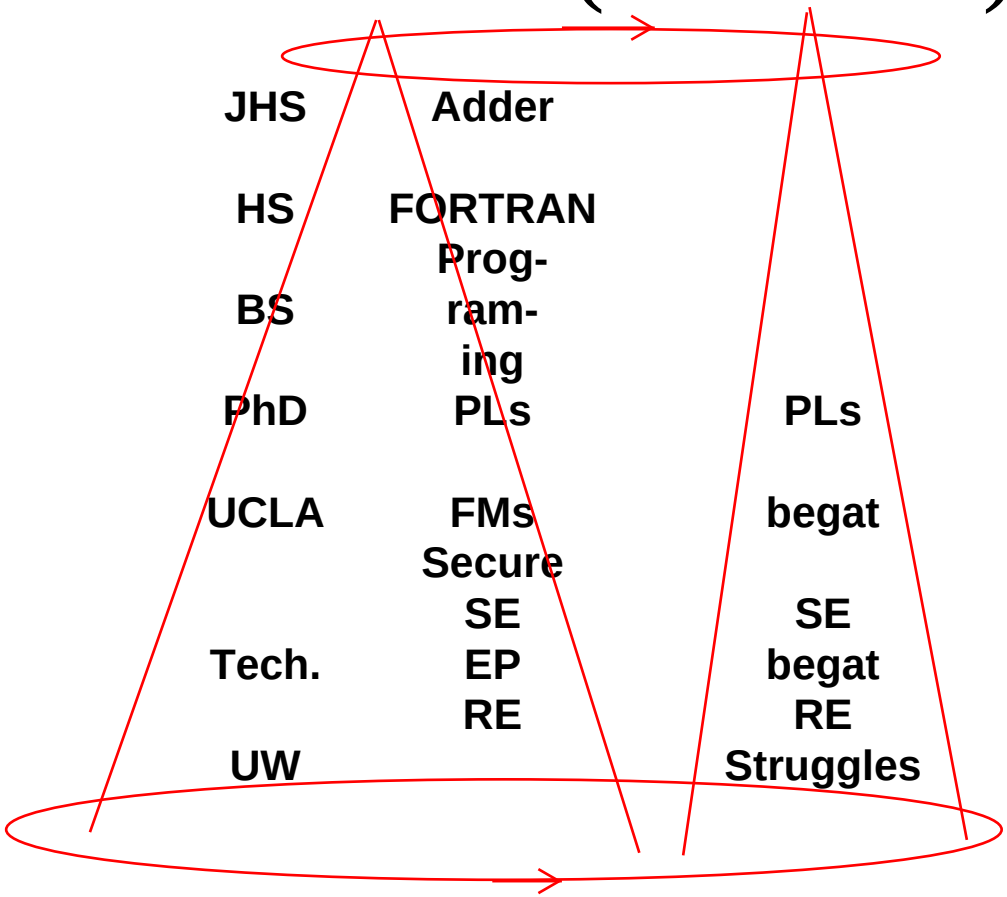
The Prehistory and History of RE as Seen by Me

Daniel M. Berry
University of Waterloo, Canada
dberry@uwaterloo.ca

Outline (Pictorial)



Vocabulary

CS = Computer Science

CBS = Computer-Based System

SW = Software

PL = Programming Language

FM = Formal Method

SE = Software Engineering

EP = Electronic Publishing

RE = Requirements Engineering

We

In the following, at any time, …

“We” = all the people in whatever field I was in at the time.

So it is context dependent.

My Start in Computing in 1960s

In the beginning, I

- **built a relay computer, an adder, in 1962, age 14, for a junior high school science fair,**
- **learned to program in FORTRAN in the summer of 1965, age 17, at an NSF SSTP at IIT in Chicago (Ed Reingold was my dorm counselor!),**


1960s

My Start, Con'd

- **wrote my first real-life application, Operation Shadchan, a party 1-1 matching program based on the questionnaire of Operation Match, a 1-*n* dating program, in the Spring of 1966, age 17, for my synagogue's youth group's annual party,**
- **studied pure math from 1966–1969, at RPI, an engineering school, to get a B.S., not a B.E. as most of my class mates,**

My Start, Con'd

- **programmed statistical and curve-fitting SW for the Chemistry Dept. at RPI, to make spending money (I wrote FORTRAN from formulae they gave me.),**
- **joined ACM in 1967 (member # 10*****), and**
- **programmed payroll applications in RPG for a service bureau in Troy, NY (home of RPI) in the Summer of 1969, to make money to go to grad school.**

SOTP BIAFIUIW

Through all this, I did seat-of-the-pants build-it-and-fix-it-until-it-works (SOTP BIAFIUIW) SW development, …

simultaneous RE, design, and coding, …

not really understanding the distinction between RE, design, and coding, …

SOTP BIAFIUIW, Cont'd

thinking that all of it were just parts of programming, …

probably like a whole lot of programmers, even professionals, did.

