was hired one week after Bill. Basically, the first week of June, which is—I think my anniversary is this coming Sunday, 40 years. And so I am still actually working for Stanford. I can’t really say I’m working at Stanford because Stanford has now what’s known as a Work Anywhere Program, and my office is basically in my home. I am not retired, so I’m still actually employed but only as a half-time employee.

Kiefer: [00:01:46] You know, I might make one comment about Dick and my relationship, that we actually worked together for several years before we came to Stanford. And that was at Control Data first, and helping build an
operating system for Control Data on the 3800. And then at a company called InterAccess, which we sort of started. It was started after our computer system was relegated to the scrap heap at Control Data. Our group bought the operating system and started a company based on that. So we’ve worked together and we know each other’s capabilities, and we’ve known it for quite a while.

**J. Kiefer:** I want to clarify early on here is, I know that there was SPIRES I. That was the very beginning of SPIRES, and it was connected with SLAC. And then SPIRES II. And that’s what developed through the years. So I would like you guys to clarify that better than I’ve just done.

**Kiefer:** [00:03:00] I think the first thing to talk about is what was SPIRES I. And as I recall, maybe Dick can let me know if I make a mistake. But anyway, I think it was a batch—essentially, a single user batch search and update system. Right? There was no interactive part of it?

**Guertin:** [00:03:22] Right. It was written in what was known as PL/1, which IBM had written as their priority language, primary language for the IBM 360 machines. And everything about the file structure and about the organization of the data was built into the program. So the file essentially only contained the data and nothing else. And the PL/1 program contained all of the definition of where the elements were located and all that kind of stuff. And I can remember going up there when Bill and I started working on this project in—we actually started right away in June of 1970.

And we went up to visit the chief librarian at SLAC. Her name was Louise Addis. And she brought us into a room that had a teletype terminal. And she had brought up the SPIRES I system. And she said, “Now, I want to show you something.” So she put in “Find title Nuclear.” And then she said, “Okay, let’s go back to my office and talk about what I want to have done.” And I said, “I don’t see anything happening.” She said, “No, it’ll take 20 minutes to get an answer. And that’s what I want to talk to you about.” So we went back to the office and she said, “We need to come up
with a way of being able to do this much faster.” So that was our mission, to come up with a new version of SPIRES.

Now, in the process of doing that early on, we got input from Tom Martin, who was a post-graduate student working at the University, and he was basically saying that we needed to put the organization of the file, what the elements were and how they were processed, into the file itself so that you could port the entire database around with all of its definitions built in. And that would allow you to write different kinds of databases, and only one system would be needed as long as you knew how to interpret or execute the stuff that was defined. Is that how you understand it?

Kiefer: [00:05:21] Yes. I did make a mistake then, because I had forgotten. Essentially, SPIRES I was interactive in that they used the teletype for input. But single user—

Guertin: [00:05:33] Single user.

Kiefer: [00:05:34] —one search for one thing. And you wait 20 minutes, and you might get it out or you might not.

Guertin: [00:05:41] But when we went back after that 20 minutes, the answer was 7,000 and something. It was not a large number of records. But the importance of this file, called the Preprint file, was that the cost of doing an experiment on the linear accelerator was very high. And the experimenters did not want to repeat an experiment that already had been done. Now, when someone did an experiment, they wrote it up, and they then put it into a form known as a preprint document which was before it actually gets printed. And that was being put into the computer. So what Louise was saying from the library’s point of view was we want the experimenters to be able to see what experiments have been done before it gets into the press because there may be a six month delay. And during that six months, they may go out and spend tens of thousands of dollars doing an experiment that’s already been done.
J. Kiefer: Okay. So then SPIRES II, which followed, we'll just refer to as SPIRES and not SPIRES II. How did this next iteration of SPIRES move from working very well for SLAC to the broad system of making the University run?

Guertin: [00:07:00] Well, that actually turned out to be an interesting odyssey. The very first file we put up was the preprint file which we gave to SLAC. But then we turned around and started looking at what we could do for the University. And the University, at this point in time, did not have any particular plans about doing anything with this system. But we found outside vendors who were interested. The first was the San Jose Police Department. They wanted to have a fingerprint file to keep track of all the fingerprints on each finger of a hand. And they could digitize this information, and then you could search for a criminal by the fingerprints. The school districts of California were interested in a database for all the various school districts to show how many students there were, and who the principal was, and all kinds of information. So the first few database systems that we built actually had nothing to do with Stanford, and yet, they were being done at Stanford.

Kiefer: [00:08:00] That's true. I guess the one thing that kind of drove the original product, I think it was described on the 1990 video, was the BALLOTS Project. So the library system at Stanford, and certainly that’s all documented in the video tape we have. But that was also a driving force for this system.

Guertin: [00:08:29] And BALLOTS was an acronym for Bibliographic and Large Library Online Timesharing System.

J. Kiefer: Okay. So then when each of you was hired, Bill, what were you hired to do in 1970?

Kiefer: [00:08:48] I interviewed with John Schroeder. And it seemed obvious to me that I would be working on the file system itself-data structures-because that was what I had done for Control Data and InterAccess, the other companies where I had recently worked. I didn’t know that essentially, we would be starting from scratch. We didn’t have a computer at the time to work with.
We were talking about what should we get. How were we going to even get started? We thought if we have our own computer, we would have to somehow have a timesharing system that goes with it, and would also need a text editor. All these were problems that hadn’t even been defined. I mean we had the problems defined, but not the solutions.

But then John Schroeder came in one day and said he’s just been talking to the data systems group at Stanford that had control of the computer in Pine Hall. They were the systems programmers. And Jim Moore I think is the one he talked to. They said, “Well, we have this thing called ORVYL and this thing called WYLBUR.” And ORVYL became the timesharing system we needed and WYLBUR became the text editor we needed. So what we had to do was shrunk quite a bit by those systems already being there for us to use.

Guertin: [00:10:28] Now I was hired for a different reason. My expertise was basically in languages like Fortran and stuff. So when John interviewed me and said he wanted to get this SPIRES project going, he commissioned me to explore various kinds of programming languages that we could use to produce this product. And once the decision was made to use WYLBUR and ORVYL, it just then became a question of what were we going to use for a programming language. I started running tests using every language that I could come up with—assembly language, COBOL, Pascal, PL/1, you name it. And nothing seemed to satisfy our needs.

Some of the languages were very hard to program in because you were working in these cryptic codes that didn’t make much sense to most people. Something like PL/1, you could write a three-line PL/1 program and it would take thousands and thousands of bytes of memory. So it was just a core hog. Back in those days memory was very expensive, so you didn’t want a big program. And then Bill discovered something. He went to the library and found an article in a magazine about a program called PL360. And then, after doing some research, we discovered that it actually was made
at Stanford. It was part of the Algol-W Project. And we discovered it was in a cardboard box of cards under a desk in a building not too far from the data center. Mike Malcolm was the guy who was holding onto it. And we got it from him. He didn’t particularly care to have the language around. And we put it on tape—got it off of cards, got it on tape. Started making modifications to it, and it turned out to be the programming language of choice. It was the thing that gave all capabilities that assembly language had, but had a much higher-level structure to it.

**Kiefer:** [00:12:33] Yes. And that’s where we went—wrote our hundreds of thousands of lines of code with that language. And it’s pretty amazing because every command you gave in that language practically was a single operation in the IBM computer. So it was extremely efficient to use. Anyway, what came after that is kind of interesting. But we’ll get into that later.

**J. Kiefer:** So you two and SPIRES came together as SPIRES began its full development. What would you say were the goals of SPIRES at that time?

**Guertin:** [00:13:20] Well, initially, it was to be able to process this preprint file at SLAC. But then, because of the forethought on the part of Tom Martin, that it should be a general database system, not specific. Like I say, it branched out to things like fingerprint files and all kinds of things. And then the University began to realize it could be used to handle virtually any kind of data. It could be used to handle student records, or financial records, or employment records, or leave of absence records, or housing, or anything you could think of. And its general nature made it possible for us to think about using it for bibliographic purposes, which is how we got joined up with BALLOTS, because that library system needed to be able to do searches against titles and authors and things in a way that had never been done before. And SPIRES was designed to make that all very simple and easy to do. So SPIRES and BALLOTS became a grouped project. And they were writing the interface to full-screen while we were doing sort of the back end
part, which is the actual database engine.

**Kiefer:** [00:14:32] One thing, just when you think about the very beginning and what was really important in our minds, I remember there were several terms that I probably should mention here. Efficiency. When you issue a search, you want to get an answer within a reasonable amount of time. Reliability. That's that the system is going to keep working. And it's—when you store data in a file that it's going to be there for as long as that file exists. The database, that is. Recoverability. When things crash, when some situation in the program—you know, we made mistakes. We had bugs. ORVYL had problems. The hardware could malfunction. When the system crashes you need to recover it, you should get back to where you were, the user would not lose what he did. That was extremely important. And when we worked at InterAccess, that was one of the major parts of what we did there. No matter what happened in any job that was running, or any command that a guy issued from his terminal, that if the system crashed, you could come back up and be able to be exactly where you were when you left off. And that was extremely important, especially with ORVYL being such a—at that time, not a very highly recoverable, reliable system. Let's say that.

