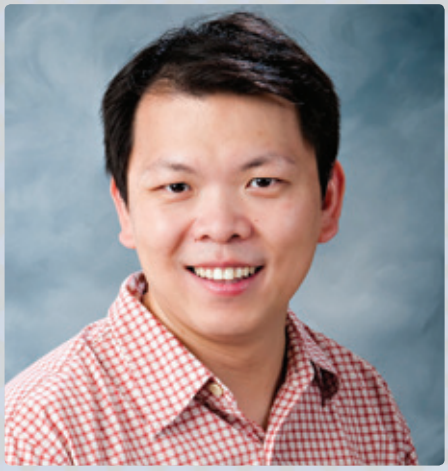
Evolutionary software engineering

Software engineer **Dr Nan Niu** understands the need to optimise the way software developers navigate the information required to debug and maintain software systems



What excites you most about computer science today?

Turing Award winner Edsger Dijkstra once said, 'computer science is no more about computers than astronomy is about telescopes'. For me, computer science is all about people. One of the most exciting aspects of the field is enabling humans to interact with each other and society as a whole in ways that would otherwise not be possible. In this context, the future can be compared to the enabling role of a sports official: although people should enjoy the game without paying much attention to the referee, without officiating or with bad officiating, things can become inconvenient, discomforting and unpleasant.

Could you give an insight into the mathematically modelled foraging mechanisms you envision informing the development of a unified theory for code navigation?

In optimal food foraging, resources are assumed to be distributed in patches (the patch model) and structural properties such as locality pose important shaping limits on the potential for optimisation. In particular, predators engage in more within-patch foraging and less between-patch foraging activities, trying to maximise their energy intake per unit cost. In information foraging on the Web, similar phenomena are observed. What we found is that the programming artefacts are not placed randomly, but are organised in a topically related manner. This patchy distribution, in turn, affects developers' navigation. Our empirical studies show that, for each between-patch navigation step taken, developers take four within-patch foraging steps on average, in order to compensate for the cost. The empirical findings not only validate foraging theory's applicability in code navigation, but also inform tool builders to take advantage of the patchy environment in which developers forage.

How has your background equipped you for this work?

After receiving my bachelor's degree from Beijing Institute of Technology, I stayed in my hometown, Beijing, and joined Lenovo, one of the top computer technology companies in China and worldwide. Two years of experience as a Lenovo programmer made me fully appreciate the practical challenges associated with developing software for millions of users. At the same time, I realised the tremendous impact that top-notch software engineering research could make.

I then pursued my graduate degree overseas, specialising in software engineering and earned my PhD from the University of Toronto, Canada, where I acquired the knowledge and skills to conduct both theoretical and empirical software engineering research. An important theme of my work has always been to address the challenges encountered by software practitioners. By forming an interdisciplinary research team and forging international collaborations, I continue to explore new and improved ways to answer the needs of today's developers.

Could you explain what sets your research apart from other work in the field?

Other studies that have applied foraging theory in software engineering so far have focused mainly on enabling developers to best shape themselves to the software and task environments. Our research tries to make a start at reversing foraging-theoretic thinking in software engineering; that is, to enable the environments to be best suited to the developers, and to do so in an automated way via clustering.

Have you encountered any specific challenges in your work?

A challenge I faced in applying foraging theory was defining a 'patch' in the code base. While certain food resources like berries are naturally organised into physical patches, source code patches may manifest themselves in many forms: stored together in folders, opened together in a programming tool, or changed together in a project's history. Instead of attempting to find the most natural source code patch, I realised the opportunity of researching multiple and possibly equally valid ways to cluster programming artefacts in patches. This fully leverages the softness of software and leads to principled ways to enrich the code foraging environment.

Looking ahead, what path will your research follow?

My research continues to take a theory-driven path and this is evidenced by my new National Science Foundation (NSF) project, entitled 'CAREER: Linking the Solo and Social Levels in Software Engineering'. Even though the specific solo-social link under investigation may vary from tagging to debugging, the research methodology will stay the same. For each instance of the solo-social link, I will research principled ways to improve the solo-level production of the information products, systematically discover the emergent knowledge structures and aggregate patterns, and leverage the social capital to boost the productivity of developers both individually and cooperatively.

Hunter, gatherer… information forager?

