Role of Domain Ignorance in Software Development

UNIVERSITY OF WATERLOO

uwaterloo.ca

Gaurav Mehrotra

Master's Thesis Presentation

Daniel M. Berry, Supervisor

Outline

- Motivation
- Related Work
- Problem Statement
- Empirical Study Design
- Results
- Applications
- Future Work
- Conclusion

Some Terminology

- *New hires* and *immigrants* in a project or team are collectively called *newbies*.
- *Domain ignorance* and *domain awareness* are together referred to as kinds of *domain familiarities*.
- We are talking about *application domain* ignorance; we assume that all employees are competent in computer science.

More Terminology

In the following slides, “I” = Gaurav Mehrotra”, the student who wrote the thesis.

Motivation

- At ICSE 2010, Berry had heard the presentation of a paper by Dagenais *et al* studying the immigration of newbies into software development projects, with an aim to determine how to make these immigrations smoother.
- They described one example of a smooth immigration, of a person assigned a debugging task

Motivation

- Berry was aware of his earlier work on the importance of ignorance in requirements engineering and knew from experience that debugging was often done best by a domain ignorant.
- Berry hypothesized that immigration was smoothest when the immigrant was put to work doing a task for which domain ignorance is helpful.

Related Work

- P. Burkinshaw, an attendee of the Second NATO Conference on software engineering in Rome in 1969, said: “Get some intelligent ignoramus to read through your documentation and try the system; he will find many holes where essential information has been omitted. Unfortunately intelligent people do not stay ignorant too long, so ignorance becomes a rather precious resource.”

Related Work

- Naggapan *et al* studied the impact of computer science educational background on requirements inspection effectiveness.
- Inspectors who had a background that was unrelated to computing were significantly more effective in identifying defects.
- Kenzi *et al* conducted an exploratory study of the perceptions of requirements analysts of the role of domain ignorance in RE, only a part of SE.

Problem Statement

This research aims to answer two important questions:

- Are there software development activities that are helped by domain ignorance?
- What role does domain ignorance play in various software development activities?

Empirical Study Design

- A cross-sectional survey listing various software development activities was chosen.

- Since the participant pool for the survey should be limited to people having significant experience managing software development, a snowball sampling based on judgmental sampling was chosen for the research.

Empirical Study Design

A five-point ordinal scale was used to categorize the importance of or the effect of each kind of domain familiarity on any software development activity.

Empirical Study Design

The categories chosen were

- Required
- Enhances

} Helps

- Neutral
- Impedes
- Prevents

} Hinders

There is an ordering, but the distance between elements is not known.

Empirical Study Design

- A pilot survey was constructed and deployed in order to ensure that the survey questions were understandable and that the list of activities was complete.

- Similar activities were grouped together as a result of participant feedback from the pilot survey.

Empirical Study Design

2.

Architectural Tasks

*** 7. Designing and specifying software architecture.**

	Required	Enhances	Neutral	Impedes	Prevents
Domain Ignorance	○	○	○	○	○
Domain Awareness	○	○	○	○	○

*** 8. Reviewing software architecture.**

	Required	Enhances	Neutral	Impedes	Prevents
Domain Ignorance	○	○	○	○	○
Domain Awareness	○	○	○	○	○

Requirements Engineering Tasks

*** 9. Eliciting requirements/Requirements gathering.**

	Required	Enhances	Neutral	Impedes	Prevents
Domain Ignorance	○	○	○	○	○
Domain Awareness	○	○	○	○	○

*** 10. Analyzing Requirements.**

	Required	Enhances	Neutral	Impedes	Prevents
Domain Ignorance	○	○	○	○	○
Domain Awareness	○	○	○	○	○

*** 11. Specifying requirements.**

	Required	Enhances	Neutral	Impedes	Prevents
Domain Ignorance	○	○	○	○	○
Domain Awareness	○	○	○	○	○