**Guertin:** [00:16:08] And that's where Bill came up with a very interesting concept known as the deferred queue. The ability to take a record and when you want—say the record already exists in the database, and you want to modify it. The update doesn't go back and modify the record itself. Instead it produces a new copy and puts it in something called the deferred queue. So the deferred queue gets a completely brand new copy. And the deferred queue is written in such a way that the last thing that's done is to update the directory of the deferred queue to say where the new record is. So if you haven't completed the transaction of writing the record, you don't even know it exists yet. But once it finally exists, you then update the thing that says, okay, here's where it is. And that is now the replacement for the new record until we actually reconcile the two. So the deferred queue was a very interesting way of minimizing the impact on the system.
J. Kiefer: So deferred queue is---

Guertin: It's a list.

J. Kiefer: Yes. Okay.

Kiefer: And at that time, the actual indexing of a database, in other words, if you want to search on an author’s name, at that time, you could not—if you had added it during the day, you couldn’t search for it and find it. It was only during overnight when all the indexing took place. So the computer was running at night generating all these indexes.

Guertin: In a recoverable manner. Making certain that as it rotated, it didn’t unwind what it had until it was completed with the new one in some new place. And then it says, okay, the pointer to where it is has now changed and it’s now in a new place.

Kiefer: Later on, we did have something called immediate indexing which people could—if they wanted to have their indexes available during the day right after they put it in, then we put in this feature called immediate indexing which would do that.

Guertin: And that was usually limited to just a few indexes. But it’s gotten to the point now where some databases do that for everything, for every index they build.

J. Kiefer: So over time, as those things got accomplished, did some of the purposes or the goals of SPIRES change, you know, over—we're talking about 40 years?

Kiefer: Well, first, the thing we had available was just searching and updating. That was the original system. It was an information retrieval system. You put data in, you get it out. But it became a full blown data management system because there were different things written that would allow applications programmers to write their own programs in language we prepared for them.
Guertin: [00:18:56] Initially, we had separate things. Like for doing the indexing at night, we had something called SPIBILD. And for compiling a new file and making the characteristics of the file, we had something called SPICOMP which would compile a new file. But, over time, those things were incorporated into SPIRES itself so that then the command language of SPIRES expanded to have commands to compile a new file, or to do the processing of the indexes, or whatever you wanted to do.

And new things came along, like Formats, that said this is how we’re going to format the data for input or output, and things like procedural languages that said this is how we’re going to be able to actually allow you to do commands without having to type them in, but have a script of commands that would be stored in a SPIRES database and would be executed out of the SPIRES database. So SPIRES began to grow, not in its own physical size, but in its capabilities by having databases that contain all these various pieces. And what’s interesting is that even today SPIRES is only 954 kilobytes. It’s less than one megabyte in size and yet it can do virtually anything you can think of because most of everything that it does is out in files. You want to explain something, there’s an Explain file that contains all the explanations. You want to compile something, there’s a file that contains the record that defines the compilation information. So everything is done outside of SPIRES, its physical—you know, core memory. It’s a very interesting concept.

Kiefer: [00:20:25] In fact, some of our commands, some of the things that a user might think are actually internal in SPIRES are actual programs that we write as systems people that are higher level, made from this command language strings. In that way, it’s kind of an incestuous system. The fact that it was less than a megabyte in size was extremely important because the mainframe we had to run on, even up until the ‘90s, didn’t have that much space for things to work. But yet, we had many—say, over 17 or 18 applications going at one time with only one program servicing. And I remember one time looking down, and I counted, there were 2,700 users, Stanford users logged on to this SPIRES system. And they were getting their responses just like...
that. So that was an amazing thing, especially when you compare it to what we were getting in the ‘70s. Sometimes you’d wait five minutes for a search for the Student system.

Guertin: [00:21:37] And then we’d find ways to improve that speed. So my answer to that is that, over time, over the past 40 years, SPIRES has grown to meet client needs. And would I consider it success? Yes, because every time we encountered some problem, we met that need, and continued to do so right on through to the time that the mainframe was shut down.

J. Kiefer: Great. As SPIRES was developing, all sorts of exciting stuff were happening in Silicon Valley. Certainly the Internet, and many other things. So I’m curious. Maybe we can start with one of those things, like the world wide web. How did SPIRES enable Stanford to hook into the web? How did it help develop that? How did it utilize developments that occurred somewhere else?

Guertin: [00:22:36] The web interfaces were written as separate entities. They weren’t built into SPIRES. So someone would write scripts to handle the web. These would normally be PHP programs, or HTML objects, or even JPEGs, and all—images, things like that. But when it got to the point where they needed to query a database, they were able to send a command to SPIRES that basically said what database they wanted to search, what kind of format they wanted to have it output in, and what search request it was. SPIRES would come up, do the search against the database requested, give the answer back in the form they wanted, which turned out to be HTML, because you’d write formats to format the data in HTML. And it would come out that way and go back to the screen. So the users would write these interfaces that would essentially search the SPIRES database and get an answer back. But all that logic was not in SPIRES itself, all that was in there was the search and display capability.

Kiefer: [00:23:39] The whole computer world changed quite a bit during the 40 years we worked on this system. And the beauty of being able to work on a
system for that long is that when a new thing comes up, you can see what you need to do to make use of that, or to allow people that use SPIRES to access those other capabilities. Like you’ve got—I think it was the late ‘70s, I went to a conference back in Boston where the big buzz was the relational database system—databases, and that was just something that E. F. Codd and his friend—

Guertin: [00:24:31] Codd and Date.

Kiefer: [00:24:32] Date, yes, had put together, and it’s a very elegant language to look at, and everybody wanted to get involved in it. Well, that’s sort of the basis that Oracle, and Sybase, and DB2, and all these other systems are using now. But in a way, SPIRES—the way we store data in SPIRES is so much more—

Guertin: [00:25:00] Robust.

Kiefer: [00:25:01] —”Efficient,” I guess is the word, because later on, when we had to build interfaces to relational systems, I built the Table part of SPIRES where we could generate tables in SPIRES to look like Oracle or Sybase tables, and had a series of processes that would convert a SPIRES database to these other systems. Oracle and Sybase were coming into real prominence at Stanford. So the old data had to be converted. When I looked at, say, the Student database at Stanford, and generated the tables to put the data from one record of SPIRES into Oracle, it generated 128 separate tables. So we could represent in one record what took 128 tables, but probably many records in each of those tables, so that the data in our system was very compact compared to the spread-out version that you get in a relational system. But those are the big things. That’s the sliced bread these days. Although, in some respects, XML, the way that they are putting documents together, like in XML, comes back to look like SPIRES—

Guertin: [00:26:44] Exactly.

Kiefer: [00:26:44] —You realize that maybe that design that Tom Martin had back in 1970 was the way to go in the first place.
**Guertin:** [00:26:52] SPIRES is a hierarchical database system as opposed to a table. A table database system says each row of the table is unique, contains single occurrences of items—

**Kiefer:** [00:27:04] Think of Excel.

**Guertin:** [00:27:04] —Across the line. It’s like an Excel spreadsheet. And you don’t have multiple occurrences. Now, whenever you do need to have multiple occurrences, you have to have a separate table to hold all the various rows of that structural—what we call a structure, the way that looks. So the main record says you’ve got to go over to all of these various records, and that’s another column of the table. You have to go to these records to find all the information about address. So that would be the street address, the city, the state, the zip code. So if you’ve got a record with multiple addresses, you have to have a separate table that shows those multiple addresses. They’re not built into the record you’re looking at.

In SPIRES, everything is like Chinese boxes. Each box contains a bunch of stuff. But you can have multiple boxes of the same kind of stuff. You could have multiple occurrences of address in a record. And the data is variable in length, and there are no fixed columns in SPIRES. There are no fixed columns of information like you get in an Excel spreadsheet. So what happens is SPIRES stores only what it needs to store, nothing more, no matter whether the value is two bytes long, or 32,000 bytes long. It only stores what’s needed and nothing more. Whereas in—I just heard this last week. This is—God’s truth. They’ve got a system where they put this thing—I think it's Remedy. They put all the Remedy tickets into the system, and the longest defined field had to contain the longest defined description that was included in a ticket. So this thing was thousands of bytes wide. And every single row of the table was that width, even if it was only two characters, like “Hi.” So this thing took terabytes of space. Well, that doesn’t sound so bad until last week they lost the Remedy system. It went belly up on them, and they had to recover it from the backup. It took 19
hours to restore it.


Guertin: [00:29:17] Nineteen hours to restore it. You know, that’s incredibly bad.

Kiefer: [00:29:22] Yes, there’s another thing, that when the Student System was probably four or five databases max. There was Courses and Students, and—

Guertin: [00:29:31] Housing.

Kiefer: [00:29:32] Whatever—yes. And so those five databases, I don’t know how big they were, probably a few gigabytes. And that included data going back as far as Stanford has students. But when we converted it to Oracle, then they had—oh, this was PeopleSoft. The PeopleSoft system, as far as I can remember, they were talking about 12,000 tables to represent it. And they were talking about several terabytes to represent the same data that were in a few gigabytes in SPIRES. So—even though the relational systems have an elegant way of looking at things, it’s very inelegant the way that you have to—

Guertin: [00:30:29] Store it.

