Information Foraging Theory for Collaborative Software Development

Irwin Kwan
School of EECS
Oregon State University
Corvallis, OR 97331 USA
kwan@eecs.oregonstate.edu

Scott D. Fleming
Computer Science Dept.
University of Memphis
Memphis, TN 38152 USA
Scott.Fleming@memphis.edu

David Piorkowski
School of EECS
Oregon State University
Corvallis, OR 97331 USA
piorkoda@eecs.oregonstate.edu

ABSTRACT
Information foraging theory describes how people gather information based on a cost-benefit model. This theory has been successfully applied to the web domain and to software engineering tools. However, little work has been done on how information foraging theory can be applied to information-seeking behavior in a collaborative software engineering setting. This paper discusses how the theory might apply to information-seeking within collaborative software-development teams, and how constructs of the theory might help aid in the design of tools and processes.

Author Keywords
Information foraging theory, collaborative software engineering, human-centric computing, theory-based software engineering

ACM Classification Keywords
H.1.2 Information Systems: User/Machine Systems

[Human information processing];

H.5.3 Information Interfaces and Presentation: Group and Organization Interfaces

[Computer-supported cooperative work]

INTRODUCTION
Software engineering is a communication-intensive activity that requires interaction with many other collaborative team members, often over large distances [1, 5]. Information Foraging Theory (IFT) explains and predicts how people navigate in response to the information in their environment [11]. IFT has been applied to software engineering [6, 9] in response to empirical evidence of both IFT's validity in behaviour prediction [11] and its practical utility for web design [14]. The potential benefit of IFT is that it provides a *theory* describing the rationale behind actions of information seekers and therefore can guide the design of tools and processes to improve collaborative software engineering (CSE).

Future of Collaborative Software Engineering'12 in conj. CSCW'12, February 11, 2012, Seattle, Washington, USA.

To date, IFT in software engineering has been applied primarily to a single developer working alone, and has modeled how a person seeks information from sources like interface elements in an IDE, web sites, and documents. Information sources in CSE, however, are not limited to the computer screen: a common situation in CSE is when a developer asks a question of a colleague. This colleague, or *informant*, will often help the developer find an answer, perhaps providing it verbally or referring the developer to another informant. How does IFT map to CSE? The objective of this paper is to explore how IFT constructs can be applied to information-seeking activities that involve multiple people who develop software.

INFORMATION FORAGING THEORY CONSTRUCTS
In IFT, a person, the *forager*, seeks information within an information environment that has a *topology*. The topology is made up of *information patches* (e.g., documents) connected by traversable *links* (e.g., hyperlinks, menus, scrolling, etc.). Each link has a cost (e.g., time to get from one end to the other, where the time is influenced by system performance and the human's cognitive and physical speed). At any moment, the forager is within a particular patch and can navigate to another patch by traversing a link. Figure 1 graphically depicts information patches and links.

Each patch contains *information features* (e.g., words, phrases, diagrams, etc.) that the forager can process. The forager seeks to find and process a particular set of information features, called *prey*, that fulfills the forager's *information goal*. Processing each information feature has a cost (e.g., time to read and understand). Some features, called *cues*, are associated with links and provide the forager with hints about what information features a link leads to.

In searching for prey, the forager is continually faced with

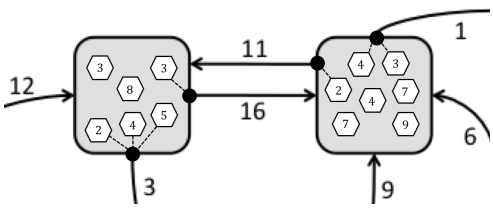

Figure 1. Example of two information patches. Each patch contains information features (hexagons) with an associated processing cost. The patches are connected to other patches via traversable links, each with a traversal cost. Cues are associated with outgoing links (dashed lines).

three choices: (1) to process information features in the current patch, (2) to move to an adjacent patch via a link, and (3) to change the environment—called *enrichment*. An example of enrichment is a forager adding a new patch, such as a Wiki page, to the topology or creating a new link. A forager may also choose to visit a new patch based on *information scent*, which is the predator's assessment, based on the cues, of the benefit to be gained by taking the associated navigation options.

