

Studying the Social Aspects of Software Development

Nan Niu

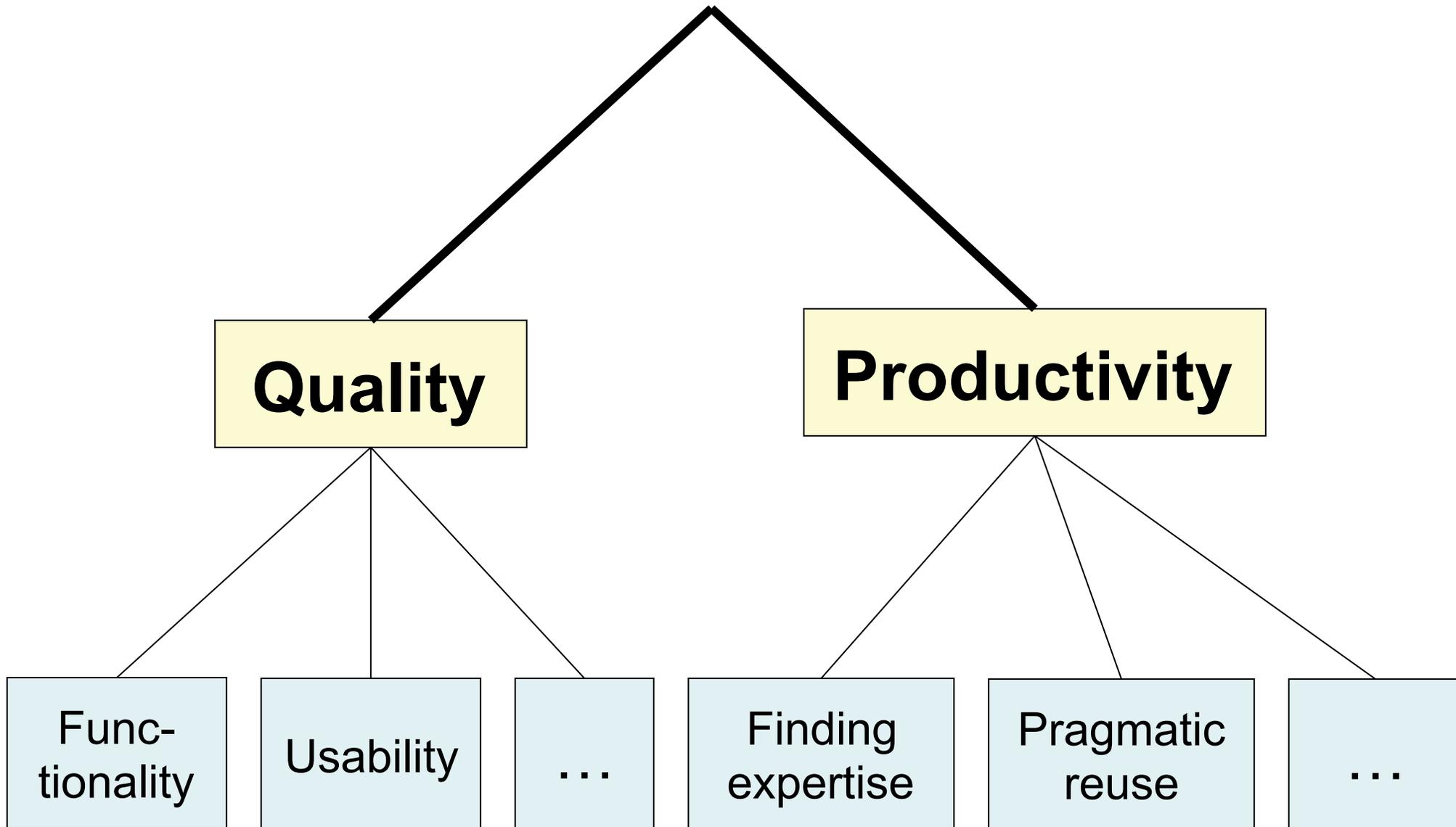
Department of Electrical Eng. & Computing Systems