Kiefer: [00:30:30] —Do things.

Guertin: [00:30:31] There’s right now over 4,000 databases still in SPIRES, and the entire amount of data physical space is only—is about 100 gigabytes. That’s the amount of total space. And that includes all the student records, housing records, financial records, any file you can imagine that was built in SPIRES since 1970, we have it archived. And the entire collection takes—because of the way we store it as compactly as we do — only 100 gigabytes. So you can put that on a 120-gigabyte drive and have everything that’s ever been done in SPIRES.

J. Kiefer: Now, how about email? Was that like an application that somebody
developed somewhere else in the University? I remember working at Stanford in the early ‘80s, and there seemed to be an internal kind of email.

Guertin: [00:31:37] Yes. SPIRES had its own email in the sense that you would simply put together a database that would contain the text that was being sent, and who sent it, and who it was going to, and all that. And basically, all these were simply nothing more than data elements, or fields, in a record. Everything’s variable length, so you only store what you need to store. And they’d be keyed off of the account number of the person who’s going to receive it. And you would put multiple copies, multiple extensions—of this structure in there for every single email that was sent to that person. So someone would go into SPIRES, would type up in a text file what they want to send and they would say that they wanted to send this to somebody else. That would be read in, stored into this database and into the record that was keyed for that person that’s going to receive it.

When the person who logs in is supposed to receive it and they go into SPIRES, they get a notification that they’ve got mail waiting. So the user would issue a command to show the mail. And it would get their record and show them all their records. And they could delete whatever they wanted to delete out of it, and that would simply delete an occurrence of that structure from the record. They could basically pare it down to zero if they wanted to. Or they could keep it. But it was the very first simple email system being done using a SPIRES database to hold the data.

Kiefer: [00:32:53] And what was interesting about that is that the person who developed the original, when I don’t know, he probably just did it on his own. Jeff Mapes was—I guess a programmer. I don’t know. He must have worked on applications at Stanford.


Kiefer: [00:33:05] For applications. Yes. And I think it was just his idea—well, could I write this thing? And that’s sort of the way a lot of things happened
at Stanford, that there were people that just—they decided, wow, this is a very interesting thing, and the more I learn about it, the more I get ideas about what it can do. And so you had many people putting these applications together that would never have been started if management had directed them to, okay, build this thing. There were certainly a lot of those, let’s build a—the Student system or whatever else. Certainly the Student system wasn’t done only by somebody getting a bright idea. But it was just the way that SPIRES was used. That’s how we got so many separate databases that, you know, private databases people did in restaurants, recipes. Anything you could think of, people would put together and make it available to others.

Guertin: [00:34:02] I can remember Jesse Caton, for example, who came up with the idea of a way of being able to write documentation, chapter by chapter, paragraph by paragraph, and organizing it in such a way that you could turn around and you’d—and deal with an individual section of a document as a record in a SPIRES database, but then turn around and ask the system to publish the entire document to get it to come out in printed form that would contain all of the appropriate indentations and bullet marks and everything else, footnotes and cross-links between the footnotes, and even build an index at the end that tells you where everything is in the document. Well, once he had developed this, somebody says, hey, this works for any document. So they began to copy it so they could start doing their own documentation that way. And that’s still in use today.

Kiefer: [00:34:54] And that’s how SPIRES’ documentation is maintained. Right?

Guertin: [00:34:57] Yes. It’s called SPIKE now.

Kiefer: [00:34:58] SPIKE, yes.

J. Kiefer: Okay. So then, back to the email for a second. So that became an internal thing.

Kiefer: [00:35:08] Initially.
J. Kiefer: When would you say it really got going?

Kiefer: [00:35:12] Late ‘70s maybe.

Guertin: [00:35:13] Late ‘70s. Yes.

Kiefer: [00:35:14] Before that we had WYLBUR mail.

Guertin: [00:35:17] Right.

Kiefer: [00:35:18] —Where you could—there was a “send” command? What was it that WYLBUR—

Guertin: [00:35:24] You could send a message to somebody, but you really didn’t have the email capability. You could just simply send a message.

Kiefer: [00:35:30] Could you have texting, too?

Guertin: [00:35:32] Yes. Direct message.

J. Kiefer: At what point was email developed so that it could be sent to somebody outside Stanford?

Guertin: [00:35:47] I don’t think that ever happened. People had to log in to use SPIRES to get their email. But then, you know, Eudora and Thunderbird, and all these other email clients came along that were basically much better clients for handling email, and so the use of SPIRES to handle email diminished rather rapidly. And people went off to using these clients to—to do regular traffic between different universities and so on. But you have to understand that the—you know, the Internet didn’t really start until October of 1969. And, you know, here we are coming in about a year later in June of 1970, so the Internet’s a fledgling thing out there. And we wanted to be able to do email within the University, and the easiest way to do it was with the SPIRES database. But later on when the—you know, we get past the startup processing of Internet, and get to the point where people are beginning to produce products, those products took over. And that’s fine. We have no problem with that.
Kiefer: [00:36:54] Actually, people outside of Stanford could use our email system, and that was all the other people from other universities, but they had accounts at Stanford through their universities.

Guertin: [00:37:07] So they could remote log-in.

Kiefer: [00:37:08] So we had programmers at University of Alberta, Michigan, Toronto who would send us messages. We could go back and forth and keep up with them. But they had to have a SPIRES or a—

Guertin: [00:37:29] Stanford account.


Guertin: [00:37:32] Now today—now just recently, by the way, six months ago on the UNIX system where SPIRES is running today, I discovered that there was a new UNIX gadget called MUTT, M-U-T-T, that allows you to send email with attachments. And you simply specify to MUTT where the attachment is. It will not send it as the body of the text, but instead will send the attachment as an attachment. So I converted the SPIRES mail command to have an attach option to say attach this object. Okay. And now in PRISM when you come up with a report and you want to email it to yourself, it takes the report and sends it to the person as an attachment, so that now the body of the text simply says “read your attachment.” That’s the body of the text. And then the attachment is the entire report that they can turn around and print, do everything they want to, without having to copy and paste it out of the body of an email message. It’s basically a separate object. So SPIRES even now is still growing, still making use of the new developments that are out there. It’s not a dead object.

J. Kiefer: What about Y2K?

Guertin: [00:38:45] That wasn’t an issue.

Kiefer: [00:38:45] What was interesting is that in Stanford, people were getting very upset because they knew that Y2K was coming. This was back in the late
‘90s, I guess, and that SPIRES would have to go away because it obviously couldn’t handle things like that because everything was done in the 1900’s. But one thing that people did not know is that from day one, all the year, four-digit year and—


Kiefer: [00:39:21] —Century, the year, and—

Guertin: [00:39:22] Month and day.

Kiefer: [00:39:23] —Month and day were always kept internally as full dates. So if a user application dealt with two-date—two-digit dates, year, centuries, and without the century mark, then they would probably have to do something to their programs in order to make it happen. But SPIRES itself was always able to deal with all the dates, and it always stored dates as full dates. So that simplified things a lot. But I know Dick had to work with a lot of formats and protocols for the Student system was it?

Guertin: [00:40:07] Yes. Students—several others. But the key was that if someone entered a two-digit date, like 01 through 99, or 00 through 99, it was assumed to be 1900’s. You assumed the century was 19. But as we were getting close to 2000, we wanted 00 to mean 2000, not 1900. So we modified the rules that allowed input of two-digit years to say, okay, you can have a starting point which isn’t zero-zero. You can have a starting point say at 30, which means 30 to 99 is 1900, and 00 to 29 is 2000. So now we’ve simply produced a 100-year range, but it doesn’t go from 0 to 99. It goes from 30, wraps around to 29 and crosses a century boundary. Okay. So they still are dealing with the data in two-digit formats where they’ve got only a two-digit year, but the system now knows that when it gets 10, it’s 2010, not 1910.

Kiefer: [00:41:10] Yes. Now, this was something that we thought of—we worked on for years before. We did a lot of work to make sure that SPIRES didn’t have any problems. And I remember one or two glitches in the year 2000 where something didn’t work right, but it was pretty minor. So all the worry and
hand-wringing that took place before that was unnecessary. It was a pretty smooth transition. I think that business of the window—windowing the dates is something that can come up to bite you as soon as you go past the window, then people that still use two-digit dates are in trouble, especially around 2100. They’re going to come up with the same problem.

Guertin: [00:42:09] If SPIRES survives that long.

Kiefer: [00:42:10] But you know, any system that had to go through Y2K had to come up with some mechanism to make it happen.

Guertin: [00:42:19] By the way, that rule that has that sliding window, you can go in and change the base of that rule. So instead of saying it’s 30, you can say it’s 40 and recompile the database. And now all of a sudden, it’s 40 to 99, 0 to 39. So as we’re approaching 2010, there’s some that—some people who put in 10 as the base. As you know, when you got to this year we have to look back and say maybe we should make it 20, push it ahead. And that’s all we’d have to do. The whole thing would move. Now anything that comes out of the database that’s outside that range comes out with the century included. So if somebody’s birth date is 1901, it comes out as 1901, not ‘01 because if came out as ‘01, it could go back in as 2001, and we don’t want that. So that window only applies for outputting of two-digit years. Once you get beyond the two digits you get a four-digit year output.