Grad School

Later, I

- **started grad school at Brown in 1969 as a pure Math PhD student (Never mind an MS; that's for people who want to *work* for a living.),**

- **took Measure Theory from Herbert Federer, who literally wrote the textbook, and discovered that I had promoted myself to my level of incompetence (the Peter Principle) in math,**

Grad School, Cont'd

- **did a lateral transformation to take computer science courses in the Applied Math department down the street,**

- **fell in love with PLs when I took Peter Wegner's course, PLs, Information Structures, and Machine Organization (PLISMO), from the book he wrote from his PhD thesis, and**

Grad School, Cont'd

- **ended up getting my PhD in 1973 from Peter on**

  the design of and the formal specification of Oregano, an improvement over Algol 68 and over Basel; …

  it was designed to be more orthogonal than either by keeping the architecture of its implementation firmly in mind; …

  that architecture became the basis for its operational VDL formal specification.

CS Journals in Early 1970s

At that time, there were only 3 journals in CS,

***CACM*, monthly,**
***JACM*, quarterly, and**
***CR*, quarterly.**

So, I read at least the abstract of *every* paper published in CS for a few years.

CS People in Early 1970s

Also, the number of people in CS in the early 1970s was small enough that any person could know just about everybody in his or her field and many in other fields.

And most of the pioneers were still alive.

So, I met just about *everybody*, …

including the authors of *IEEE TSE* January 1977, at conferences or even in LA while at UCLA.

Assistant Professor at UCLA

I started as an assistant professor in 1972 at UCLA, where the ARPAnet that later became the Internet, was happening.

I started off in the field of PLs.

SIGPLAN was the *biggest* SIG of the ACM at the time.

We all knew how difficult it was to write correct SW that does what its client wants.


1970s

PL Research in Early 1970s

The overarching concern of PL research in the early 1970s was:

- **to design a PL in which people would write correct and good SW, and**
- **to try to design a PL in which it was difficult, even impossible, to write bad SW**

Mission Impossible

But of course, that is impossible

We realized that you could easily write *really* atrocious SW in even the most structured PL …

At one meeting, someone (I forgot whom) came up to the blackboard & showed us the following goto-free structured program:

Atrocious SW

```
for i from 1 to 4 do
    case i in
        1: S1,
        2: S2,
        3: S3
        out S4
    esac
od
```

which, of course, is equivalent to

S1; S2; S3; S4 ☹

My PL Research

My own PL research was in

- **making PLs more orthogonal,**
- **adding features to PLs in an orthogonal way**
- **operational formal semantics of PLs and their features.**

My PL Research, Cont'd

I ended up being involved with the Algol 68 committee.

My PL Research, Cont'd

I supervised research

- **on new PL features integrated into existing orthogonal PLs, e.g., Algol 68, in the cleanest, orthogonal way, with few or no leaky abstractions,**
- **finding optimal implementations for these features, e.g., for garbage collection, and**
- **formal semantics of the features or of PLs, e.g. of Algol 68.**

Early Signs of RE Thinking

Note my own RE orientation of trying to fit a new feature into the existing language in the cleanest way, exploring it thoroughly before beginning to implement it.

SARA

All this time at UCLA, I was a member of Jerry Estrin's SARA group.

SARA was a multi-notation system design language, a competitor of SA and PSL/PSA, and …

a precursor of UML.

SARA, Cont'd

SARA was implemented with textual input but line-printer graphic display of models so that it could be used over ARPAnet.

SARA provided analysis tools to verify well-formedness and mutual consistency of models, to run simulations, etc., like PSA for PSL.

SARA, Cont'd

Several of my PhD students built pieces of, analyzed parts of, or applied SARA for their theses.

It was in connection to this research that I met some of the authors of the papers of the papers in the January 1977 issue of *TSE*, …

e.g., Doug Ross, John Brackett, Dan Teichroew, and Mack Alford.

SARA, Cont'd

The irony of all this SARA work is that …

while other things I did feel to me as having used what became RE thinking or having facilitated my realization of the importance of RE and its activities, …

this SARA work did nothing of the sort.

SARA, Cont'd

In fact, I will admit to being *totally* surprised that the organizers of this conference thought that the collection of papers in the January 1977 *TSE* marked the birth of RE.

To me, the work they did is more technical and notational, than attacking the fundamentals of RE, but that's my viewpoint.

RE: Only Three Journals in CS

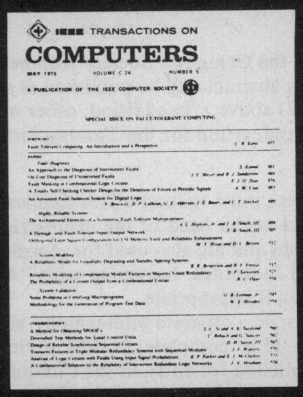
Look at the advertisement that appeared in the January 1977 *TSE* …

about all three IEEE computer journals!