The central prediction of IFT is that the forager attempts to make *optimal choices* that maximize valuable information V (value) gained per cost of interaction C (cost). However, the forager is not omniscient and cannot always predict the values of V and C that will his decisions will produce. Thus, the forager bases decisions on expected values for V and C that he infers from his knowledge, his previous experience, and information from the environment.

Extension of IFT to Social Foraging

Pirolli presented mathematical models based on optimal foraging theory to predict the effect of various social influences on IFT [12]. He theorized that diversity would increase the likelihood of discovery, that cooperative foraging would increase the likelihood of high-value discoveries, and that there is a point when having too many foragers results in a lower rate of return for each individual.

Pirolli's social IFT differs from CSE in two ways. First, the nature of how cues are exchanged among collaborators differs. Pirolli's social model includes calculations that presume that multiple foragers are cooperative, and are sending each other hints about useful information. This continuous exchange cannot be assumed in CSE because CSE is heavily specialized and involves a large amount of individual work. The individual developer might seek information by himself and contact others only when initial options are exhausted. Before this contact is made, the useful information that one developer finds is not necessarily passed on to the others.

Second, one of the sources for information can be *elicited* from the informant. If you ask a question to another software developer, and he provides an answer, he is generating in real time an information patch for the forager. Informal communication is an extremely important way that developers gather information [5, 13]. This differs from traditional IFT because the patch is elicited from a human informant.

IFT AND THE DESIGN OF COLLABORATIVE SE TOOLS

IFT already applies to many existing software engineering tools. For instance, the TagSEA tool [15] enables developers to enrich the environment by leaving cues for themselves and other developers. Another tool, Codebook [2], provides developers with a graph of relationships among artifacts and people, reducing the cost of identifying who to talk with.

The theory not only defines a taxonomy and a rationale for existing behavior, but can be used to motivate the requirements for a number of new, not-yet-invented tools. IFT can be used to develop techniques that consider the effects of the cost and the benefits of actions. In applying IFT constructs, Budiu et al. [4] found that a low-cost tagging method (click to tag) strengthened the forager's memory of the content compared to a high-cost tagging method (type to tag). The concepts of cost and value provided by IFT offer a framework for reasoning about a tool's potential foraging benefits and liabilities.

How IFT May Influence Tool Design

The two fundamental questions are, "How does my tool reduce cost C?" and "How does my tool increase value V?" Framing tool design in the form of those questions can empower the tool designer to discover new ways to achieve these ends, and using an IFT-based analysis, the tool designer can verify that her tool in fact affords these benefits. We list some potential applications of IFT below.

Reducing the cost of contacting others

By providing easily accessible contact information, the cost of contacting an informant is reduced. One of the reasons that distributed software development is so difficult is because the cost of reaching someone is dramatically increased. One example of a strategy is using a bridge team: this team can connect two remote sites by being available to both distributed sites if those two remote sites do not have overlapping working hours.

Incorporating the "social cost" of requesting information

The concept of cost C differs in a social setting compared to a non-social one. The simplest measurement of cost in IFT is *time*—in this case, the forager's time, as well as the informant's time. However there is also a social cost as well, which may be determined by personality or culture [8]. The forager may feel that the cost of talking to the informant might, for example, make him look incapable of doing his own job [3], or may annoy the informant to the point where she avoids working with the forager.

Reducing the cost of eliciting information from others

The forager must communicate to the informant his information goal set and *elicit* an information patch from her in a form the forager can process. By reducing the cost of this communication and elicitation, a tool may be able to improve a forager's effectiveness.

Increasing value of the interaction

An expertise recommendation tool can increase the value of an interaction by recommending the "right informant" for a particular question. However, the value of the interaction can be further increased if the tool also provides information about the context of the inquiry. For example, if the forager can transfer his information goal set to the informant using an expertise recommender, the informant will be better able to help the forager. Similarly, if the informant is aware of what information patches the forager has already processed, then the informant can avoid processing information patches that have already been visited.

Determining the effort of the respondents

In Pirolli's description of social IFT based on optimal foraging theory [12], a forager's selected choice is based on the optimal benefit per unit of cost. However, in a collaborative environment, the effort required on the part of the informant

is a necessary consideration. Most research on collaborative software development involving expertise seeking involves finding the expert, but does not consider cases where the informant is busy, angry at being bothered, or is otherwise incapable of answering the question. It is not a secret that people can be disgruntled or annoyed by their co-workers asking questions (e.g., [13]). Optimal information foraging theory may be able to explain why a person chooses to join another forager in the hunt for information, and can delve into conceptualizing "social cost".