Kiefer: [00:43:14] One little comment on that. I always write dates as four digits, as, you know, full century year-year. And whether I’m signing a check or anything else, just because that’s the way it should be. Nobody should have a two-digit date.

Kiefer: [00:43:35] Oh, yes. I think of SPIRES as a toolbox. It’s just chock full of tools that other people can use to do what they want to do and for application programmers. Certainly the SPIRES system would grow to handle these tables or whatever else. But there are a lot of functions that
people can use that we kept adding, little things that would enhance a person’s capability to do what they wanted to do. And always we had user groups that said we’d like to be able to do such and such. And we would always give them a tool that would enable them to do what they wanted to do. So that’s why I think of it as a toolbox.

Guertin: [00:44:26] You know, one of our finest achievements in that line is going back to dates again. Somebody came to us, I think it was the financials group, every time we come to August, the end of our fiscal year. And sometimes the 30th or the 31st of August occurs on a weekend when we’re not working. And so we can’t finish the books on time. We need to be able to extend August. So we allowed them to define a rule that says that all the dates are normal, except August can have 32, 33 and 34. Those dates are still August dates, but they’re beyond the normal realm so that you can get your processing done in August, supposedly by rule. And then we allowed people to input dates in what we call relative dates. So you can do things like today, yesterday, tomorrow, the last day of last month, the 1st of last year. And you can actually put those dates in that form and the system will figure out what the date is and store the appropriate century, year, month, day for the particular—and as time moves along, the last day of last month changes. But the system knows what today is, and if it knows what today is, it could figure out what the last day of last month was. So it’s just a question of being able to compute it, which is what computers do.

Kiefer: [00:45:53] There’s another thing we did that’s similar to that, and that has to do with personal names. Right? Dick worked on that part of it. But names will always come out if they use the proper processing rules as last name first.

Guertin: [00:46:10] Comma, first name.


Kiefer: [00:46:15] Standard name. And that way things are always in alphabetical
order, if you need them that way.

J. Kiefer: So you had different application programmers in different parts of the University, like finance and other areas. Correct?

Guertin: [00:46:3] Student—yes, projects. Whole projects of people.

J. Kiefer: And they were like clients, would you say?

Guertin: [00:46:36] They'd come to us with requests for things that they didn't have. Early on in the '70s, we had developed about 150 rules that could be built into a database that would allow you to transform an input value from a numeric string of characters into a binary quantity that could be stored. And then the reverse rule that would take the binary quantity and output the actual character string equivalent. Rules like that. Well, people kept coming to us with “we need a rule that does this.”

We have to keep writing all these new rules. And we said to ourselves, wait a minute, this is getting above and beyond what SPIRES should be doing. These people should be able to write their own rules. And so we came up with a rule called “the user defined processing rule,” where they write their own script of what they want to have it do to compute the answer they want for either input or output. And now we didn’t have to do any more addition of new rules. They simply went out and added those to their databases. And so if you wanted to compute what the person’s age is, they’d compute the age. Based upon today’s date and the birth date, they’d figure out month, day, year, how many months, years, and days old you are, and they would be able to communicate an age for you. We didn’t have to put an age processing rule in, they would put their own in.

Kiefer: [00:47:56] There were quite a few different groups of application programmers, and I think there were departments that at one time had their own programmers. And then sometime or another, about every five years, those programmers would become part of ITSS or whatever the equivalent was, and they'd be working in conjunction with their original departments,
although they’d be in a different department. Then they’d go back to the
original departments that—the whole structure of what ended up as ITSS
changed from SCIP to—what—CIT? I don’t know—the name kept
changing.


Kiefer: [00:48:44] Yes. And so there was always a group of programmers that knew
how to write applications that would be available to work with various
departments to build these applications. And they were generally part of
what we called a user’s group. One of these was called Boxers or—

Guertin: [00:49:07] There was Boxers. Yes.

Kiefer: [00:49:08] And there were other user group names too.

Guertin: [00:49:12] And they all appear in the Stanford video that we have.

Kiefer: [00:49:16] Yes. So that was a rich group of people that really kept us
hopping for all those years.

J. Kiefer: So you think that they really used the potential of the system.

Kiefer: [00:49:28] Yes. They defined the potential in a lot of ways that—I don’t
know. Well, one thing that happened is all the applications that grew up with
Stanford were based upon the way Stanford did business. And they were
using internal people that worked with the secretaries or the department
people that—

Guertin: [00:49:53] Provost.

Kiefer: [00:49:5] And developed things that they needed. So when later on Stanford
started turning to somewhere else, the conversion became pretty rough from
SPIRES to whatever other systems they were looking into.

Guertin: [00:50:18] We’ll get to more of that later on, I think.

J. Kiefer: Now Bill, you were going to mention some of the people that you felt should
be mentioned in developing this.

Kiefer: [00:50:27] I think first and foremost would be Tom Martin. At the beginning Dick and I each consulted with him on his original design for basic SPIRES internals. Of course we did a lot of design and implementation on our own for some things, but the BNF language, the command language is driven by something that—

Guertin: [00:51:02] The Backus Naur Form.

Kiefer: [00:51:04] That was something he developed. The way the compiler worked, which I think is just a fascinating way to compile something, we used that—he developed that capability. He described it. Essentially, Dick and I did implementation.

Guertin: [00:51:22] You mean the file compilation process.

Kiefer: [00:51:23] Yes. The file compilation.

Kiefer: [00:51:25] Tom also developed the way the structure of the data in records were—we were talking about the way the compact form that was put together, the tree structure that we used in our file systems.

Guertin: [00:51:38] The passing logic.

Kiefer: [00:51:39] The passing logic that we used for indexing, and the whole concept of the way indexes look in SPIRES which probably looks different from anywhere else in the world.

Guertin: [00:51:48] SPIRES indexes are nothing more than other records in the database. In most systems, they’re off somewhere else in the backwater.

Kiefer: [00:51:54] So in SPIRES, one module can handle indexes or full records or whatever else. There’s so much that is interconnected—

Guertin: [00:52:05] In common.

Kiefer: [00:52:06] I’m saying Tom Martin was indispensable.
Guertin: [00:52:12] Absolutely.

Kiefer: [00:52:13] Another person that was very interesting was Joe Berman, who was writing his Ph.D. thesis, and he came to visit with John Schroeder. Schroeder called me in because Joe needed certain things, and in formats, which I wrote. And he started the development of this protocol language that Dick was talking about, the scripting language. And I found him to be kind of sullen. I mean, why should I deal with something that these hicks over here, non-academics, are working on.

And the more he got to see what was in SPIRES, the more excited he got. Pretty soon he was writing code and actually consulting with us to work on it. So that’s what his thesis was in. John Sack was pretty important. He was in the SPIRES users—what do you call it—the group that services the users when he first came onboard, and—

Guertin: [00:53:26] He was a consultant. People would come to him. He was a user consultant. So he knew what SPIRES did, but he didn’t code it, or had anything to do with its coding.

Kiefer: [00:53:35] But he just had a knack for seeing things that could be done, and he came up with this system when we finally had a full-screen system he called the Ed Shaw System, which I think had circulating lights or something, and Ed Shaw was the big head of the department at that time, and he came in, and John showed him this thing. Ed got excited about the potential of what he saw. This was before terminals were really installed in offices around Stanford. And he along with John—I think—I may be wrong, but I think that this is where terminals were actually installed - on the professor’s desks. And then John later became our manager, and he was very good. And now he’s the head of the HighWire System at Stanford. Lynn McCrae and Sandy Laws actually worked on developing the original Socrates system which the library used for quite a while. Now it’s under some other
system—I don’t know what.

Guertin: [00:54:59] It’s PRISM. It’s part of PRISM now.

Kiefer: [00:55:01] Well, but the one that’s on the web, I’m not sure where that might have come from - UCLA or something. I’m not sure where that’s from. And Lynn was the lead architect of the PRISM system, which was for full-screen applications. Before that everything was line-by-line terminals. John Klemm wrote most of our SPIRES documentation. And the problem with SPIRES documentation is it documents the complexities, everything about SPIRES, and it’s probably about two feet tall when you put all the manuals together. But he was always right with us as far as, “What—how does this work? Are you sure? I don’t see—I don’t understand this.” And he would ask questions, and he would find problems that the users could not deal with. So in some respects, he was our quality control—quality assurance person.

Guertin: [00:56:02] But he also got the documentation out so that the users understood what they could do.

Kiefer: [00:56:06] And I don’t want to forget Peter Tuttle, because he was also someone who was extremely important in enabling the application programmers to understand how to write applications, how to put files together, and provided all the tutoring they needed in this system. He was there as a consultant to the applications people.

