Importance of Ignorance

Based on:

Importance of Ignorance in Requirements Engineering

Daniel M. Berry

Journal of Systems and Software
28:2, 179–184, February, 1995

Requirements Engineering (RE)

“Programmer-Client Interaction in Writing Program Specifications” by Daniel M. and Orna Berry was written in 1980.

Ignorance Hiding

This paper makes the point that *ignorance hiding* can be used to hide the requirement engineers ignorance of the client's domain, by encapsulating that ignorance behind abstractions that can be taken as primitive.

Abstractions

Buzz words

- **nouns = types and objects**
- **verbs = functions and procedures**

Practice

Since the paper's publication, Orna and Dan have practiced ignorance hiding on a number of requirements engineering efforts.

RE Experiences -1

Many times, though, there was not much ignorance to hide.

- **Orna worked in her area of expertise, networking.**
- **Dan worked in his area of expertise, electronic publishing.**

RE Experiences -2

Both were quite satisfied with the general success of the method and would not use any other.

Orna, in industry, became known for her ability to get to the heart of requirements quickly and was in demand among several projects in the company for which she worked and in other companies.

Most Recent Experience -1

Recently, Dan was called in as a consultant to help a start-up write requirements for a new multi-port Ethernet switching hub.

Dan protested that he knew nothing about networking and Ethernet beyond nearly daily use of telnet, ftp, and netfind.

At one point, earlier in his life, he worried that the ether in Ethernet cables might evaporate!

Most Recent Experience -2

The engineers in the start-up

- **were almost exclusively hardware engineers**
- **were struggling 4 months to come up with a software requirements document and were getting nowhere fast**
- **knew the technology cold but not how to structure the software for it**

Most Recent Experience -3

The engineers in the start-up (cont'd.)

- **had not stated the requirements in full and were cycling with no convergence between requirements gathering and software design**

- **had a much stronger understanding of how to specify hardware, so that the hardware part of the project was on schedule but the software part was way behind schedule**

Most Recent Experience -4

Dan asked each person to supply him with complete lists of the pieces of the system and of the features (operations) of each.

Dan read these and began to build abstractions.

Dan noticed lots and lots of inconsistencies.

Most Recent Experience -5

Dan asked lots and lots of questions and nudged for resolutions of all inconsistencies during a 2-hour meeting.

Dan worked for 4 more hours to produce a first draft specification that seemed to have electrified the engineers.

Most Recent Experience -6

That is, in 6 hours, Dan had put down in words and diagrams what the engineers had been trying to say in 4 months

Dan continued to work over 2 more months to produce a functional specification and an architectural specification that were carefully maintained to be consistent.

Ignorance is the Key -1

Prior to this most recent experience, while Dan was lecturing at CMU on "Programmer-Client Interaction in Writing Program Specifications", Jim Alstad remarked that maybe the very fact that Dan knew so little about Orna's application area had been a significant factor in the success of the first experience.

Ignorance is the Key -2

By being ignorant of the application area, Dan was able to avoid falling into the tacit assumption tarpit!

The latest experience seems to confirm the importance of the ignorance that ignorance hiding is so good at hiding.

Ignorance is the Key -3

It was clear to Dan that the main problem preventing the engineers at the start-up from coming together to write a requirements document was that

- **all were using the same vocabulary in slightly different ways,**
- **none was aware of any other's tacit assumptions, and**
- **each was wallowing deep in his own pit.**