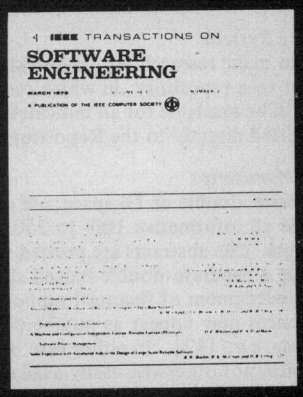
if YOUR COMPUTER ENGINEERING LIBRARY DOESN'T SUBSCRIBE TO all 3

it's not complete.

Tutorial and survey papers covering the entire range of hardware and software design and application, with each issue focussing on a single major theme. Regular features include conference and workshop reports, book reviews, calls for papers, and the Computer Society Repository—a unique microfiche and hard-copy collection of over 3000 technical papers on computer research. (monthly)



The oldest and most respected scientific journal in the field of computer engineering, features state-of-the-art research and design papers in all areas of computation and information processing. (monthly)



The newest archival journal published by the IEEE Computer Society — and the only scientific journal devoted to this vital area. Features archival research papers on all aspects of the specification, development, management, test, maintenance, and documentation of computer software. (quarterly)

Published by the Computer Society of the Institute of Electrical and Electronics Engineers

IEEE COMPUTER SOCIETY® INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS®

IEEE COMPUTER SOCIETY PUBLICATIONS OFFICE: 5855 NAPLES PLAZA, LONG BEACH, CALIFORNIA 90803

E-mail

In 1977, I started using e-mail as a replacement for the difficult-to-use telephone to connect with most of my acquaintances, who were CSers!

In 1980s, I started a campaign to convince my non-CS friends and my family to do the same.

Foment in PL Field in Mid ’70s

In the mean time, in the PL field, we realized that the key to getting better SW was *not* to improve PLs, *but* to improve the *process* of SW development.

NATO meetings

The NATO meetings in the late 1960s had already suggested in response to the SW crisis (bad and badder and badderer SW is being produced as the need for SW is growing) that *maybe*

- **we should be systematic and science based and**
- **we should be *engineering* our SW,**

just like bridge builders engineer their bridges based on the laws of physics.

Birth of SE field

Thus, was born the field of SE, initially populated with PL people who realized

- **that the PL used in programming has little or no effect on the quality of the SW programmed with it, and**

- **that programmers' behavior had a far bigger impact on the quality of SW they produced than the PLs they used.**

Switching to SE

So I, like a whole bunch of other PL people, ended up switching in the mid to late 1970s to SE.

We tried during the 1970s and 1980s (when ICSE met only every 18 months) to find methods, possibly assisted by math, to develop correct SW meeting its client's needs.

The study of PLs morphed to the study of methods, and …

formal semantics for PLs morphed to FMs.

My Sojourn into Security

In the early 1980s, as a result of superivising several people doing formal methods, and in particular Richard Kemmerer who did (1) a formal specification of the kernel of the UCLA secure UNIX and (2) a formal verification of that the kernel met the specification of security, …

I got involved in the security community.


1980s

Security, Cont'd

I consulted for the Formal Development Method (FDM) group of SDC that was working on secure operating systems, in particular Blacker.

I ended up publishing a paper in *IEEE TSE* showing how the theorems that the group's verifer proved about an Ina Jo formal specification of a system were sufficient to prove that the system, if implemented as specified, would meet the specified criteria.

Security, Cont'd

From all this work and from its community that included such people as Peter Neumann, I learned a lesson that goes right to the essence of RE:

There is no way to add security to any CBS after it is built; the desired security must be *required from the beginning* so that security considerations permeate the entire development lifecycle.

My Sojourn into EP

While I was doing this SE and FM stuff, I made a parallel diversion in the mid 1980s through mid 1990s into Electronic Publishing (EP).

I got to design and build SW for multi-lingual and multi-directional word processing.

I tried to find the most orthogonal way to integrate the new features, using the least leaky user abstractions.

EP SW

It was all based on troff (piped architecture with a separate program for the feature bundle for one class of document artifact, e.g., table, formula, line drawing, etc.).

This way, I could add a new feature or artifact by building a relatively independent program for the feature or artifact and stick it into the pipe in the right place.

RE Orientation Even in EP

Note the RE orientation here

- **in the concern for orthogonality and**
- **in finding the least leaky user abstractions.**

These make the new features easier to use because they suffer no surprising exceptions.

I Left the Field

I left the EP field when

- **EP's leaders decreed that all future papers in the area had to be written in LaTeX, even papers about additions to troff.**

  (There was no way I could keep the rule of using the SW a paper is about, to produce the camera ready copy of the paper in the venue's traditional format.)