Guertin: [00:56:33] He was mainly a teacher. People would come to seminars and he would teach classes on how to build a database or how to search a database or how to build an application using PRISM. He was basically our primary instructor. And he taught sometimes classes to other people who became instructors. This was a way of being able to get a group of people to come in from a department, learn what they needed to know for their department, then go back and teach their own crew how to do the things that they needed to do. So he was a key person. He’d come to me with, “We need this.” And I’d say, “Okay.” So I’d start working on it. Then he’d say, “And while the
patient’s open, we need this.” That was one of his favorite phrases - “While the patient’s open.”

**Kiefer:** [00:57:26] “While the patient is open.” Right.

**J. Kiefer:** So you mentioned back there about terminals. The professors had them on their desks. About what time - ’80s, ’90s - was this? And when we’re talking about terminals, were these PCs or something different?

**Guertin:** [00:57:48] Initially, they were just terminals, ADM-3A, or any of these devices. You plug them into a socket in the wall, and all they did is give you a screen. There was no real computer built in. There was obviously logic circuitry in it to take care of the keystrokes and code them and send them back and forth. But initially they were just dumb terminals. In fact, they were called dumb terminals.

**Kiefer:** [00:58:13] Yes. And I don’t know what years—probably—

**Guertin:** [00:58:17] ‘80s. Late ’70s, early ’80s.

**Kiefer:** [00:58:19] Late ’70s. When we started out, essentially, people used what looked like typewriter terminals to interact with the system. But then you got—

**Guertin:** [00:58:30] But they actually had a twitching ball. We’re talking about IBM Selectrics.

**J. Kiefer:** Is that what you used when you first came?

**Kiefer:** [00:58:38] Yes. And it was a few years before we got the dumb terminals. And that was a big step. And that was probably what John Sack used to do with his Ed Shaw system, which was probably late ’70s, I would think.

**Guertin:** [00:58:52] Yes. CRT screen thing. The original IBM Selectrics—one of the things that the data center had to do was they had to send out a pulse every five to ten seconds that would cause the ball to twitch. Basically, a shift. They had to do a shift to cause the ball to go tweet-tweet, so that you knew
the terminal was still alive. Otherwise, people would walk up to it and they wouldn’t know whether the terminal was operational or not. But if the ball would twitch every once in a while, tick—tick—they’d say, oh, okay, it’s working, it’s active.

J. Kiefer: So that you worked with this Selectric-looking thing, and then you got these terminals that you mentioned, dumb terminals. And then you say some of the faculty also got dumb terminals?

Guertin: [00:59:41] Probably.

Kiefer: [00:59:41] I’m sure that’s what happened.

J. Kiefer: But that wouldn’t be all of the faculty, because I’m just thinking that some may not be—

Kiefer: [00:59:49] Well, I think they made them available and tried to make sure that people got them. But this is not something I was a part of, it’s just what I’ve heard, that they were trying to get them out to as many departments as possible.

Guertin: [01:00:03] But there was some point, and it was either in the ‘80s or early ‘90s where we had, like you said, one—one day, or one instant, 2,700 people logged into the computer at the same instant all doing something.

Kiefer: [01:00:21] Using SPIRES.

Guertin: [01:00:22] Using SPIRES.

J. Kiefer: So when did you get PC’s to do your work?

Kiefer: [01:00:28] I’d know it took me a while to even accept a Mac on my desk. I remember Rich Levitt—he was our supervisor at the time—would say you’ve got to get off that terminal and get a Mac. You need that. And I said, well, why, I can get my work done this way very well. Finally—this is probably early ‘90s, I said, “Well, okay, I’ll try it.” So I got my Mac and was able to do my work just as well as before. But then I found the Internet. And I said,
“Well, I'll try to search for the Cleveland Indians.” And I did, and their Chief Wahoo was on my screen in full face. And I said, “Okay, I'm hooked. That’s all I need.” So that’s how I got mine. But I was late.

J. Kiefer: Okay. And you, Dick, were already using PC's?

Guertin: [01:01:35] I started with a Macintosh when the Macintosh first came out.

J. Kiefer: In 1984 or so.

Guertin: [01:01:42] ‘85. My daughter had to go off to college and I went out and bought a little Mac for her to have. And as I was showing her how to use the computer—because I was obviously involved with computers and she wanted to help me, I became intrigued by this little box. And when it came to an offer from my boss to put one on my desk, that was not a problem for me. I said, yes, fine. But I remember going back to the Apple II. When personal computers first came out. Parker Gillespie was playing around with this Apple II and he was playing some game on it. He kind of got me interested in the idea of a little computer that could do things on your desktop. And I started looking into that.

J. Kiefer: Interesting. So meanwhile, the other piece of hardware in all of this would be the IBM mainframe. Right? That was the engine that made everything run.

Guertin: [01:02:37] And that changed. It started out as the IBM 360 with very limited memory. I think maybe it only had a few megabytes of memory. It wasn’t very big. Then it got upgraded to having virtual memory, which was a new concept - memory stored on a drum - and it used something called page swapping where you’d swap the pages in another memory. This opened it up to make SPIRES a much more viable product for multiple user access, because their sessions would get archived off onto that swap space, and they could still be able to run. So that’s why the load on the machine went up dramatically with virtual memory. And SPIRES took off. I mean it literally took off at Stanford.
J. Kiefer: So when was the mainframe retired?

Guertin: [01:03:29] Well, I think it was March of 2004.

Kiefer: [01:03:37] No. It—well, it was—

Guertin: [01:03:38] But had you already retired?


Guertin: [01:03:44] Went away. So—yes, it was that year.

Kiefer: [01:03:46] I think it was December of 2003 that was going to be the last—they had to get rid of it, because otherwise they’d be paying another $200,000 a year, or a month. I don’t know.

Guertin: [01:04:01] It was very expensive.

Kiefer: [01:04:03] Yes.

Guertin: [01:04:04] It was a monthly lease.

Kiefer: [01:04:05] Right. So essentially, they forced SPIRES to go away in a lot of respects and have all these other systems take over, which they’d been working on for quite a while.

Guertin: [01:04:22] Which brings us to the part about—at some point in here we’re going to get to the emulator.

J. Kiefer: So the mainframe was—what happened? It wasn’t necessary, it was less and less necessary?

Guertin: [01:04:39] Oh, it wasn’t that it wasn’t necessary. In my opinion, that wasn’t the reason. It was a financial problem. It was getting to be too expensive. And these little pizza boxes, as they called them, the Solaris systems, these UNIX systems were really dirt cheap. And the University could buy a whole bunch of them. And a gang of those little boxes would do what a mainframe could do. And they would own them. They wouldn’t lease them, they’d own
them. So there’d be a one-time investment and you’d have them for a long, long time. And it was like $100,000 a month to lease the IBM mainframe, and this was just money out the door to IBM. And they said, no, we can’t afford to do this anymore. So they wanted to get rid of it.

**J. Kiefer:** So what about your emulator?

**Kiefer:** [01:05:23] Well, before we do that—in actuality, the expense wasn’t in the mainframe. It was in converting the systems to use something besides the mainframe. And they essentially forced the mainframe out by building all these other systems, using Oracle. But the expense that they went through to do that, to rush it through, as far as I’m concerned, was so—they hired all these consultants, they had a lot of people. And, you know, the millions—there were millions of dollars spent in trying to get—

**Guertin:** [01:06:06] get rid of Spires, get off the mainframe.

**Kiefer:** [01:06:06] —get rid of SPIRES, essentially. And so I think it could have been done. Even if you’re paying a million dollars in a year for holding onto the IBM mainframe for another year, it probably would have simplified some of the problems that took place when they went away from SPIRES. I think they could have done it a lot more smoothly than they did. Anyway, go ahead.

**J. Kiefer:** So, Dick, you developed something called an emulator, and that did what?

**Guertin:** [01:06:42] Well, let’s go back to why it was done. There was a project at one point being done by the consortium, which is all of the other universities who were using SPIRES as a system, and we would have these semi-annual or annual conferences. There’d be one at Stanford, and then there’d be one at some other site. I could go to Alberta or Newfoundland or some place else. And so all these universities were making use of SPIRES, and then we had this project — very important — called the SPIRES in C Project, which is the conversion of SPIRES to be run in a C language so that they could basically transport it to these little pizza boxes off the mainframe. Well, they
worked on that for over a year. And at the annual conference, when they were presenting it, they asked the guy who was doing the development work whether SPIRES was going to have all the capabilities that the existing SPIRES had. And he started to say, well, we won’t have formats, and we won’t be able to compile, and we won’t be able to do this. And there were so many key things that it wouldn’t be able to do that people were beginning to grumble.

Now, I had already had a conversation at the semi-annual conference when I went to Alberta with somebody who was saying, wouldn’t it be possible to emulate the IBM mainframe instruction set and be able to run SPIRES on any kind of platform by emulation? And I said, well, yes, that would be possible, but it probably would be inefficient. So this guy challenged me to go back and try it. I went home. This was Thanksgiving. And I spent Thanksgiving through the end of the year working on an emulator, running it in Small-C on a Kaypro machine. And I got to the point where I was able to load the program up and execute the program and have it do things, and I began to realize it’s not as slow as people think.