Results

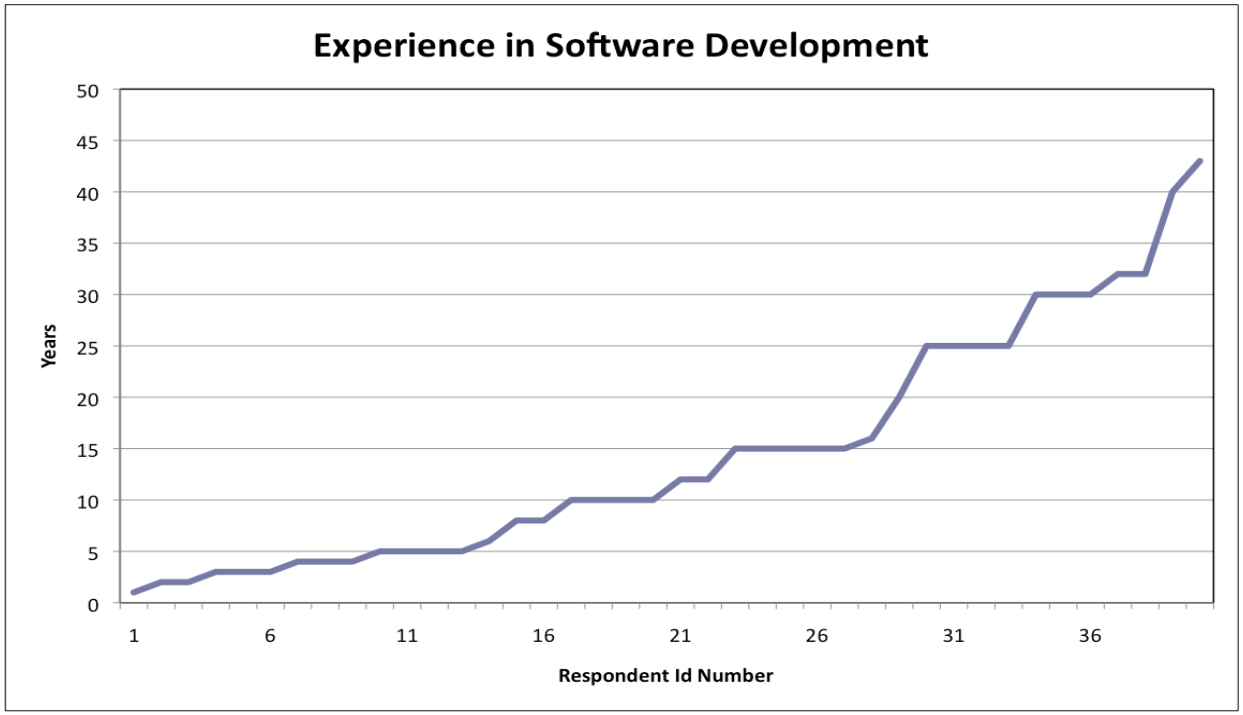
40 respondents from Canada, India, Israel, UK, and US industry and academia completed the online survey.

Type of organization:

- Commercial – 30
- Research – 10

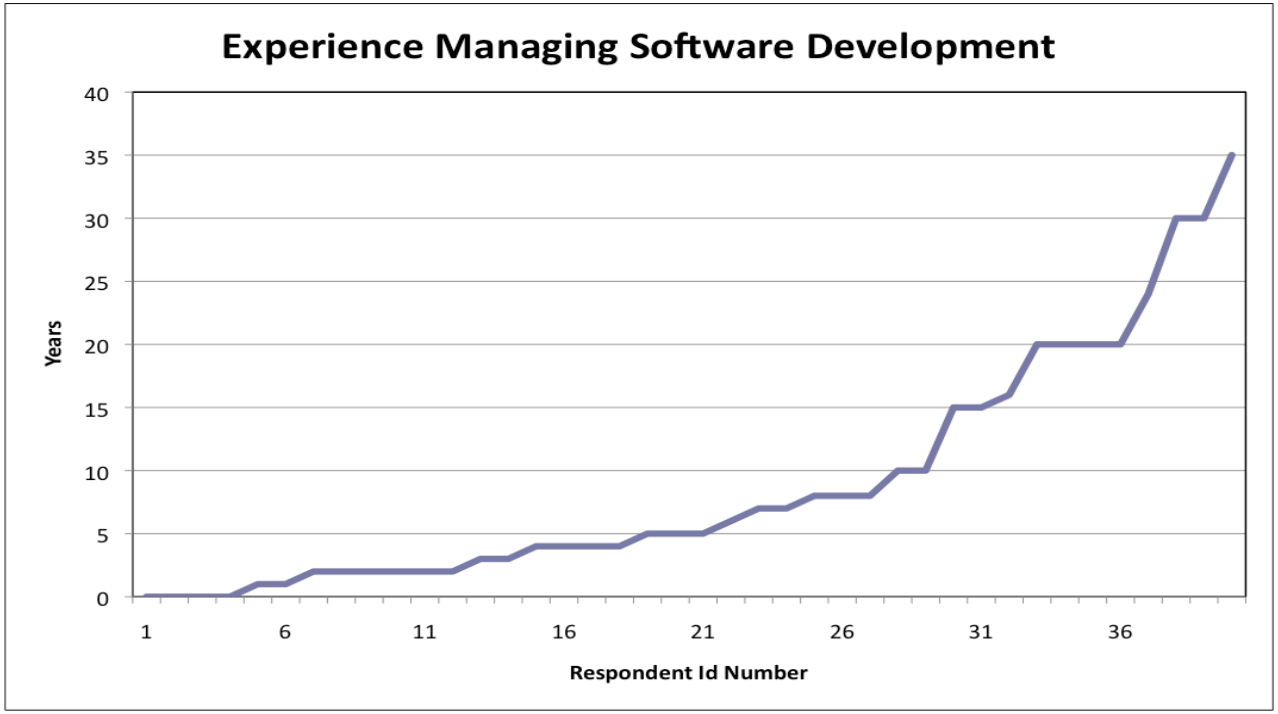
Moreover, I tried to target among the research people those who probably had experience managing software developments.