Left the Field, Cont'd

- **The Unicode consortium ignored my command-heavy, but simple commands and leak-free abstractions for bidi word processing to …**

  develop their standard, which uses defaults to avoid commands in the normal case, but has invisible commands for the exceptional cases, the commands requiring an incredibly complex algorithm that is still being corrected, and forming very leaky abstractions.

Beginning My Move to RE

During this time, in 1981, I published a paper with Orna Berry about how I managed to do the best job ever in specifying software that she had to write, in a domain that I knew nothing about.

I agreed to do this job *only* because I was married to her at the time!

Beginning My Move, Cont'd

In retrospect, I consider this to be my first RE paper.

It's certainly one of the very earliest on the elicitation aspect of RE.

Ignorance Hiding

She had to write some programs that played statistical games with experimental data.

I got my lowest Math grade in the undergrad Probability and Statistics class, a B, (it ruined my perfect Math GPA.) because I had *no* intuition for probability.

So, I was ignorant in the statistics domain.

Ignorance Hiding, Cont'd

To be able to hide my ignorance so I could work effectively with the requirements as she expressed them to me, …

I made the experimental data an ADT, with each magic function that I did not understand, e.g., standard deviation or standard error, being a method of the ADT. I knew that the client understood what they mean and how to implement them. So I worked with this ADT with its methods taken as primitive.

Ignorance Hiding, Cont'd

I thought and claimed in this paper that this ignorance hiding technique was the basis of the success …

as well as my ability to nudge the client to give information

and to do strong-type checking on natural language sentences.
(Using the same verb with different numbers and kinds of direct objects in different sentences is a type error.)

Importance of Ignorance

By 1994, I figured out that the reason for the success was not the ignorance hiding, but the very ignorance!

Importance of Ignorance, Cont'd

So in 1994, I published "The Importance of Ignorance in RE" claiming that every RE team for a CBS requires along with domain (of the CBS) experts at least one smart ignoramus of the domain, who will

- **provide out-of-the-box thinking that leads to creative ideas, and**
- **ask questions that expose tacit assumptions.**

Empirical Validation

In 2013–2015, my PhD student, Ali Niknafs, conducted controlled experiments to empirically validate that

for the task of brainstorming for requirement ideas, …

among 3-person teams consisting of only computer scientists or software engineers, …

Empirical Validation, Cont'd

the teams with *one or two members ignorant* in the domain …

generated *more and better requirement ideas* …

than teams consisting of …

- ***only ignorants* of the domain or …**
- ***only awares* of the domain.**

The Birth of the RE Field

After a while, in the mid 1980s, a subset of the SE people began to notice that SE methods and FMs do not really solve the problem of ensuring the production of quality SW.

- **They don’t scale well, particularly FMs: For some funny reason, FM people did not use FMs when building tools to help do FMs.**

Birth of RE Field, Cont'd

- **A method works well only the first time on any CBS. After that, when the CBS must be updated, e.g., for requirements changes, the artifacts produced by the method must be updated to be consistent with the changes.**

Birth of RE Field, Cont'd

- **This updating is difficult because it is akin to lying perfectly consistently, which is very hard to do.**
- **The lie is making *all* artifacts appear as if they were produced during an application of the method to produce the current version from scratch!**

- **Change is relentless, and therefore, lying is perennial!**

Change is Relentless

Why is change in a CBS relentless? Because of changes in the CBS's requirements:

- **We did not understand the CBS's requirements to begin with.**
- **We made a mistake in expressing what we understood.**
- **We deployed the CBS into the real world, giving rise to the Lehman feedback loop that changes the CBS's own requirements!**

A Realization

Then, a subset of the SE field came to the realization that the real problem plaguing CBS development was that we did not understand the requirements of the CBS we are building.

A Realization, Cont'd

Brooks, in 1975, had said it well:

"The hardest single part of building a software system is deciding precisely what to build…. No other part of the work so cripples the resulting system if it is done wrong. No other part is more difficult to rectify later."


1990s

A Realization, Cont'd

This subset of the SE folk formed the RE field,

1. **by piggybacking on the nearly annual International Workshop on Software Specification and Design in the mid to late 1980s and early 1990s,**
2. **from 1993, in two alternating conferences, ISRE and ICRE, that later merged into one (RE),**
3. **from 1994, in an annual working conference, REFSQ,**
4. **from 1996, in a flagship journal, REJ.**


2000s

RE Now

Even within RE, there has been a lot of concern for …

technology: notation, methods, tools, FMs, etc. …

as well as for …

the human side: elicitation, creativity, emotions, politics, psychology, sociology, etc.

Both Are Important

Both technology and the human side are important.

Both need to be studied thoroughly and should be the subject of research.

A Continuous Struggle

However, I find a continuous struggle within RE that mirrors the decades-long struggle that created RE from PLs.

The struggle is that:

- **As CSers, we *love* technology. We like to think that technology can solve *all* problems.**
- **But, we discover that it doesn't, sometimes to our surprise.**

Struggle Within PL Field

For example, we thought that designing the perfect PL would improve SW development.