Ignorance is the Key -4

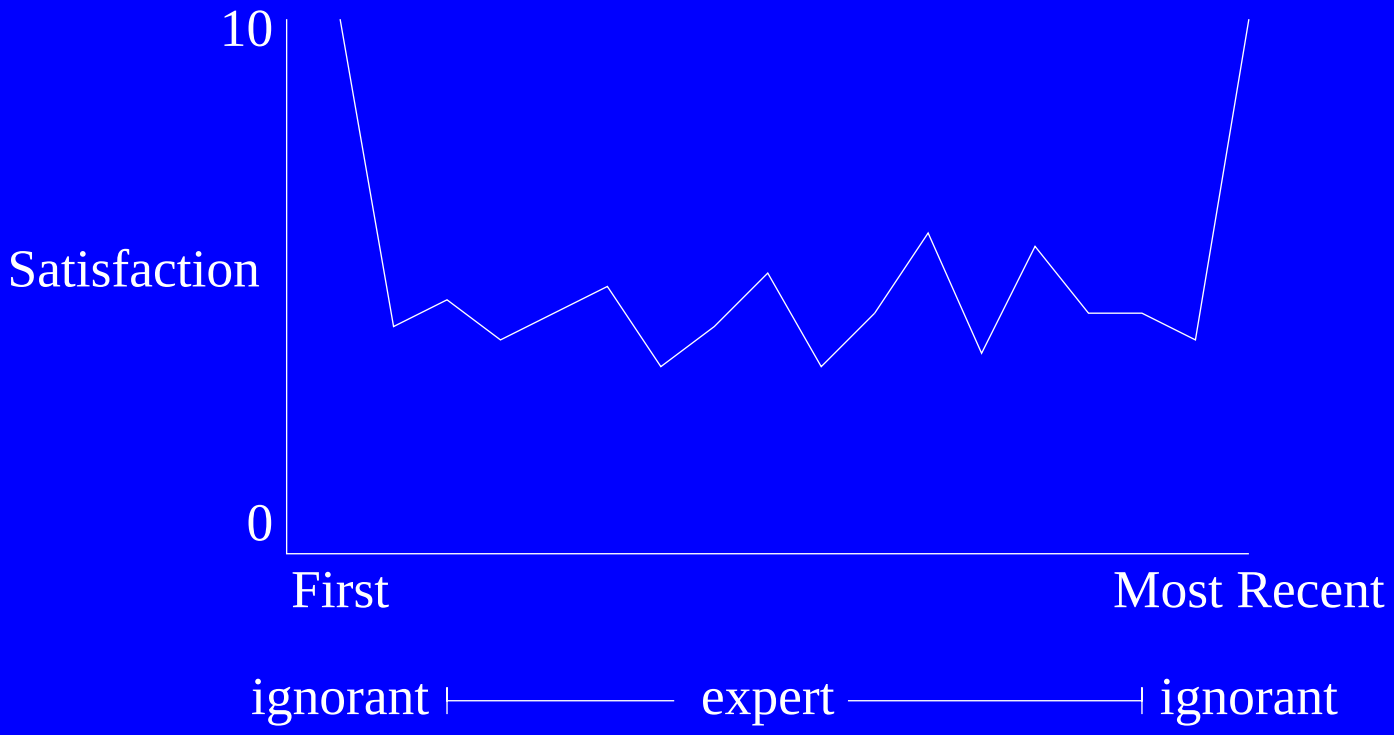
Dan's lack of assumptions forced him

- **to ferret out these assumptions and**
- **to regard the ever so slight differences in the uses of some terms as inconsistencies.**

Retrospective -1

Looking back over the history of applying ignorance hiding, Dan and Orna observed that the first and the most recent applications were the most successful in terms of their own satisfaction with the results.

Retrospective -2



History of Ignorance Hiding Experiences

Retrospective -3

In the first and most recent cases, the requirement engineer was ignorant of the domain.

In all other cases, the requirements engineer was in his or her field of expertise.

Need Ignorance

Our conclusion is that every requirements engineering team requires a person who is ignorant in the application domain, the ignoramus of the team, who is not afraid to ask questions that show his or her ignorance, and who *will* ask questions about anything that is not entirely clear.

Still Need Experts

We are not claiming that expertise is not needed.

***Au contraire*, you cannot get the material in which to find inconsistencies without the experts.**

Ignorance, Not Stupidity!

We are not claiming that the ignoramus is stupid.

***Au contraire*, he or she must be an expert in general software system structures and must be smart enough to catch inconsistencies in statements made by experts in fields other than his or her own.**

Other Sightings

Bob Glass reports how successfully non-software-knowledgeable English majors were able to write high-quality descriptions of the grubby details of programs in documentation about these programs for maintainers.

Recommendations

Each requirements engineering team needs

- **at least one domain expert, usually supplied by the customer**
- **at least one smart ignoramus**

What if Not Ignorant

What do you do if you're not ignorant?

Linda McCalla reports that she has become expert in faking ignorance.

"I know that I've been fakin' it!"
— Paul Simon

"A Rational Design Process:
How and Why to Fake it"
by David L. Parnas and Paul Clements

Resumes of the Future

Resumes of future software engineers will have a section proudly listing all areas of ignorance.

This is the only section of the resume that shrinks over time!

The software engineer will charge fees according to the degree of ignorance: the more ignorance, the higher the fee!