Respondents' Experience in Software Development



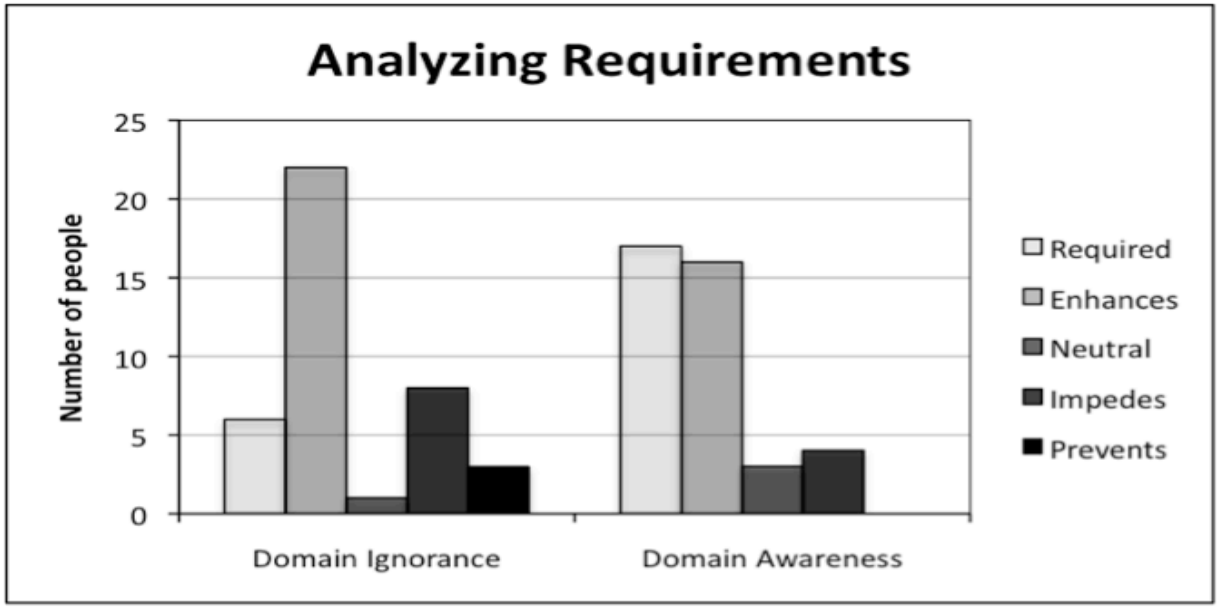
Min: 1 yrs, Avg:14.5 yrs, Max: 43 yrs.

Respondents' Experience Managing Software Development



Min: 0 yrs, Avg: 9 yrs, Max: 35 yrs.

How Results Were Obtained



Mode (Domain Ignorance) = “Enhances”

Mode (Domain Awareness) = “Required”

How Results Were Obtained

Mode(DI)= “Enhances”, Mode(DA)= “Required”

➔“Domain ignorance enhances analyzing requirements.”

➔ “Domain awareness is required for analyzing requirements.”

Note that the perceptions of the respondents are described as facts to avoid having to say “is perceived by the respondents as” for everything.

