Role of Domain Ignorance in Software Development: How Domain Ignorance Helps Immigration to Software Development Projects

Gaurav Mehrotra and Daniel M. Berry
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
gaurav.iiita@gmail.com, dberry@uwaterloo.ca

***Abstract*—A companion paper has yielded a list of software development activities that are thought to be at least helped by domain ignorance. This paper reports on an examination of transcripts from fourteen interviews of presumably-domain-ignorant immigrants to new software development projects at one large company to determine if the activities performed by those with the most successful immigrations were the activities that are at least helped by domain ignorance. The conclusions are that ignorance can play an important role in software development but there are a lot of other factors that influence immigration success.**

***Keywords*-assigned software engineering tasks, domain awareness, domain ignorance, immigration success, immigration to a new project, importance of ignorance, new hires,**

I. Introduction

This paper is the second of two companion papers that explore the role of domain ignorance in software development. The first paper [1] determines, with the help of a survey conducted among software development managers of varying experience, what software development activities they thought are at least helped by ignorance of the domain of the software system under development, hereinafter called simply "domain ignorance". The opposite of domain ignorance is domain awareness. Moreover, it is assumed that ignorance in the development personnel is restricted to the domain of the software system under development. That is, it is assumed that all personnel involved in the development are competent in the skills required for their tasks in the development.

This second paper examines transcripts from fourteen interviews of presumably-domain-ignorant immigrants to new software development projects at one large company to determine if the activities performed by those with the most successful immigrations were activities that are at least helped by domain ignorance.

Ignorance of the domain is thought by some to be helpful in software development activities that require some critical, out-of-the-box thinking [2] [3, p. 18].

A *new hire in an organization* or an *immigrant to a software system development project within an organization* could use her[1] domain ignorance to perform tasks of the development that are helped or at least not hindered by her ignorance. *New hires* and *immigrants to a project* are collectively called "newbies" in this paper.

A newbie is often clueless about the domain into which she is thrust upon arrival in the new environment and does not possess the skills necessary to be productive immediately [4], [5]. She is left to wander on her own and learn the tasks by trial and error or, in some cases, a senior member of her new team, a mentor is assigned the job of training her, but the mentor spends a lot of his time training rather than doing his normal activities for the team [6]. Despite her ignorance about the new domain, there seem to be some software development activities that a newbie can perform effectively even better than a seasoned expert in the same domain. An expert takes many things as assumed or implied that an ignorant newbie would have to explicitly think about and evaluate from first principles.

A review of the existing literature (See Section II) shows that that newbies who perform specific activities *seem* to have more successful start ups than newbies who perform other activities [7]. Here "more successful" is used as in the vernacular to mean that the start up occurred with much more positive than negative events so the newbie reports feeling good about the start up and regarding it as a success. Perhaps the activities done by newbies with more successful start ups require or are helped by some level of domain ignorance. The review showed also that some activities seem to yield better results when performed by a domain ignorant than by a domain aware.

These two observations lead to the hypothesis that *a newbie who starts with software development activities that are thought to be helped by domain ignorance has a more successful start up than a newbie who starts with other activities.* The first step was to collect data about the effect of domain ignorance in various software development activities. The results of this first step were reported in the companion paper [1]. The main result of that paper was a

[1]The gender of the first general individual in any discourse toggles with each section. The gender of the second general individual in any discourse is the opposite of that of the first.

list of software development activities thought to be at least helped by domain ignorance:

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

The main contributions of the present paper are

1) an examination of the histories of some newbies' start ups to obtain additional *lists*, of the software development activities performed by those newbies with successful start ups and
2) a comparison of these *lists* of software development activities to find any correlations.

Specifically, the research reported in this paper aimed to answer one important research question:

RQ2: Can the presumed domain ignorance of a newbie in a software system development be used to an advantage in the development?

Related work is discussed in Section II. Section III reports the results of the comparison of the lists of activities. Section IV discusses the threats to all conclusions in Section III. Section V describes applications of the results, and Section VI summarizes the conclusions and discusses future work.

II. Related Work

Begel and Simon observed eight graduate students during their first months of work at Microsoft [4]. They found that most of the difficulties encountered by the new hires came from their inexperience with a corporate environment. However, employers recognize that students entering the workforce directly from university training often do not have the complete set of software development skills that they will need to be productive, especially in large software development companies [8].

Sim and Holt interviewed four recently hired developers at a big software company and identified seven integration patterns [5]. Based on the patterns they drew several conclusions regarding the strengths and weakness of the naturalization process within an organization. Some of their important findings were: mentoring is an effective although inefficient way to train newbies, administrative and environmental issues were a major frustration during the immigration, and initial tasks assigned to newbies were often open-ended problems.