So I started writing the emulator to be able to do everything that ORVYL did. ORVYL was our timesharing system which ran SPIRES. So as long as I was able to mimic, attach a file, read a file, write a file, all those various requests that SPIRES would put into ORVYL to do things, I was able to come up with and emulate all the instructions—add, subtract, multiply, divide, shift, everything. Then I was able to run SPIRES under emulation by loading it into memory and treating it like a recipe where I’d read the instructions, see what it is, and then in the code, I would do what the IBM equivalent of that instruction would be. Then I’d go to the next instruction and do that one. It’s like reading a recipe and doing what the recipe says—add the sugar, stir it up, et cetera.

Well, it ran very well, and I got it running on my Mac. At the next conference, I brought a laptop computer out, a laptop Macintosh out to the
podium, and I plugged it into the overhead projector, and I ran SPIRES. And I was doing searches. And then I held the thing up and I said, “By the way, this is not connected to anything. It’s all coming off this box. I’m running SPIRES off this Macintosh.” And the audience went bananas. They couldn’t believe it. And that’s how we progressed to where we are today. I think it was Louise Addis at one of these annual conferences when the guy was telling them what they wouldn’t be able to do, and I poked Louise and I said, “You know, I’ve got this emulator that does everything.” So she stood up and she challenged Bo Parker. She said, “Bo, if this thing won’t do all those things, why don’t you try Dick’s emulator?” The next year when I came out and presented that, that was it. That was the end of the SPIRES in C project. Because there was no way they could beat what the emulator was doing.

Kiefer: [01:10:40] Right. It could do everything. In some ways, your emulator hastened the demise of the mainframe—

Guertin: [01:10:52] I think so.

Kiefer: [01:10:53]—because there’s no way that the University could get along without SPIRES, because it still hasn’t gotten along without SPIRES. Right?

Guertin: [01:11:05] That’s right. That’s why I’m still employed.

J. Kiefer: Okay. And so tell me again, what year was that emulator in full use?

Kiefer: [01:11:15] I know that I was writing C—a lot of what I did in preparation for C was to try to re-modularize a lot of the internals in SPIRES so it would be easier to translate the code, the PL360 code into C.

Guertin: [01:11:35] Yes.

Kiefer: [01:11:36] And Dick essentially—I didn’t have the emulator in mind. What I had in mind was doing as much as I could to clean up SPIRES to make it translatable.
Guertin: [01:11:53] Oh, yes. To go from 24 bit to 31 bit addressing, for one thing.

Kiefer: [01:11:56] Yes. There was—well, but mainly for C, it was—because Sandy Laws asked me to work on the C project. Well, I kind of cheated. I wrote some processing rules in C but I decided that—to translate was the only way that it would be possible to move code. And that was about ‘87 or so. So it was probably after that that you started the emulator.

Guertin: [01:12:23] The emulator wasn’t until late ‘90s. I actually—

Kiefer: [01:12:26] But when did you start it though?

Guertin: [01:12:27] Oh, I started it earlier—I started it in the early ‘90s.

Kiefer: [01:12:30] Right. That’s what I thought. Yes. So the fact that it was available. Anyway, yes. That leads to the other things that we’re talking about.

J. Kiefer: So if we’re talking about this emulator that’s sort of flowering in the early ‘90s, would you say, it’s still being used?


J. Kiefer: Growing. Okay. During that time, tell me about the University’s administration and what they thought the future of SPIRES was, or was not.

Kiefer: [01:13:00] Well, one thing about it was that I think from almost the beginning, there were anti-SPIRES people in the University. And the first one was when they decided to get rid of the OASIS system and move it to SPIRES, that made a lot of people angry, the people that were working on this other system. I can’t blame them. But the problem was they just had a very small computer with a 100K of memory, and the amount of work that had to be done to do anything would take so long. I think I figured that I was sent over to work with Peter Norton to see if maybe the OASIS system would be usable instead of building our own system. And when I saw that there were numbers like 50 IO’s to get a decent search done, I brought it
back and told the management, you know, I don’t think it’s going to work.
Anyway, so—

**Guertin:** [01:14:03] And John Sack was the management, and John agreed.

**Kiefer:** [01:14:05] No, this was way back at the very beginning.

**Guertin:** [01:14:07] Oh, before John.

**Kiefer:** [01:14:08] This was at the beginning when we were first starting out.

**Guertin:** [01:14:10] Oh, okay.

**J. Kiefer:** In the early '70s?

**Kiefer:** [01:14:13] Yes. What was the first question? I got off—it was—

**J. Kiefer:** No. It was just, when did the University administration—

**Kiefer:** [01:14:19] Okay. There were people against SPIRES. Well, later on when SPIRES was actually in use and being more and more used by application programmers, there were commercial systems becoming available—ADABAS, System 2000—I don't know if IMS would be classified as one of those, but there were several commercial systems, and the higher-ups at the University thought that they probably were much better than SPIRES because they were built by these companies that had all these people working for them and building them. And so—I think it was John Sack that had to do a study on comparing SPIRES against four or five commercial systems. And in every case SPIRES came out ahead, whether it was talking about the full capability of the system. I've got the report somewhere in my files. When he presented that, things calmed down for a while. But later on, again, people said we've got to get off the mainframe, therefore, we'll start this thing called Core Financials and we'll use GUI Interfaces. I guess this meant that the whole program would be on the PC’s. They had I don't know how many people working on that, I think two or three hundred.

**Guertin:** [01:15:52] Two or three hundred.
Kiefer: [01:15:54] —trying to build this system that I’m thinking, this is so easy, it’s working now, why are you doing this? Anyway, it turned out that it didn’t take long before the Internet—or the World Wide Web—became the way to do things, and the whole GUI interface project died. I heard numbers, and I think it was probably $60 million dollars down the drain on that whole project. That was early ’90s. And that’s about the time that the World Wide Web came about. And funny enough, the first World Wide Web search that took place outside of Switzerland was using SPIRES.

Guertin: [01:16:44] At SLAC.

Kiefer: [01:16:45] Yes, SLAC. That was Louise Addis. She decided, hey, we can get this physics database available to our users all over the world through this thing called the World Wide Web. And that was using SPIRES as the basis, not that I cared anything about the World Wide Web. But it’s kind of an interesting thing.

Guertin: [01:17:07] The first website outside of CERN was at SLAC. It opened on December 13th, 1991. And if you go and look up the history of the web, there’s one site that says it was ‘92, but that’s simply because by the time you got through Christmas and it had finally got known that it was at SLAC, it was 1992, not ‘91. But it started in ‘91, December 13th, Friday the 13th. And SLAC was the killer app. I mean that was the system that showed that the World Wide Web was a viable tool for being able to get at data, and was doing it through SPIRES.

J. Kiefer: Okay. And GUI stands for?


J. Kiefer: Let’s follow through then with SPIRES. We get to the Year 2000. In the early 2000s, I recall PeopleSoft was brought in. Why did that decision come about, and what was the result?

Guertin: [01:18:24] Well, at that time I believe one of the chief protagonists, one of
the people who was working for the University was [then Provost] Condoleezza Rice. And she was kind of in control of a lot of things, and she basically said that the University should not be doing its own programming. That it’s very difficult to hire someone into a job that requires that you learn PL360 to implement SPIRES because PL360 is not a transportable skill. Nobody in the commercial world is using that language. So you can’t hire somebody to learn that language and expect them to be able to have any job mobility. So people wouldn’t come.

**Kiefer:** [01:19:05] You know, also the languages that we use, the programming languages were protocols and that were specific to SPIRES.

**Guertin:** [01:19:13] Right. They were unknown outside of the University.

**Kiefer:** [01:19:15] So the idea was that you should be able to find a commercial off-the-shelf product to do the thing. And I think that there were a lot of people at the higher level—trustees, the board, that really wanted to get off of SPIRES because they couldn’t imagine that they could continue with a home-grown system. In a lot of respects, they were right in that much could be done by commercial systems. I think it’s just the way that it was done was not good because they just moved too quickly. I think there could have been an evolutionary way they could have done it, because we built a lot of tools that would allow SPIRES to talk to other systems back and forth. You could send data to one system and have their updates come back to us, and we’d put them into the SPIRES files. I think that if they’d have picked and chosen with care what is best outside of SPIRES and what’s best in SPIRES and then taken it, looking at each process that way, it would have worked. I didn’t mind that SPIRES would go away, there’s probably still good reasons that was happening, but I think the way it was done was just forcing things down people’s throats, and it caused a tremendous amount of problems throughout the University. Some end users and departments couldn’t do their work anymore because they didn’t have the data to do it with.

**Guertin:** [01:21:04] There was a comparison that was done along those lines where
they compared the search and reporting capability out of SPIRES with the searching and reporting capability out of the Oracle Financials system. And I remember my daughter saying she had to get a report. And out of SPIRES, she had the report out in three seconds. In Oracle Financials, she said she started the report and she walked away from the thing and had to come back three minutes later to get the answer. So there were people who were frustrated by the fact that it was taking a lot of time. And the head of one department went so far as to buy punching bags on suction cups employees could put on their desk, so they could take it out on the punching bag while they were sitting there waiting for the system to deliver their information so slowly.

Kiefer: [01:21:54] Yes. And I think these other systems just used terminology that was not part of the Stanford language. Now to get the same thing done, you weren’t speaking the same language that you had at Stanford with the systems that grew up with Stanford. So it was a difficult process. It was a trying time.