Domain Ignorance Helps:

(Bold face means that also domain awareness helps the activity.)

- **requirements gathering**,
- analyzing requirements,
- identifying project risks,
- creating high-level software design,
- **user interface design**,

Domain Ignorance Helps:

- **developing black box test cases**,
- **analyzing defects to find common trends**,
- identifying security risks,
- writing user manuals/release notes,
- inspecting/reviewing design documents,
- **inspecting/reviewing test plans**,

Domain Ignorance Helps:

(Italics means that domain ignorance enhances, while domain awareness is required for the activity.)

- *inspecting/reviewing requirement documents*,
- **inspecting/reviewing user manuals**,
- **reading user manuals/design documents/ other product documentation**, and
- learning processes/technology/practices used in the project.

Domain Ignorance Hinders:

(Bold face means that Dagenais *et al* report that the activity was done by newbies with smooth immigrations.)

- designing and specifying software architecture,
- **reviewing software architecture**,
- specifying requirements,
- validating requirements,
- reusing and managing requirements,
- inspecting code,

Domain Ignorance Hinders:

- managing builds of a software,
- deployment planning,
- risk planning/monitoring and control,
- creating low level software design,
- preventing security threats,
- identifying design and implementation rationale,
- **fixing bugs**,

Irony

"fixing bugs" is the very task that Berry thought was helped by domain ignorance and whose description in the talk by Dagenais *et al* led to his hypothesis, which is being tested by this empirical study.

More later about this irony.

Domain Ignorance Hinders:

- developing unit test cases
- developing white box test cases,
- developing integration test cases,
- determining source of a bug,
- test planning for a release,
- developing system/performance test cases,
- **manually executing test cases**, and
- providing technical support to users.

Domain Ignorance Not Affect:

(Bold face means that Dagenais *et al* report that the activity was done by newbies with smooth immigrations.)

- learning processes/practices/technology used,
- **source/version control tasks**,
- coding simple features,
- other code oriented tasks,
- automating test cases,

Domain Ignorance Not Affect:

- reviewing trace information,
- attending courses/trainings,
- attending formal project meetings,
- attending code/project walkthroughs,
- **compiling project code**, and
- **installing and configuring development environment**.

Recall Berry's Hypothesis

A newbie's immigration into a project is smoothest when he or she is assigned tasks that are helped by domain ignorance

- He or she is immediately useful while
- learning the domain proceeds more naturally and with less pressure

I decided to test the hypothesis by getting raw data from Dagenais *et al* and comparing those data with the results I got from the survey.

Testing Hypothesis

- Dagenais *et al* studied the immigration of newbies into software development projects, with an aim to determine how to make these immigrations smoother.
- Transcripts containing information regarding the tasks a newbie was assigned during his initial days and the difficulties faced by him in doing those tasks were obtained from Ossher (one of the co-authors).

Two Groups

I analyzed the transcripts and grouped the activities performed by the 14 newbies into two groups:

1. a *positive group* that contains each activity for which at least one newbie said that the activity helped him immigrate, and
2. *a negative group* that contains each activity for which at least one newbie said that the activity did not help him immigrate.

Positive Group Activities

[e]=enhanced in survey, [i]=impedes, [n]=neutral

- Reading product documentation [e]
- Inspecting test plans, design documents [e]
- Fixing bugs [i]
- Learning processes [n]
- Coding simple features [n]

Positive Group Activities

- Reviewing trace information [n]
- Attending code/project walkthroughs [n]
- Compiling project code [n]

Negative Group Activities

- Installing/configuring development environment [n]
- Source/version control tasks [n]
- Attending formal project meetings [n]
- Writing design documents/software architecture [i]

Drawback of Transcripts

A drawback of the transcripts is that they did not contain any data about how smooth the immigrations were.

It would be very nice to be able to correlate the transcript results with such data.

Drawback of Transcripts

The best judge of the smoothness of a newbie's immigration is the newbie himself.

So, I decided to request additional data from Ossher.

I got back Dagenais's binary classification of each newbie's immigration experience, successful or not.

I conflated "successful" with "smooth".

Two Lists

I built two lists of activities:

1. one of all activities done by anyone who had a smooth immigration and

2. another of all activities done by anyone who did not have a smooth immigration.