They did, …

but not anywhere nearly enough.

PL Field Struggle, Cont'd

The problem was that the process of making the SW has a bigger impact than the PL on the eventual quality of the SW.

So we invented SE to focus on the actual process of developing SW.

Struggle Within SE Field

We thought that methods and tools applied to the actual programming would improve SW development.

They did, …

but not anywhere nearly enough.

SE Field Struggle, Cont'd

The problem was that following the best methods was useless if we did not know what to build, and …

the available methods had no effect on getting that knowledge.

So we invented RE to focus on the process of deciding what to build.

Struggle Within RE Field

It's always a tension between technology, methods, tools, etc.

and a human thing, e.g. how do we *humans* develop, how do we *humans* find out how to build.

As in SE, both are essential, …

but within the RE field, this tension continues.

Struggle Over Technology

You find RE researchers developing techniques, methods, and tools, i.e., technology.

Often this technology is being developed to assist in doing a task that people do not like to do, e.g., tracing.

Struggle Over Technology, Cont'd

The main reason a person doesn't like to do such a task, is that the beneficiary of the task is someone else down stream, and …

the person who has the knowledge to do the task gains nothing from doing the task [Arkley & Riddle] other than a pain in the tukhis, …

mainly because he or she already has the knowledge.

I.e., there is no incentive to do the task, *even* if there is assistive technology.

Struggle Over Technology, Cont'd

But, if people have no incentive to apply the technology,

- **in the interest of being more agile,**
- **because the technology is too cumbersome, or**
- **they don't even see the value of the doing what the technology helps them do,**

then, the technology is not going to be applied, no matter how good it is.

Struggle Over Technology, Cont'd

Unfortunately, many technology developers are failing to consider this human aspect.

(Note that this is all independent of NIH (not invented here).)

Another Struggle

RE in practice involves a lot of talking with people and asking them questions.

Yet, you find an attitude that just talking with people and asking them questions isn't sexy enough to be the subject of good RE research.

RE is Very Inclusive

To me, RE includes *anything*, I repeat, *anything*, that can be shown to …

improve the process by which we determine the requirements of a CBS and …

that leads, downstream, to a better CBS …

as a result of what is done to determine the CBS's requirements.

Very Inclusive, Cont'd

I don't care *what* the *anything* is —

technology, psychology, sociology, management, role playing, fun and games, and even feeding your client milk and cookies before eliciting requirements

— so long as it has an empirically demonstrable positive effect on the requirements gathering and on the eventual CBS!


2010s

RE Everywhere

I see RE problems and lessons …

walking down the street, …

everywhere!

(The ambiguity of who is walking is intended.)

E.g., in house building, house remodeling, NY bagels, a synagogue's kitchen, the atrium of UW's Davis Centre, Waterloo Region's light rail, U.S. income tax instruction booklet, and even Biblical passages.

In Today's World!

In today's world, everything, especially SW development, is multi-disciplinary.

At Google, requirements elicitation teams have people from multiple disciplines, experts in the CBS's domain, engineers, lawyers, psychologists, sociologists, HCI experts, UX experts, and even SW engineers, ☺ …

- **to gather what is needed for the CBS, and**
- **to gather *new, out of the box* ieas.**

RE Struggle 1

It really rankles me when I see people younger than I being more conservative and protective about the boundaries of their field.

RE Struggle 1, Cont'd

I am talking about reviewers for conferences and journals who reject papers because

- **"It's too much psychology | sociology | management | games."**
- **"It's really an HCI paper." (and the HCI reviewer says "It's really an RE paper.")**

RE Struggle 1, Cont'd

- **"It's too much of a story." (about a case study of the successful application of some technology)**

- **"But the method did not work." (about a case study showing that a believed technology didn't work in at least one situation)**

RE Struggle 2

Why do we insist that a tools for doing an RE task on natural language documents have high precision when the task is one in which humans are not good?

RE Struggle 2, Cont'd

If we are not good at the task and need a tool's help to do it, then it seems clear enough that recall is the key criterion for the tool's success, not precision, …

particularly when it takes a human an order of magnitude longer time to find a good answer than it does to reject a tool-offered nonsense answer.

RE Struggle 2, Cont'd

It seems to me that in borrowing Information Retrieval's methods to build these natural-language processing tools, we have adopted their measures without considering the *requirements* for our tools.

We are failing to do RE for our own RE tools!

RE Struggle 3

I see that many a talk or paper on a cognitive RE process ends with a promise to build a tool to assist human requirements analysts in carrying out the process.

Yet these tools never get built.

I am *not* complaining about the fact that they don't get built. I don't think anyone really expected any such tool would be built.

RE Struggle 3, Cont'd

I *am* complaining about our need to promise to build such a tool.

It's as if the promise is admitting that we do not feel comfortable doing this research about soft cognitive stuff. So, to justify doing this research, we say that we will build a tool.