J. Kiefer: Now when was the decision to lose SPIRES, stamp it out, or whatever, being made?

Kiefer: [01:22:44] It was the ‘90s.

J. Kiefer: Okay. So how does SPIRES continue to be used?

Guertin: [01:22:56] Well, I’m the only one who can really answer that because Bill’s been away from it now.

Kiefer: [01:23:00] Although I use my own private SPIRES.

Guertin: [01:23] Yes, SPIRES runs on virtually any machine now because the emulator has been modified to run on Macs, Intel or PowerPC’s. Runs on UNIX, it runs on Linux. In other words, it runs either big-endian or little-endian, which is a terminology that’s used to describe the way the data is stored on a machine. So it runs virtually anywhere that has a C compiler. And we
moved it to UNIX by simply taking all of the databases that were stored in the ORVYL file system, the actual SPIRES files. We just secure ftp’d them across to a UNIX platform as is, without changing them. And the emulator treats the files exactly the same way as it did under ORVYL, because it was designed to work as though it was replacing ORVYL. So all this stuff was moved over.

And when they decided to turn the mainframe off, people would log in, and all of a sudden, they’d be on a UNIX machine instead, and they would do all the same things they did before. Once they got into SPIRES or PRISM or any of those things, they couldn’t see any difference. It didn’t matter that it had moved to them. And that’s the way it’s still going. So there are still those 4,000 files out there, and there are 4,000 sessions a month, sessions where someone logs into the system and goes into SPIRES and does something, or goes into PRISM and does something.

There’s over 4- almost 5,000 sessions a month. Three hundred individuals are doing this on a regular basis. Sometimes there’re as many as 200 logged in simultaneously. And they’re doing 70 different applications actively changing the data, in 70 different applications and just searching through others. For example, when the Student System was migrated into the new form, the Network for Student Information project rewrote the whole thing to work in PeopleSoft or Sybase or whatever it was, they didn’t move all of the data. But then someone who had gotten their degree and had left Stanford years before whose records hadn’t been moved would come back to the University to get a post-graduate degree, and they’d say, well, I’ve already got a student number, you’ve got me in your database. No, it’s not. So they had to go back and get the data from the old system, which is why we had it all archived. Plus, there were rules from the IRS that said you’ve got to maintain financial records for over seven years. And when they moved the stuff to the new financial system, they did not bring the data with them. They just started brand new.
So people are still going back to search for and obtain the data. So there’s a lot of searching going on, of people, of archival information, but there are some things for which there never was found an off-the-shelf equivalent. And so all of the sponsored projects that are being funded by the government don’t have an equivalent. All of the faculty affairs, administrative stuff, don’t have an equivalent. And those are actively being changed on a daily basis.

Kiefer: [01:26:14] Also, what about the SLAC system?

Guertin: [01:26:18] SLAC is still running with SPIRES. And they’re trying to rewrite it in some other system, but they haven’t succeeded yet.

Kiefer: [01:26:25] And that is also replicated around the world, isn’t it? I mean the actual system?

Guertin: [01:26:29] Yes. They use what they call mirror sites. So SLAC has a site in, say, Russia that is another computer, and on a weekly basis they send them a whole brand new copy of the preprint file. And so each week, these various sites—Japan, Germany, Russia—there are all kinds of mirror sites—get this information and then the people there are doing the searching and retrieval just as though they were local.

Kiefer: [01:27:02] So they’ve got their own SPIRES systems.

Guertin: [01:27:04] They have their own SPIRES systems running on various machines using the emulator, because they’re all under emulation right now. There’s nobody with IBM mainframes anymore.

J. Kiefer: So what I’d like to do is ask you to kind of envision yourself that both of you were working fulltime in the SPIRES—as SPIRES was being developed in, say, the ‘70s, ‘80s, ‘90s. And how would you describe the atmosphere of your work group? Was it eight-to-five? Was there a lot of initiative? How was the interaction? Tell me about it.

Guertin: [01:27:43] Well, if you ask my wife, she would tell you there was no such
thing as me not working. I either would go to the office and work there, or I 
would be at home working at home. And sometimes that meant working on 
things because I was interested in it. It wasn’t because there was an 
emergency situation or anything like that. It was simply that I was just really 
interested in making SPIRES, or the emulator, or something, get better and 
better and better. And so I would be spending time at any hours that were 
necessary.

I woke up this morning and realized I’m trying to get something to work in 
AFS space, and the AFS, the Andrews File System is what it’s called—and 
I’ve never worked in that kind of space. And I finally find out it doesn’t have 
the same security that UNIX has. The UNIX security is completely 
oblivious to Andrews File System. It uses something called access control 
lists. Well, I wasn’t geared up in the emulator to deal with access control 
lists, but I’m going to have to be, because that’s the way I’m going to have to 
deal with it in AFS. And if they terminate SPIRES, which they are planning 
to do at the end of this year, 2010, I will get terminated at that point. I think 
I’m going to be laid off.

But once that happens and they terminate SPIRES on the UNIX platforms 
where they’re running it, it’s going to have to move. It’s going to have to 
migrate. And one of the places it can migrate is to this AFS file system 
because every individual at Stanford who has SUNet ID has two gigabytes of 
space for free. And they can do anything with it. They can put up websites, 
they can put up database systems, they can put up whatever they want. So 
here is a way of being able to take all these files that are running right now on 
the SPIRES machine and put them up under various places with their own 
SPIRES system so they can continue to do what they’ve been doing. And it’s 
perfectly legal because the University says you have an account, you have this 
space, you can do whatever you want with it. And most of these systems are 
under two gigabytes, when you consider just the files that they’re involved 
with, and the SPIRES part, which is one megabyte. I mean a one-megabyte 
SPIRES, and then the file system that they need, there’s not a lot of space
taken up. So that’s why I’m trying to get this to work.

**Kiefer:** [01:30:09] Yes. As far as I’m concerned, with just working there for 34 years, I think the only thing—what really kept me going, I think, was serving people, people that needed things in order to get their work done, and being able to interact with them, and also to get their comments or questions, or why can’t we do such-and-such, gave me ideas about things I could do in the future. So I think I always had a list just a mile long of things that I would like to be able to do. And sometimes it took years to get to that point. I remember we had—we have things in SPIRES that are just so neat, like virtual elements, dynamic elements. We have dynamically built databases, dynamically built—I don’t know, it was just like everything that you could think of—phantom structures.

**Guertin:** [01:31:09] Yes.

**Kiefer:** [01:31:10] Phantom structures. That was nice—that was something I thought about, oh, probably in the early ’70s, mid ’70s. But it took me seven years before I had all the ducks lined up where we could put it in. And so it’s things like that. You just have something, a goal somewhere in the future and you’re able to work towards it. So it was always fun, and it was always demanding, and always, just neat people to work with. They were accessible. They were people that were anxious to see.

**Guertin:** [01:31:41] They were enthusiastic.

**Kiefer:** [01:31:42] You could have bugs, and you could cause them lose a day of work and they weren’t mad. That’s what was kind of neat. That they would apologize for getting you in the situation where something you did failed. And so that was a neat atmosphere. So the whole thing about it was just wonderful.

**Guertin:** [01:32:01] Last week I got a phone call from Mike Sullivan at SLAC. He said that somebody updated a record and it was causing a passing error and he didn’t know how to fix it. This was an email I got from him. I responded
back in email. And then we started getting into this email conversation, and I said, this is going too slowly. Just pick up the phone and call me. So he did. It was nine o’clock at night. And within 15 minutes the problem was fixed.

And then we got into the post part of it, which is how did this happen in the first place. Well, it turned out this record was extremely old. And when it was transferred out and brought back in, the characteristics of the database had been changed so that the data that came out didn’t go back in correctly because it was not expected to come—I mean, it was a bad data coming out, and now it came back in and it caused this problem. It couldn’t pass it to an index. So it’s very strange how sometimes when these things live for a long time, and you make changes to it but don’t realize that you’re invalidating the data that’s in it, that you’re causing your own problem. But, you know, he got a 15-minute response and it was fixed. You try to do that with a vendor. You call a vendor about a problem, maybe six months from now, they’ll get back to you with a response about what’s wrong. You can’t wait six months. But that’s what the University’s into now. They have to wait six months for things. It’s—or they don’t come back at all because, you know, in these large systems they’re not willing to institutionalize their general products. They’re not willing to tailor it to Stanford’s needs because they’re selling these things to all kinds of people and we’re just another customer, and they don’t have the time to do that. And that’s something that we did. When Stanford said to us we need to have this, they got it, and they got it quick.