Dagenais et al. [7] categorized the project landscape that newbie need to learn and also identified the obstacles and orientation aids encountered by newbies in the context of various integration factors.

Schein proposed that there were three main aspects to introducing newbies to organizations: function, hierarchy, and social networking [9]. Function represents the tasks and technical requirements of a position. Hierarchy is the organizational command structure, and social networking is the movement of the newcomer from the periphery of the network towards the centre as new personal connections are made.

DeMarco and Lister [6] observed, "We all know that a new employee is quite useless on day one or even worse than useless, since someone else's time is required to begin bringing the new person up to speed."

Fred Brooks observed that adding a new person to a late project makes it even later [10].

III. Comparison of Lists of Activities

At ICSE 2010, prior to the beginning of the work on the thesis leading to this paper, author Berry had heard the presentation of a paper by Barthélémy Dagenais, Harold Ossher, Rachel Bellamy, Martin Robillard, and Jacqueline de Vries [7] studying the immigration of newbies into software development projects, with an aim to determine how to make these immigrations more successful. Based on the one example presented in the talk of a successful immigration, and aware of his earlier work [2] on the importance of ignorance in requirements engineering, Berry offered the hypothesis that

> an immigrant's immigration was most successful when the immigrant was put to work doing a task for which domain ignorance is helpful.

The example newbie who had a successful immigration had been assigned the task of fixing bugs, and Berry's experience told him that fixing bugs is an activity that benefits from domain ignorance[2].

Armed with the list of software development tasks that are believed to benefit from domain ignorance, the next step was to try to test the hypothesis by re-examining the data from the immigration study [7] to determine the tasks performed by the immigrants with the most successful immigrations.

Mehrotra and Berry approached Dagenais et al. for access to their raw data, transcripts describing their newbies' immigration experiences. After some discussions, Mehrotra

[2]Note that the results of the survey say otherwise, but this hypothesis was formed long before the survey was even written, and it prompted the research leading to this survey.

and Berry signed a non-disclosure agreement and obtained the transcripts about the immigrations of 14 newbies. These transcripts contained information regarding the tasks a newbie was assigned during her initial days and the difficulties faced by her in doing those tasks.

A. Analyses of Transcripts

To do the analysis, Mehrotra decided to examine the transcripts and group the activities performed by the 14 newbies into two lists,

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

If an activity were initially in both lists, then Mehrotra would put it finally in the positive list if it helped more people than it did not help, and Mehrotra would put it finally in the negative list if it did not help more people than it helped. As it turned out, no activity was initially in both lists. The results are summarized in the two lists below. For each entry in each list, the number in the parentheses is the number of newbies mentioning the activity for the list, and the text in the square brackets at the end of the entry denotes the mode of domain ignorance of the entry's activity in the results of the survey presented in Sections III.C–E of the companion paper [1].

The activities in the positive list are:

- Reading product documentation (4) [enhances]
- Inspecting test plans, design documents (1) [enhances]
- Fixing bugs (1) [impedes]
- Learning processes/practices/technology (4) [neutral]
- Coding simple features (1) [neutral]
- Reviewing trace information (1) [neutral]
- Attending code/project walkthroughs (1) [neutral]
- Compiling project code (2) [neutral]

The activities in the negative list are:

- Installing/configuring development environment (2) [neutral]
- Source/version control tasks (1) [neutral]
- Writing design documents/software architecture (1) [impedes]
- Attending formal project meetings (3) [neutral]

If the activities that are thought to be neutral are eliminated from the two lists,

- the positive list is left with a majority of activities that domain ignorance is thought to enhance, and
- the negative list is left with one activity that domain ignorance is thought to impede.

Therefore, there is very marginal support for the hypothesis.

A drawback of the transcripts is that they did not contain any data about how successful the immigrations were. It would be very nice to be able to correlate these results with such data. Recognizing that the best judge of the success of a newbie's immigration is the newbie himself, Mehrotra decided to request additional data from Dagenais *et al.* Mehrotra asked Ossher to send the following question to each of the 14 participants of his study, whose transcripts Mehrotra had received:

> Rate your immigration experience using the scale:
> Torture, Painful, Neutral, Smooth, Ecstatic

As the immigration study was done a long time ago, Ossher was reluctant to contact the participants again and did not agree to send the question to the 14 participants. He did offer instead some additional data derived by Dagenais during the study for each participant, a binary classification, successful or non-successful, of her immigration experiences and the reason for the classification. Note that this classification was performed by an independent third party using some definition that he chose. Mehrotra decided to accept this classification at face value, taking Dagenais's "successful" to have the vernacular meaning of "successful".