You see, all this research is not just about soft stuff; it's going to build a good respectable tool!

RE Struggle 3, Cont'd

The reality however is that this cognitive stuff is *fundamental* to understanding RE and to doing it well; …

So doing research on it is the *right thing* to do!

RE Struggle 4

I see a lot of work on RE for security.

Only rarely does this work ever cite the work done in the early 1980s.

Recall how I learned a fundamental lesson of RE from this work.

Security, Cont'd

From all this work and from its community that included such people as Peter Neumann, I learned a lesson that goes right to the essence of RE:

There is no way to add security to any CBS after it is built; the desired security must be *required from the beginning* so that security considerations permeate the entire development lifecycle.

RE Struggle 4, Cont'd

We in RE need to be looking at that old work for

- **what it has already solved and**
- **insights that are relevant today.**

RE Struggle 4, Cont'd

Probably the best place to start is with the Oakland Symposium on Security.

Its current instantiaion has a Web site, https://www.ieee-security.org/TC/SP2017/

Its past proceedings can be found at http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000646
Security and Privacy, IEEE Symposium on
Click on "More History"

I Just Don't Understand

How is self-adaptive SW different from …

ordinary well-designed, robust SW, which is able to field *any* input and has responses for each already programmed into the SW, …

especially since adaptations in self-adaptive SW have to already be programmed in for them to be invoked automatically by the SW?

Don't Understand, Cont'd

The RE problem for both is the same:

- **anticipating all situations that need (adaptation = unusual responses), and**
- **anticipating the correct (adaptation = responses) for them.**

Conclusion

I have been in computing in one way or another since 1963 and have been programming since 1965.

While I have been in a whole gamut of CS fields and have picked up understandings of other CS fields, …

often, by supervising a graduate student who picked his or her own topic and taught it to me.

Conclusion, Cont'd

I see now that I have always been heading towards my current field, RE, …

because, in retrospect, no matter what X I was building, the hardest problem that demanded most of my attention was "What is really required of X?", i.e., "What are X's requirements?"

Lessons I Have Learned:

- **importance of talking with customers and users,**
- **importance of domain ignorance in RE,**
- **security, robustness, user interfaces, etc. have to be *required* into a CBS from the beginning,**
- **importance of knowing what the CBS is to do, as much and as early as possible,**
- **RE is everywhere, and**
- **every RE rule has exceptions.**

Take Away

My main take away message is very simple:

The RE field includes whatever helps do RE in real life.

And I intentionally left off “for CBSs” in the previous sentence.

Acknowledgements

Thanks to Derek Rayside and Vicky Sakhnini for their comments on a dry run of this talk.

Appendix

My PhD Students

P **Meyers, Barbara F. "Design of a Language for an Associative Processor" 1975 (UCLA)**

P **Schwartz, Richard L. "An Axiomatic Definition of ALGOL 68" 1978 (UCLA)**

P **Erlinger, Michael "Analysis of Retention Storage Management for Generalized Block Structure Languages" 1979 (UCLA)**

R **Kemmerer, Richard A. "Verification of the UCLA Security Kernel: Abstract Model, Mapping, Theorem Generation, and Proof" Co-advised, 1979 (UCLA)**

R **Leveson, Nancy G. “Applying Abstract Data Type Methodology to Data Base Systems” Co-advised, 1980 (UCLA)**

P **Misherghi, Shakib H. “An Investigation of The Architectural Requirements of SIMULA 67” 1980 (UCLA)**

S Penedo, Maria Heloisa “The Use of a Module Interconnection Specification Capability in the synthesis of Reliable Software Systems” Co-advised, 1980 (UCLA)

P **Yemini, Shaula “The Replacement Model for Modular, Verifiable Exception Handling” 1980 (UCLA)**

R **Paolini, Paolo “Abstract Data Types and Data Bases” 1981 (UCLA)**

R **Schwabe, Daniel “Formal Techniques for the Specification and Verification of Protocols” 1981 (UCLA)**

P **Mazaher, Shahrzade “An Approach to Compiler Correctness” 1981 (UCLA)**

R **Burstin, Meir D. “Requirements Analysis of Large Software Systems” Co-advised, 1985 (Tel Aviv Univ., IL)**

S **Worley, Duane R. “A Methodology, Specification Language, and Automated Support Environment for Computer Aided Design Systems” 1986 (UCLA)**

S **Krell, Eduardo A. “A SARA-based Ada Programming Support Environment” 1986 (UCLA)**

S **Lor, Edward Kar-Wing “A Requirement-Driven System Design Environment” 1988 (UCLA)**

R **Maarek, Yoëlle “Using Structural Information for Managing Very Large Software Systems” 1989 (Technion)**

Schwarz, Avner "Representing and Solving the Automated Building Design (ABD) Problem" Co-advised, 1992 (Technion)