**Kiefer:** [01:34:05] Yes, that’s an interesting part of these commercial systems. They come and they make their changes, and the people that use those systems have to upgrade to keep up with the vendor. At Stanford that takes a lot of work because Stanford has its own modifications to these systems in order to get Stanford stuff done. That means they have to go through all that stuff again. That’s expensive. So every year, or every time there’s a new version of some software coming in, they have to do it again; whereas when we were there, we were always responding to their problems right off the bat.
I should have mentioned when we were talking about people, that there were key people in other groups like WYLBUR. Mike Durket rewrote WYLBUR in C, and that’s what is used in conjunction with Dick’s emulated SPIRES. Jim Nesbitt did a lot of work interfacing with us, with WYLBUR, too. But I was going to talk about Doug Fletcher who, when he came on, maybe 2000, maybe 1999, he did some things in ORVYL, the timesharing system, that really made a difference in how the system worked. And that was the 31-bit addressing, the ability to have larger block sizes. He was just a wonderful person to work with. He was excited to help us out, that kind of thing. He made it fun, and he just enjoyed being there for all that time.

**Guertin:** [01:35:56] So now there’s more than one kind of SPIRES out there. If you go to the UNIX platform and there’s a 24-bit addressable SPIRES, there’s a 31-bit addressable SPIRES, there’s a SPIRES that runs on a big-endian machine, there’s a SPIRES that runs on a little-endian machine. I mean there’s just a whole host of varieties of SPIRES all from the same source simply by just setting flags that say I want to produce this version of it as opposed to that version of it. And all the source codes, still this PL360 code, but the compiler was written to run under ORVYL.

And since the emulator is emulating ORVYL, I can run the compiler on the UNIX platform under emulation and compile PL360 code and get object decks. And the linker was written to run under ORVYL. So I run the linker to link the stuff. And I’m able to produce a new version of SPIRES by compiling PL360 code, or analyzing the BNF, or whatever it is necessary to produce the objects, and then link the objects together to get a new version of SPIRES. And I do that on a regular basis. Whenever something new has to be put into SPIRES, I just go ahead and do it, even though there’s no IBM mainframe around anymore.

**Kiefer:** [01:37:09] So, yes, there—instead of a $10 million dollar computer, you’ve got a little desktop Mac.

**Guertin:** [01:37:20] In fact, I still update all the SPIRES documentation at home. And
the beauty of the documentation is once you’ve got one of these records updated, it usually turns into an explanation in the Explain file. So there might be a record that talks about how the dollar squish function works. And so here’s the description of the dollar squish function. So I go and I edit that and I update the documentation. And then I ask the system to produce me the Explain output, and I input that to the Explain file, and now the Explain file has the latest explanation of the squish function. And so the entire Explain file is built basically from individual records out of these documents, and the documents are in all kinds of different files. There’s the formatting file and the searching file and all the various pieces. All those pieces together make up the entire Explain file of SPIRES. And all of that’s accessible from the web. You can go to the web and ask the system to go and give me an explanation for something in SPIRES and it gets it out of the SPIRES database.

Kiefer: [01:38:28] So finally, what was good about working at Stanford was having Dick as a co-worker.

J. Kiefer: Great.

Kiefer: [01:38:38] Because he did all the things I didn’t want to do and I probably did things he didn’t want to do.

Guertin: [01:38:45] No, everything you did was wonderful.

J. Kiefer: So a lot of you could have gone to work, surely at Apple or any of these other Silicon Valley places, and perhaps made more money than at Stanford. What do you think kept people at Stanford?

Kiefer: [01:38:59] The freedom, for one thing.

Guertin: [01:39:02] Yes. Not just the freedom, it was the intellectual freedom. One of the problems with companies, I’ve found, because I talk to people who work for these companies, and I say, can you do this? And they say, oh, no, the company won’t let me do that. We were on our own. Our bosses weren’t
really our bosses. And they weren’t even our managers. I can remember Jim Moore coming in and saying is there anything going on in SPIRES I should know about? And I’d say, no. He’d say, okay, thank you. And he’d leave. That was it. That was my contact with my manager. All he wanted to know if there was a problem or not. If there wasn’t a problem, just “Dick, keep doing what you’re doing.” You don’t get that in corporations.

The other thing was I think Stanford actually was a benefit to me in the sense that they made my life too easy to live and they contributed to my retirement, so that when I put a little bit into my retirement fund, Stanford put in twice as much more. Most corporations, they maybe put in half of what you put in. But to double what you put in, not—that’s unheard of.


Guertin: [01:40:15] And they paid for my daughter’s college education, and my son’s college education, because they had this tuition grant program for the children of employees. There were a lot of perks—I would call them perks, that made being there wonderful.

Kiefer: [01:40:30] Yes. And the thing that I got—well, when I worked—10 years for how many different companies—there was one, two, three, four before I got to Stanford. So that was 10 years. Each one, something happened about one and a half or two years out where everything changed. In Control Data, the computer you’re working on is obsolete. We’re getting rid of that whole project you’ve been working on. The operating system for the 6000, well, we may pick the one you’re working with, or we’ll—maybe we’ll use this other one that somebody else is working on. So things changed drastically. Like at Hercules Powder Company, they lost a contract in building these big missiles. Well, at the end of that it just made things so different. So even Control Data got a new management and everything changed.

Guertin: [01:41:33] People in and people out.

Kiefer: [01:41:34] Anyway, and then at InterAccess. And then they came and took
Guertin: [01:41:39] Because the boss wasn’t paying for it.

Kiefer: [01:41:42] You could count that within two to four years, everything will be different. You’ll have different management. You’ll have different processes. Whereas Stanford, it was—yes, there were things that kind of got in the way at times but 35 years working on the same program. Can you imagine? I mean what a wonderful thing that was. I don’t think there’s too many people in the world that’s been able to do that in, especially in the business we were in. We were able to see changes that were monumental going on, and we were still able to keep up with it.

J. Kiefer: So did you have much turnover over time?

Kiefer: [01:42:20] The department?

J. Kiefer: Your work group.

Guertin: [01:42:24] There was hardly any turnover in our workgroup until the demise of the mainframe, and then virtually everybody was gone.

Kiefer: [01:42:32] Yes. The systems programmers and the applications programmers, too. When the new systems came on, they found a way of getting rid of them. The older people that knew the business at Stanford were given their pink slips, and the people that were around fewer years, made less money, they were the ones that got the jobs that were available when a project was over. Like, okay, let’s put the Financial Systems together. Okay, we got it. Okay, now all you hundred people that were working on that, you’re fired or you’re laid off. Excuse me. But we got 30 jobs available. You can apply for them. Well, it was like that.

Guertin: [01:43:13] There was a cultural change at the University. They used to be very employee-oriented. And there were virtually no lay-offs. The only persons that ever got thrown out were people who were absolutely incompetent. And the University would say you’re just simply not doing the job, we have
to lay you off. But typically people were in there for life. You could have a job for life. And then the culture changed, and all of a sudden, instead of being an educational institution, it became a commercial operation and started to work like a commercial operation, where the bottom line seemed to make a difference, even though it was all Chinese money, one department paying another department. It was all money being spent within the University itself.

But all these construction projects at the University began to put an impact on the endowment and a bunch of other things, and now they became interested in what was going on with the bottom line, even though they’re a nonprofit organization. They weren’t simply not making a profit, they weren’t making enough to sustain what they were doing, in my opinion. And now they had to get rid of people.

**Kiefer:** [01:44:22] I know what took away my fun and enjoyment was, instead of working on things that people knew they needed and were asking for, I was working on things that I thought they would need if they did things right. Some things they used, some things they didn’t, and I was just trying to figure out what’s going to keep me busy now. It was just a whole different world, so the last three or four years I think were not as enjoyable.

**J. Kiefer:** When do you think this cultural shift occurred that you were just describing, Dick?

**Guertin:** [01:45:00] With the decision to stop making our own stuff.

**J. Kiefer:** And again, that would be around the mid ‘90s.

**Guertin:** [01:45:07] Yes.

**J. Kiefer:** Okay. Anything else you want to add?

**Guertin:** [01:45:13] Well, the University has recently basically admitted that they didn’t save any money by doing this. They were anticipating that all these vendor supplied products were going to reduce their cost. It turns out that that’s
been just the reverse. It’s increased their cost because they have to pay the salaries of these people who work for these companies who come in as on-site contractors to get the job done. And it’s costing the University more and more and more. Remember when you mentioned virtual elements?


Guertin: [01:45:48] There’s a project that takes SPIRES data and puts it up in what’s called EDW, the electronic data warehouse. And that’s nothing more than a reporting function now, where you get the data there and then you output it. But it’s all table structured. It’s data in table form. Well, you can’t put virtual elements in a table database. No such thing. It actually physically has to exist. And an example of what a virtual element would be is I have a birth date stored in your record. A virtual element would be what your age is. And when I ask the system what’s your age, and it gives me the value of the age element, it computes the age instantly and gives you the answer as though it were stored in the database when, in fact, it’s not. It’s computed. Well, they had to take virtual elements and turn them into columns of data that would be good for one day, and then the next day they had to replenish them all and re-compute them all over.

Kiefer: [01:46:45] Are you kidding me?

Guertin: [01:46:46] No, I’m not kidding. In order to get virtual data into the database, they had to have columns of it, and it would only be good for one day. So if it changed during the day, you wouldn’t know about it until the next day when it actually would get updated by them going through the whole thing and replenishing all of its values.

Kiefer: [01:47:03] Yes, I can believe it.

J. Kiefer: Okay. Well, Dick Guertin and Bill Kiefer, thank you very much—very much for your time.

[End of Interview with Richard Guertin and William Kiefer]