Out of the total 14 participants, Participants 6, 7, 11, and 14 reported an overall non-successful immigration. Some of the reasons given for the non-successful immigration are:

- Participant 6[3] got assigned to a job for which she was not qualified; her colleagues told her in various indirect ways that she should not have this position; and she got assigned to critical tasks with insufficient support. C
- Participant 7 was a team leader and her team was supposed to take over a project from another team, but the original team did not want to relinquish the project; the original team put a lot of obstacles in her way, including rude comments, outdated documentation, and long delays in answering e-mail.

Mehrotra divided the newbies into two groups:

1) one of those who had a successful immigration and
2) and another of those who did not have a successful immigration.

Mehrotra then decided to build two lists of activities.

1) one of all activities done by anyone who had a successful immigration and
2) and another of all activities done by anyone who did not have a successful immigration.

If an activity were initially in both lists, then Mehrotra would put it finally in the successful immigration activities list if more people in the successful immigration group performed the activity than people in the other group, and Mehrotra would put it in the non-successful immigration activities list if more people in the non-successful immigration group

[3]Recall that the gender of the first arbitrary individual in a discourse toggles for each section, but remains constant in a section. Thus, the use of the female gender in this odd numbered section does not imply that the referenced participant is necessarily a female.

performed the activity than people in the other group. As it turned out, no activity was initially in both lists.

The activities in the successful immigration list are:

- Reading product documentation (4) [enhances]
- Inspecting test plans, design documents (1) [enhances]
- Fixing bugs (1) [impedes]
- Learning processes/practices/technology (4) [neutral]
- Coding simple features (1) [neutral]
- Reviewing trace information (1) [neutral]
- Attending code/project walkthroughs (1) [neutral]
- Installing/configuring development environment (2) [neutral]

The activities in the non-successful immigration list are:

- Writing design documents/software architecture (1) [impedes]
- Inspecting Code (2) [impedes]
- Source/version control tasks (1) [neutral]
- Attending formal project meetings (3) [neutral]

If the activities that are thought to be neutral are eliminated from the two lists,

- the successful immigration list is left with two activities that domain ignorance is thought to enhance, and
- the non-successful immigration list is left with two activities that domain ignorance is thought to impede.

Therefore, here too, there is very marginal support for the hypothesis.

Mehrotra now had two pairs of lists that should be the same if the hypothesis held and the transcripts provided full information. While the two pairs of list are not exactly the same, there is good overlap

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

The activities that ended up in only one of the pairs of lists are:

1) **Compiling project code [neutral]**: only in the positive group in the first pair of lists,
2) **Installing/configuring development environment [neutral]**: only in the negative group in the first pair of lists,
3) **Inspecting Code [impedes]**: only in non-successful immigration activities group in the second pair of lists, and
4) **Writing design documents/software architecture [impedes]**: only in the negative group in the first pair of lists.

Nevertheless, if the activities that are thought to be neutral are eliminated from the two pairs of lists,

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

Therefore, in the end, there is very marginal support for the hypothesis.

It is somewhat ironic that the original task that prompted Berry to suggest the 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. Alan Wecker believes that "fixing bugs" is too big a task. If it were divided into its constituent subtasks, (1) finding the defect and then (2) fixing the defect, then different Likert values can be assigned to each subtask. Possibly, finding the defect would be viewed as being helped by domain ignorance while fixing the defect would be viewed as impeded by domain ignorance but helped by domain awareness.

B. Discussion

That the support for the hypothesis, that immigration was most successful when the immigrant was put to work doing a task for which domain ignorance is thought to helpful, is only very marginal is not surprising. In real life, there are many factors affecting the success of one's immigration, including her personality. There are not enough data in the immigration study to determine root causes of the outcome of any immigration. The ultimate cause of a successful or non-successful immigration could be any of the other factors, some combination of factors, or yet other factors not even considered. Without doing a controlled experiment, which perhaps will not simulate real life, we cannot isolate any factor. The best that can be said is:

> All other factors being equal, there is some support that a newbie should be assigned an activity that is helped by domain ignorance.

A newbie should be assigned an activity that is helped by domain ignorance, even if for no other reasons than that

- she becomes useful to her project immediately, and
- she learns the domain of the project in a more leisurely natural manner with less pressure to apply her knowledge prematurely.

IV. Threats

The main conclusion of this paper is that the immigration of a newbie is the most successful if a newbie is assigned a task which is thought to be helped by domain ignorance. The threats to the validity of this conclusion can be divided into two classes,

1) threats to internal validity that concerns how well the case study was executed and
2) threats to external validity that concerns whether the conclusion obtained is generalizable.