Computer scientists at the College of Engineering and Applied Science at the **University of Cincinnati** are combining evolutionary psychology and biology theories to understanding the human-information interactions within software engineering

DEVELOPED BY EVOLUTIONARY biologists and anthropologists, foraging theory describes how humans and other animals hunt for food. Latterly, this has been applied by Peter Pirolli, a research fellow in interactive intelligence, to information foraging and, in particular, foraging by humans for information on the Web; now known as information foraging theory. The theory draws parallels between a user searching for information on websites and settling on the information they want, with a predator hunting for prey in a particular environment and making a decision on their ideal food source. Information foraging theory links current methods of searching for information with those our ancestors evolved in order to forage for food. Understanding and applying this theory is key to optimising the layout of websites to maximise the ease of information access by users.

Despite its use in this area, information foraging theory is yet to benefit software developers that are spending time debugging software or creating it using a code base; the collection of source code and other artifacts the developer must work through in order to improve or debug a software system. Code navigation is significantly time-consuming for developers and understanding how they can evolve methods of navigation through the code base – similar to how users navigate the Web – would improve the information foraging environment and potentially enable developers to more efficiently navigate code.

EAGER

Dr Nan Niu heads up a research group that aims to understand whether software development efficiency can be improved by optimising code navigation efficiency. He and his colleagues prefer an unusual fusion of disciplines, combining evolutionary biology, evolutionary psychology and software engineering. Supported by an EArly-concept Grant for Exploratory Research (EAGER) from the National Science Foundation (NSF), the project, entitled Clustering Programming Artifacts to Enrich Code Foraging Environment, centred on three main aims. Firstly, the group sought to determine whether the biological optimal foraging theory could be applied to code navigation, as it could for information foraging on websites. Secondly, the researchers wished to determine the extent to which clustering data, ie. grouping together similar items using different algorithms, could improve the software environment. The third aim was to identify new ways to optimise the software environment; improving software development and minimising time lost to code navigation. Ultimately, the researchers aim to improve the productivity of software engineers.

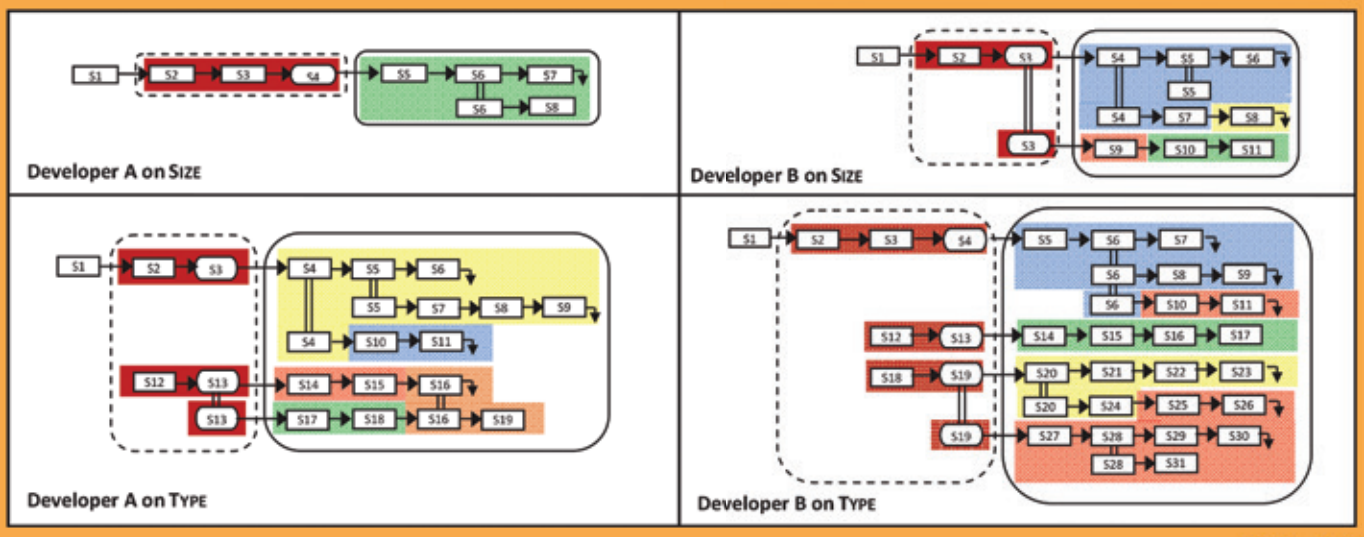
THE PATCH MODEL

Biological optimal foraging theory describes an animal that has to spend some between-patch time in travelling between food patches (eg. a bush) and when at a patch can make the decision to either continue exploiting this area for diminishing food sources, or expend more time and energy on travelling to a new patch. The patches in this model are equated to a website, an information patch, in information foraging theory.