R **Goldin, Leah "A Method for Aiding Requirements Analysts in Requirements Elicitation for Large Software Systems" 1994 (Technion)**

Jehuda, Jair "A Top-Layer Design Approach to Complex Real-Time Software" 1998 (Technion)

R **Breitman, Karin "Evolução de Cenários (Evolution of Scenarios)" Co-advised, 2000 (PUC Rio de Janeiro, BR)**

R **Kamsties, Erik “Surfacing Ambiguity in Natural Language Requirements” Co-advised, 2001 (Universität Kaiserslautern, DE)**

R **Ramos, Isabel “Aplicações das Tecnologias de Informação que Suportam as Dimensões Estrutural, Social, Política, Simbólica do Trabalho” Co-advised, 2001 (Universidade do Minho, Guimarães, PT)**

R **Svetinović, Davor “Increasing the Semantic Similarity of Object-Oriented Domain Models by Performing Behavioral Analysis First” Co-advised, 2006 (Waterloo)**

R **Tjong, Sri Fatima "Avoiding Ambiguity in Requirements Specifications" Externally Co-advised, 2008 (University of Nottingham Malaysia Campus, MY)**

R **Niknafs, Ali "The Impact of Domain Knowledge on the Effectiveness of Requirements Engineering Activities" 2014 (Waterloo)**

R **Ribeiro, Cristina "The Prevalence and Impact of Persistent Ambiguity in Software Requirements Specification Documents" 2016 (Waterloo)**

External PhD Students

R **Gonzales, Regina "Capturing Requirements by Formalizing and Integrating Stakeholder Conceptual Models" 2000, New Mexico State Univ., USA**

R **Sobczak, Andrzej "Techniques for Ranking Priorities of Initial Requirements for Public Sector Management Information Systems Based on Strategic Changes" 2003, Warsaw School of Economy, PL**

R **Mauger, Cyril “Méthode de Définition des Exigences d’un Produit Intégrant ses Services Appliquée aux Bâtiments Publics” 2015, Arts et Métiers ParisTech, FR**

My MS Students

P **Yueh, Betty "Contour Model Display Processor" 1975 (UCLA)**

P **Burgess, Henry W. "An Abstract Data Type Extension to a Typeless Language" 1975 (UCLA)**

P **Rempe, Barbara "Structured Programming of a Compiler for a Structured Language" 1975 (UCLA)**

P **Walters, Linda K. "A Method for the Comparative Evaluation of Computer Architecture" 1975 (UCLA)**

P **Riggins, M. Christian “A Definition of Run-Time Error Handling in ALGOL 68” 1975 (UCLA)**

P **Allen, Steven James “The Implementation of String Manipulation in Strimula 76” 1976 (UCLA)**

P **Kaplan, Ronald E. “The Strimula 76 System” 1976 (UCLA)**

S **Linden, Nancy M. “A Software Development Processor: A Tool for Program Design” 1976 (UCLA)**

P **Schwartz, Richard L. “Parallel Compilation: A Design and its Application to SIMULA 67” 1976 (UCLA)**

F **Eggert, Paul R. “A Constructive Definition of Vienna Objects” 1977 (UCLA)**

P **Hethely, Attila “Structured Documentation for On-line Digital Systems” 1977 (UCLA)**

Kaufman, Lawrence J. “Another Capability Based Computer” 1977 (UCLA)

F **Takemura, Joan E. “Proof of Correctness of Implementations of Pointers” 1977 (UCLA)**

P **Omi, Bert Y. “Implementation Techniques for a High Level Microprogramming Language” 1977 (UCLA)**

P **Chan, Francis Yiu-Tung “Type Checking and Type Breaching” 1978 (UCLA)**

P **Campbell, Douglas C. “An Almost SLR(1) Grammar with Semantics for STRIMULA ’76” 1978 (UCLA)**

S **Mujica, Sergio T. “Data Flow Languages and Interpreters” 1978 (UCLA)**

P **Pugh, Eric “Forth: A Redesign and Implementation” 1978 (UCLA)**

F **Smallberg, David “Automatic Verification Condition Generation Using Intermittent Assertions” 1978 (UCLA)**

F **Beyschlag, Ulf “An Euler Style Definition of Simula 67” 1981 (UCLA)**

W **Dempsey, John “The Design, Development, and Maintenance System” 1983 (UCLA)**

E **Buchman, Cary “DITROFF/FFORTID, An Adaptation of UNIX’s DITROFF for Formatting Bi-Directional Text” 1983 (UCLA)**

P **Krell, Eduardo A. “An ADA Translator for the Syntax Directed Editor” 1983 (UCLA)**

P **Eterovic, Yadran “Porting a Unix Version 6 Algol 60 Interpreter to Unix 4.1 BSD” 1985 (UCLA)**

E **Fuller, David A. “A Ditroff Device Driver for the Line-Printer and the Diablo” 1984 (UCLA)**