Determining changing information goal sets

One aspect of IFT that can use more development is *reactive IFT* [7], which examines how acquiring information changes one's information goal set. For example, if a forager asks a question to an informant, the informant may start her own information search with the intention of helping the forager. Though they are both looking for the same information, their goals are different—one wants to find information and the other wants to find information to transmit it to the original forager. One way to accomplish this is to enable sharing or monitoring of another person's visited information patches.

Future Areas in Which to Apply IFT

Predictions of behavior

There is evidence that IFT can predict developer foraging behavior [6, 10]. By examining situations when software engineers seek information and eventually look for information from peers directly, we may be able to generate a model that identifies the likelihood that a software engineer will go to colleagues for information.

Crowdsourced software engineering

Rich sources of information are forums, social networks, and collaborative documents written by groups that are not necessarily collaborating [16]. Identifying the cost of eliciting crowdsourced data should be weighed against other choices.

CONCLUSION

The constructs of IFT are useful for highlighting principles around which to build improvements to CSE tools and processes. By viewing interactions and information in terms of cost, value, and features, a tool designer can choose different aspects to optimize for. In addition, the theory's constructs enabled us to identify situations that occur in a collaborative setting but not in an individual setting.

Understanding and using these principles as a design guideline is a first step toward using IFT for CSE. The next step would be to use the principles to attempt to predict foraging behaviour—for example, to identify if certain individuals are likely to be contacted first based on optimal foraging theory. IFT is an extremely promising theory that can model not only information seeking through documents and computer screens, but also information seeking through interpersonal communications.

REFERENCES

1. J. Aranda and G. Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proc. ICSE '09*, pages 298–308, 2009.
2. A. Begel, K. Y. Phang, and T. Zimmermann. Codebook: Discovering and exploiting relationships in software repositories. In *Proc. ICSE '10*, 2010.
3. A. Begel and B. Simon. *Novice Professionals: Recent Graduates in a First Software Engineering Job*, chapter 26, pages 495–516. in Making Software: What Really Works, and Why We Believe It. O'Reilly Media, Inc., 2011.
4. R. Budiu, P. Pirolli, and L. Hong. Remembrance of things tagged: How tagging effort affects tag production and human memory. In *Proc. CHI '09*, 2009.
5. A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proc. ICSE '07*, pages 344–353, 2007.
6. J. Lawrance, C. Bogart, M. Burnett, R. Bellamy, K. Rector, and S. Fleming. How programmers debug, revisited: An information foraging theory perspective. *IEEE Trans. Softw. Eng.*, 2011. To appear.
7. J. Lawrance, M. Burnett, R. Bellamy, C. Bogart, and C. Swart. Reactive information foraging for evolving goals. In *Proc. CHI '10*, pages 25–34, 2010.
8. K. Nakakoji, Y. Ye, and Y. Yamamoto. *Supporting Expertise Communication in Developer-Centered Collaborative Software Development Environments*, chapter 11. Collaborative Software Engineering. Springer-Verlag, 2010.
9. N. Niu, A. Mahmoud, and G. Bradshaw. Information foraging as a foundation for code navigation (NIER track). In *Proc. ICSE '11*, pages 816–819, 2011.
10. D. Piorkowski, S. D. Fleming, C. Scaffidi, L. John, C. Bogart, B. E. John, M. Burnett, and R. Bellamy. Modeling programmer navigation: A head-to-head empirical evaluation of predictive models. In *Proc. VL/HCC '11*, pages 190–116, 2011.
11. P. Pirolli. *Information foraging theory: adaptive interaction with information*. Oxford Univ. Press, 2007.
12. P. Pirolli. An elementary social information foraging model. In *Proc. CHI '09*, pages 605–614, 2009.
13. A. Schröter, J. Aranda, D. Damian, and I. Kwan. To talk or not to talk: Factors that influence communication around changesets. In *Proc. CSCW '12*, 2012.
14. J. M. Spool, C. Perfetti, and D. Brittan. *Designing for the Scent of Information*. User Interface Engineering, 2004.
15. M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Trans. Softw. Eng.*, 35:470–483, July 2009.
16. C. Treude, O. Barzilay, and M.-A. Storey. How Do Programmers Ask And Answer Questions on the Web? (NIER track). In *Proc. ICSE '11*, 2011.