The threats to internal validity of the conclusion are:

- **the lack of control over variables**:
  Because Mehrotra used transcripts supplied by a third party from a study in which they controlled the variables they needed for their study, he had no control over the data that transcripts reported. All he could do was hope that he would be able to see evidence of the variables he needed in the transcripts provided. Fortunately, he was able to find some usable evidence in all of the 14 transcripts provided.
- **the methods of determining positive, negative, successful, and non-successful activities**:
  There may be other possible methods for doing the categorization, but they did not present themselves. Note however, that the positive–negative activities classification was done in a direction different from that of the successful–non-successful activities classification. Moreover, the successful–non-successful activities classification was based on an independent classification of immigration success. That the two pairs of lists resulting from the classifications agreed so well strengthens confidence in the correctness of the methods and the validity of the conclusions.

The threats to external validity of the conclusion are:

- **representativeness of the sample**:
  The threat to the present study is the same as to the Dagenais et al. study [7].
- **number of respondents**:
  The threat to the present study is the same as to the Dagenais et al. study.

The threats described in the companion paper [1], particularly those about

- the survey questions and
- the method to compute the results using modes,

and the threats, described above, about

- the lack of control over variables and
- the methods of determining positive, negative, successful, and non-successful activities

could easily have conspired to make it impossible to draw *any* conclusions. After all, what are the chances of drawing any conclusion if people do not agree on the meanings of the descriptions of the activities? For example, deciding whether the hypothesis is supported depends on the

1) survey respondents',
2) Mehrotra's,
3) Dagenais et al.'s subjects', and
4) Dagenais et al.'s

all agreeing enough on the meaning of the descriptions of the activities, e.g., that one person's, "eliciting requirements/requirements gathering" is similar enough to all others'. The fact that these four independent sources of data have come together to support the hypothesis even marginally when there are so many other variables that could have affected the results is something of a miracle. Nevertheless, each reader must decide for himself whether to believe the conclusions.

V. Applications of Results

A software development manager can use the results of this study as a guide to help assign suitable roles to a newbie in any team. The tasks that appear to be more suitable for a newbie are the ones that are thought to be helped by domain ignorance, listed in Section I.

There are two aspects to a newbie's immigration within a project. The first is productivity during the immigration and the second is learning about the new domain. By assigning the right task to a newbie, a manager can ensure that she will be productive earlier because her ignorance is put to good use while she learns the domain, i.e., to not be ignorant. Moreover, she is not draining the productivity of the rest of the team by her needing close mentoring. As a result, her immigration into the new project is likely to be much more successful, thereby increasing the productivity of the team as whole.

VI. Conclusion and Future Work

The conclusion of this paper is that the immigration of a newbie into a software system development project may be more successful than otherwise if the newbie is assigned a development task that is thought to be helped by domain ignorance. A manager can use the results of this research in order to assign the right tasks to a newbie in his team. A newbie in turn can use his domain ignorance to be productive right from the start while beginning to learn the domain under less pressure to do it too quickly. The productivity of the entire team is increased, and the precious time of other experienced team members is saved.

There is a lot of scope for future work in this area. It would be useful to perform a study in a real work environment where newbies can be observed working on the assigned tasks during their immigration.

Acknowledgements

The authors thank Chrysanne DiMarco, Mike Godfrey, and Jo Atlee for their comments on the thesis on which this paper is based. They thank also and particularly Barthélémy Dagenais, Harold Ossher, Rachel K. E. Bellamy, Martin P. Robillard, Jacqueline P. de Vries, and their anonymous subjects for agreeing to share the data obtained from their research for comparison purposes.

References

[1] G. Mehrotra and D. M. Berry, "Role of domain ignorance in software development: Software development tasks benefiting from domain ignorance," School of Computer Science, University of Waterloo, Tech. Rep., 2012, http://se.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/MehrotraBerrySurvey.pdf.

[2] D. M. Berry, "The importance of ignorance in requirements engineering," *Journal of Systems and Software*, vol. 28, 1995.

[3] J. N. Buxton and B. Randell, "Software engineering techniques: Report on a conference," 1969, http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF.

[4] A. Begel and B. Simon, "Novice software developers, all over again," in *Proceeding of the Fourth international Workshop on Computing Education Research (ICER)*, 2008, pp. 3–14.

[5] S. E. Sim and R. C. Holt, "The ramp-up problem in software projects: A case study of how software immigrants naturalize," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 1998, pp. 361–370.

[6] T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams*. Dorset House Publishing, 1987.

[7] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. P. de Vries, "Moving into a new software project landscape," in *Proceedings of the International Conference on Software Engineering (ICSE), Volume 1*, 2010, pp. 275–284.

[8] A. Begel and B. Simon, "Struggles of new college graduates in their first software development job," *SIGCSE Bulletin*, vol. 40, pp. 226–230, 2008.

[9] E. H. Schein, "The individual, the organization, and the career: A conceptual scheme," *The Journal of Applied Behavioral Science*, vol. 7, no. 4, pp. 401–426, 1971.

[10] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975.