P **Fung, Antony “The Architecture of A Forth Machine” 1985 (UCLA)**

Nomicos, Sylvana Garlepp “An Assessment of a Software Quality Metrics Model for Distributed Systems and its Application to the Evaluation of the LOCUS Distributed System” 1985 (UCLA)

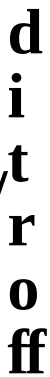
E **Ip, Chok-Ho "CWPR, A Chinese-Japanese Word Processor for Ditroff" 1985 (UCLA)**

E **Fornaciari, William P. "An Outline Editor" 1986 (UCLA)**

F **Holtsberg, Steven "Ina Jo Axioms for Ada's Data Types" 1986 (UCLA)**

E **Takata, Kris "Indx, A Semi-Automatic Indexing Program" 1987 (UCLA)**

R **Aguilera, Christine "Finding Abstractions in Problem Descriptions Using findphrases" 1987 (UCLA)**

E **Becker, Zeev “ditroff/ffortid/  , An Adaptation**

of the UNIX ditroff for Formatting Tri-Directional Text” 1988 (Technion)

E **Habusha, Uri “vi.iv, a Bidirectional Version of the vi Full-Screen Editor” 1989 (Technion)**

E **Allon, Gil “Minix.Xinim, a Bidirectional Version of Minix, a UNIX variant” 1989 (Technion)**

E **Wolfman, Tony “flo—A Language for Typesetting Flowcharts” 1989 (Technion)**

E **Yanai, Shimon “An Environment for Translating METAFONT to PostScript” 1989 (Technion)**

P **Erez, Ruth “A Contour Model Displaying Interpreter” 1990 (Technion)**

E **Srouji, Johny “Bi-Directional Formatting with Arabic and Farsi” 1990 (Technion)**

E **Shpilberg, Faina "WD-pic, a WYSIWYG, Direct-Manipulation pic" 1997 (Technion)**

W **Hornreich, Harry "A Case Study of Software Reengineering" 1997 (Technion)**

E **Ravid, Alon "A Method for Extracting and Stating Software Requirements that a User Interface Prototype Contains" 1999 (Technion)**

E **Denger, Christian "High Quality Requirements Specifications for Embedded Systems through Authoring Rules and Language Patterns" Co-advised, 2002 (Universität Kaiserslautern)**

R **Ou, Lihua "WD-pic, A New Paradigm for Picture Drawing Programs and its Development as a Case Study of the Use of its User's Manual as its Specification" 2002 (Waterloo)**

R **Fainchtein, Igor "Requirements Specification for a Large-Scale Telephony-Based Natural Language Speech Recognition System" 2002 (Waterloo)**

R **Kwan, Irwin "On the Maintenance Costs of Formal Software Requirements Specifications Written in the Software Cost Reduction and in the Real-Time Unified Modeling Language Notations" Co-advised, 2005 (Waterloo)**

W **Chen, Hsing-Yu "Analysis of Software Configuration Management" 2007 (Waterloo)**

W **Yu, Colin "Field Based Development: Case Studies of IBM Product Development" 2007 (Waterloo)**

McDonald, Keith “Combining Processor-Sharing and First-Come-First-Served Queueing Disciplines Using Estimated Job Size” Co-advised, 2007 (Waterloo)

***W* So, Joel “Autonomous Dynamic Workflow: Explicating the Problem Space and Designing a Viable Solution” Co-advised, 2007 (Waterloo)**

***W* Chodos, David “Creating a Web-based Statistical Tool” Co-advised, 2007 (Waterloo)**

E **Mohsen, Shahab "The Problem of Stretching in Persian Calligraphy and a New Type 3 PostScript Nastaliq Font" 2009 (Waterloo)**

R **Weber, Janna-Lynn "Privacy and Security Attitudes, Beliefs and Behaviours: Informing Future Tool Design" Co-advised, 2010 (Waterloo)**

R **Isaacs, Daniel "Developers Like Requirements, Project Managers Don't" 2010 (Waterloo)**

R **Mehrotra, Gaurav "Role of Domain Ignorance in Software Development" 2011 (Waterloo)**

R **Mak, Andrew "Agile Requirements: A Case Study of the Agile Practices in the OSGi Applications Tools Development Team" 2011 (Waterloo)**

R **Werner, Colin "An Industrial Case Study of a Very Large Organization" 2011 (Waterloo)**

R **Ellis, Keith "Quantifying the Impact of Requirements Definition and Management Process Maturity on Project Outcome in Business Application Development" Co-advised, 2011 (Lancaster University, UK)**

R **Dembla, Shivam "The Effect of Several Tradeoffs in the Implementation of Large Displays on the Performance of the Users of the Displays" 2015 (Waterloo)**

R **Lan, Xiao Ye "An Experimental Study towards Achieving 100% Recall of Synonyms in Software Requirements Documents with Selected Methods" 2015 (Waterloo)**