Niu and colleagues wanted to test if the patch model could be applied to code navigation by testing topical locality, eg. analysing the extent to which a header can accurately communicate the information found in the body. Topical locality is the idea that semantic distance is similar to spatial distance within the system. The researchers studied topical locality in a series of long-term open-source projects to determine if code navigation can be modelled by the patch model. If there was little 'patchiness' or topical locality observed, foraging theory could not be applied to code navigation.

ENRICHING THE SOFTWARE ENVIRONMENT

When applying biological optimal foraging theory, the foraging patch is equated to a



Topical locality reflected by developers' more within-patch and less between-patch foraging.

CLUSTERING PROGRAMMING ARTIFACTS TO ENRICH CODE FORAGING ENVIRONMENT

OBJECTIVE

To develop a unified theory for code navigation based on mathematically modelled foraging mechanisms that evolved to help our animal ancestors to find food.

KEY COLLABORATORS

Dr Gary Bradshaw, Department of Psychology, Mississippi State University

Dr Jing-Ru C Cheng, Information Technology Laboratory, US Army Engineer Research and Development Center

Dr Domenico (Mimmo) Parisi, National Strategic Planning & Analysis Research Center, Mississippi State University

FUNDING

National Science Foundation (NSF) – award no. 1238336

CONTACT

Dr Nan Niu
Principal Investigator

Department of Electrical Engineering and Computing Systems
University of Cincinnati
PO Box 210030
Cincinnati, Ohio
45221-0030
USA

T +1 513 556 0051
E nan.niu@uc.edu

www.nsf.gov/awardsearch/showAward?AWD_ID=1238336

http://homepages.uc.edu/~niunn/

DR NAN NIU is an assistant professor of computer science within the Department of Electrical Engineering and Computing Systems at the University of Cincinnati. He was recently recognised by an NSF CAREER Award that spans over the next five years. He received his BS in Computer Science and Engineering from the Beijing Institute of Technology and his MS in Computing Science from the University of Alberta, in 1999 and 2004, respectively. He received his PhD degree in Computer Science from the University of Toronto in 2009, under the supervision of Professor Steve M Easterbrook. Most recently, he was an assistant professor in the Department of Computer Science and Engineering at Mississippi State University.

cluster of information that a developer can access. As the code base may not be clustered already, clustering algorithms could be used to bring related data together, thus facilitating code navigation for software developers by reshaping the information environment. The challenge, once the source code has been rearranged to create a more favourable code navigation environment, is to assess the quality of clustering the software into 'code patches' and determine whether this has been profitable for the software developers.

Niu's findings indicate there was some 'patchiness' within software, drawing a parallel between code navigation and foraging theory. Clustering was also shown to be useful in terms of grouping the quality of 'traceability links', the links between software requirements and the rationale behind it, and information needed for that requirement to be implemented (eg. code). The researchers also demonstrated that semantic relatedness methods can be used to reshape the information foraging environment, enabling software developers to traverse it in a more principled fashion.

HUMAN TASK, HUMAN LIMITATIONS

Niu's recent studies continue on from previous work linking evolutionary psychology to software development. Building on his EAGER-funded research, a new NSF CAREER award is allowing Niu to work on linking the behaviours of software developers to their social information foraging, learning and co-creation. "My CAREER project aims to quantitatively characterise the intertwining relationship of the solo and social levels in software engineering, uncover the essential limits of the relationship, and create optimal solutions within or even beyond those limits," he explains.

Looking at software engineering problems from an evolutionary-ecological standpoint is important, as the human software developers, who are subject to their past evolution, are potentially exacerbating these issues. "Software engineering, long recognised as a human activity, is challenged by human limitations," Niu states. "To overcome human-centred challenges such as code navigation, we must gain a new and improved understanding about the process itself." Better understanding of these processes will provide an insight into how the work of software developers can be better supported, thus increasing developer productivity.

A FUTURE ELECTRONIC ENVIRONMENT

The impacts of the research carried out by Niu's group are far-reaching. The researchers understand that information foraging requires an interaction between the user and the information environment, and by employing biological and psychological principles they acknowledge that understanding this interaction can help improve information environments. Such research could enhance the development of tools available to programmers, thereby facilitating code navigation by professional programmers as well as novice software engineers learning how to navigate an unknown code base. The team's findings may kick-start the development of applications that can favourably alter the information environment to make everyday code navigation and software maintenance easier. Providing the tools to allow software developers to adapt their electronic environment to suit human behavioural and information foraging patterns may ultimately lead the way into a more economic infrastructure for software engineering.

> The results from the groups' tests and analyses of topical locality indicated that [...] a parallel can be drawn between code navigation and foraging theory

UNIVERSITY OF Cincinnati