Smooth Immigration Activities

- Reading product documentation [e]
- Inspecting test plans, design documents [e]
- Fixing bugs [i]
- Learning processes [n]
- Coding simple features [n]

Smooth Immigration Activities

- Reviewing trace information [n]
- Attending code/project walkthroughs [n]
- Installing/configuring development environment [n]

Non-Smooth Immigration Activities

- Source/version control tasks [n]
- Attending formal project meetings [n]
- Writing design documents/software architecture [i]
- Inspecting Code [i]

Comparison of Two Pairs of Lists of Activities

Positive Group vs Smooth Immigration Activities

Reading product documentation [e]

Inspecting test plans, design documents [e]

Fixing bugs [i]

Learning processes [n]

Coding simple features [n]

Reviewing trace information [n]

Attending code/project walkthroughs [n]

Compiling project code [n]

Reading product documentation [e]

Inspecting test plans, design documents [e]

Fixing bugs [i]

Learning processes [n]

Coding simple features [n]

Reviewing trace information [n]

Attending code/project walkthroughs [n]

Installing/configuring development environment [n]]

Comparison of Two Pairs of Lists of Activities

Negative Group vs Non-Smooth Immigration Activities

Installing/configuring development environment [n]

Source/version control tasks [n]

Attending formal project meetings [n]

Writing design documents/software architecture [i]

Source/version control tasks [n]

Attending formal project meetings [n]

Writing design documents/software architecture [i]

Inspecting Code [i]

Comparison of Two Pairs of Lists of Activities

Note that there is a good overlap

- between the positive and the smooth immigration activities lists and
- between the negative and the non-smooth immigration activities lists.

Distillation of Activities Domain Ignorance Enhances

If the neutral activities are eliminated from the two pairs of lists,

- the positive plus smooth immigration lists are left with a majority of activities that domain ignorance is thought to enhance, and
- the negative plus non-smooth immigration lists are left with only activities that domain ignorance is thought to impede.

Distillation of Activities Domain Ignorance Enhances

Therefore, there is very marginal support for Berry's hypothesis.

It is somewhat ironic that the original task that prompted Berry to make his hypothesis, the task of fixing bugs, that Berry's experience told him benefited from domain ignorance, ended up being thought as one that is impeded by domain ignorance.

The Irony About Fixing Bugs

Perhaps, as suggested by Alan Wexler, the problem with the “fixing bugs” activity is that it really consists of two activities:

1. finding the source of the big, and then
2. fixing it.

- The first benefits from domain ignorance.
- The second requires domain awareness.

We need to change the list of activities in the future.

Why is support only marginal?

It is clear that in real life, there are many factors affecting smoothness of immigration, including personality.

Without doing a controlled experiment (which perhaps is not real life) we cannot isolate any factor.

So, the best we can say is that all other factors being equal, perhaps it's best to assign a newbie to an activity that domain ignorance helps.

Threats to Validity of Survey Conclusions

Internal Validity

- chosen sampling method,
- survey questions,
- potential non-uniform understanding of terms, &
- method to compute results, using modes.

External Validity

- is the sample representative?
- is the sample big enough?

Threats to Validity of Hypothesis Conclusion

Internal Validity

- are the right variables controlled?
- method of determining positive & negative and smooth & non-smooth activities, &
- potential non-uniform understanding of terms

External Validity

- is the sample representative?
- is the sample big enough?

Miracle?

With all the threats, with words possibly meaning different things to different people, and three independent sources of data, …

the expectation was that the results would wash out and nothing would be shown.

Nevertheless, a *small* effect of domain ignorance was observed, consistent with the fact that there are factors other than domain familiarity at play here.

Applications

- As a checklist to assign suitable roles to a newbie. Assign him or her to a task for which domain ignorance helps.
- Finding activities suitable for crowdsourcing
- Selecting the right mix of people for an activity.

Assignments for Newbies

Pnina Soffer: change recommendation to say that newbies should be assigned to tasks for which DI helps *or is neutral*.

At least newbie is not a drag on others.

Future Work

- Repeat study using a focus group of senior managers.
- Try a different grouping of software development activities, including splitting tasks that are helped by both ignorance and awareness into their components, e.g., bug fixing = bug finding + bug repair.
- Directly observe newbies working on the assigned tasks during their immigration.

Conclusions

- There are many factors that influence newbie immigration in an organization.
- Assuming all factors have equal weight, domain ignorance might be one of the important factors which affects newbie immigration.

Questions/Comments