The Optimist

From the Berlitz Italian Book:

The head of an important firm, looking at an application, is astonished when he notices that the applicant, though lacking experience, asks for a high salary.

Rather puzzled, he asks him, “Doesn’t it seem to you that you are asking for an excessive salary, considering the little experience you have?”

“On the contrary,” replies the applicant. “Work performed by one who knows nothing about it is harder and should be better paid.”

Mathematicians as Ignoramuses

Martin Feather of JPL on Importance of Ignorance Paper:

I have often wondered about the success stories of applications of formal methods. Should these successes be attributed to the formal methods themselves, or rather to the intelligence and capabilities of the proponents of those methods? Typically, proponents of any not-yet-popularised approach must be

skilled practitioners and evangelists to come to our attention. Formal methods proponents seem to have the additional characteristic of being particularly adept at getting to the heart of any problem, abstracting from extraneous details, carefully organizing their whole approach to problem solving, etc. Surely, the involvement of such people would be beneficial to almost any project, whether or not they applied “formal methods.”

Daniel Berry’s contribution to the February 1995 Controversy Corner, “The Importance of

Ignorance in Requirements Engineering," provides further explanation as to why this might be so. In that column, Berry expounded upon the beneficial effects of involving a "smart ignoramus" in the process of requirements engineering. Berry argued that the "ignoramus" aspect (ignorance of the problem domain) was advantageous because it tended to lead to the elicitation of tacit assumptions. He also recommended that "smart" comprise (at least) "information hiding, and strong typing ... attuned to spotting inconsistencies ... a good memory ...

a good sense of language...," so as to be able to effectively conduct the requirements process.

Formal methods people are usually mathematically inclined. They have, presumably, spent a good deal of time studying mathematics. This ensures they meet both of Berry' criteria. Mastery of a non-trivial amount of mathematics ensures their capacity and willingness to deal with abstractions, reason in a rigorous manner, etc., in other words to meet many of the

characteristics of Berry's "smartness" criterium. Further, during the time they spent studying mathematics, they were avoiding learning about non-mathematics problem domains, hence they are likely to also belong in Berry's "ignoramus" category. Thus a background in formal methods serves as a strong filter, letting through only those who would be an asset to requirments engineering.

Formal methods proponents would have us believe that there is intrinsic value to the application of the methods themselves. With

the above considerations in mind, there are two ways in which this might be pursued: (1) teach those of us who are not so mathematically adept to employ them (and observe whether or not we succeed); (2) embed those methods inside automated tools suitable for application, with little extra effort, by non-formal-methods people. If we are to believe Berry, we will find some of these tools that are deliberately made to be ignorant of the application domain to be valuable for requirements engineering!

The Brainstormers

Dear Dan

Let me tell about our business. Both Sandy & I are writers. We love to express thoughts on paper. We often write together. We are comfortable editing each other's work. We often brainstorn (brnstorm, our e-mail address) together while writing. We have been trying to earn a living writing grant proposals for schools and other non-profit institutions. We are both very involved in the

schools. We often develop programs in order to apply for funds. We have been able to get to the heart of a need, describe it in detail, and suggest a resolution which appears to be fundable. The biggest problem has been getting paid to do this. We believe our specialty is to be able to think in creative paths. We do not tend to repeat others' work. I often look at something upside down or inside out. For instance, when faced with the problem of teachers who have tenure, but are ruining kids' love of learning, I suggested giving them no kids, paying their contract as

obligated by union rules, but not compoundiing the error by also giving them kids to destroy. On more mundane levels, I offered to set up charter schools to give those kids who want a minimum education, and the degree without having them take and fail the higher level courses which they were taking to fill in their schedule. This frees up space for those who really want to take the couorse, and frees up the teacher from having disinterested kids. Those kids would graduate anyway, but now they can get done quicker and make room for our growing enrollment.

Basically, we try to think new thoughts. We often say that our specialty is NOT knowing anything about your business, so we have no preconceptions, no blinders, no paradigms to overcome.

A hell of a way to earn a living, eh?

Dick

BRAINSTORMERS!!!!
We can think for you,
like you've never thought before!

Dick Nepon
Sandy Tenney
brnstorm@postoffice.ptd.net
1724 West Congress Street
Allentown, PA 18104-3104

610.437.9194
610.770.0770 fax
brnstorm@postoffice.ptd.net