University of Cincinnati

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

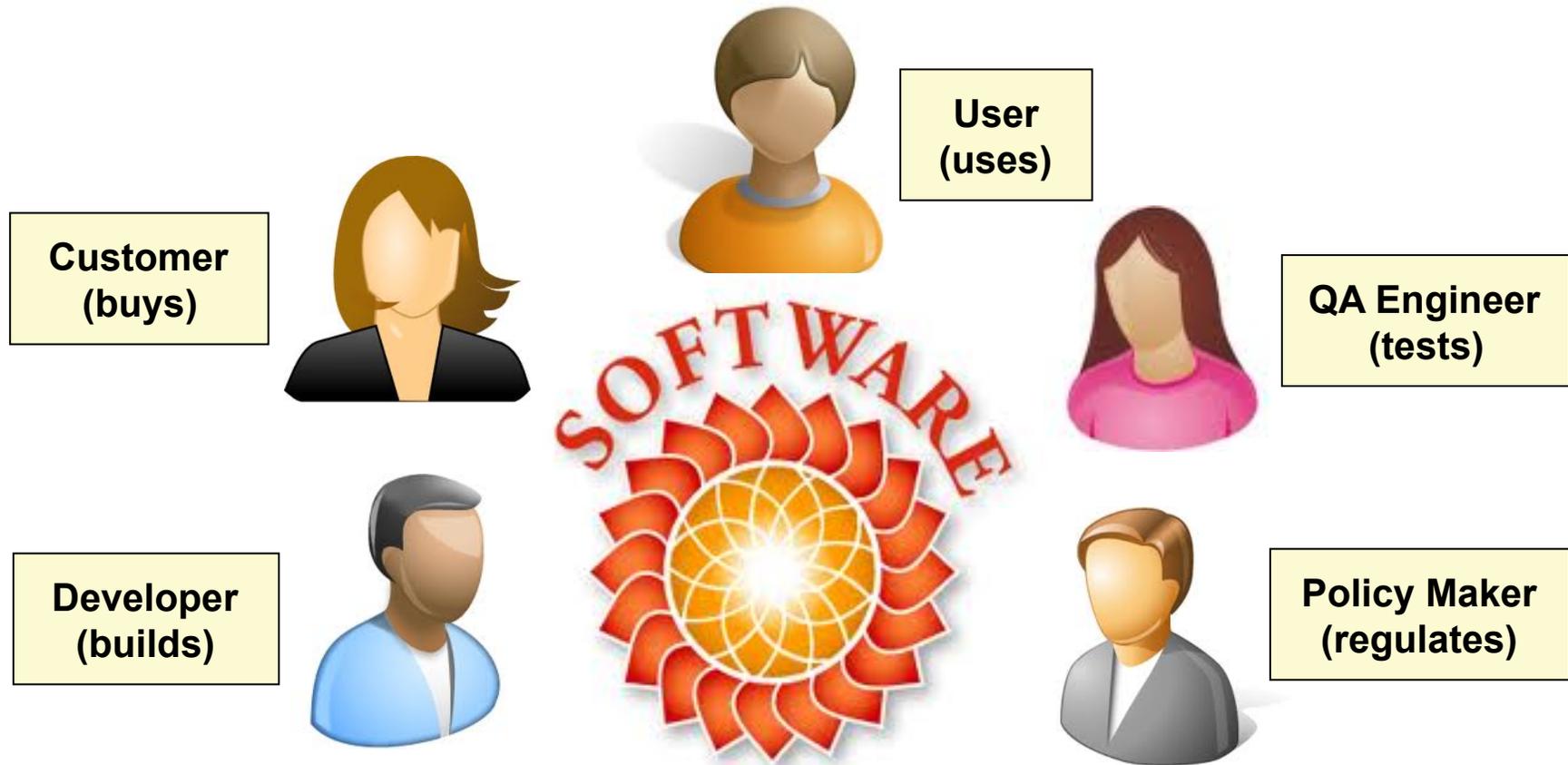
December 5, 2016 @ DIANA/HEP

Software Engineering



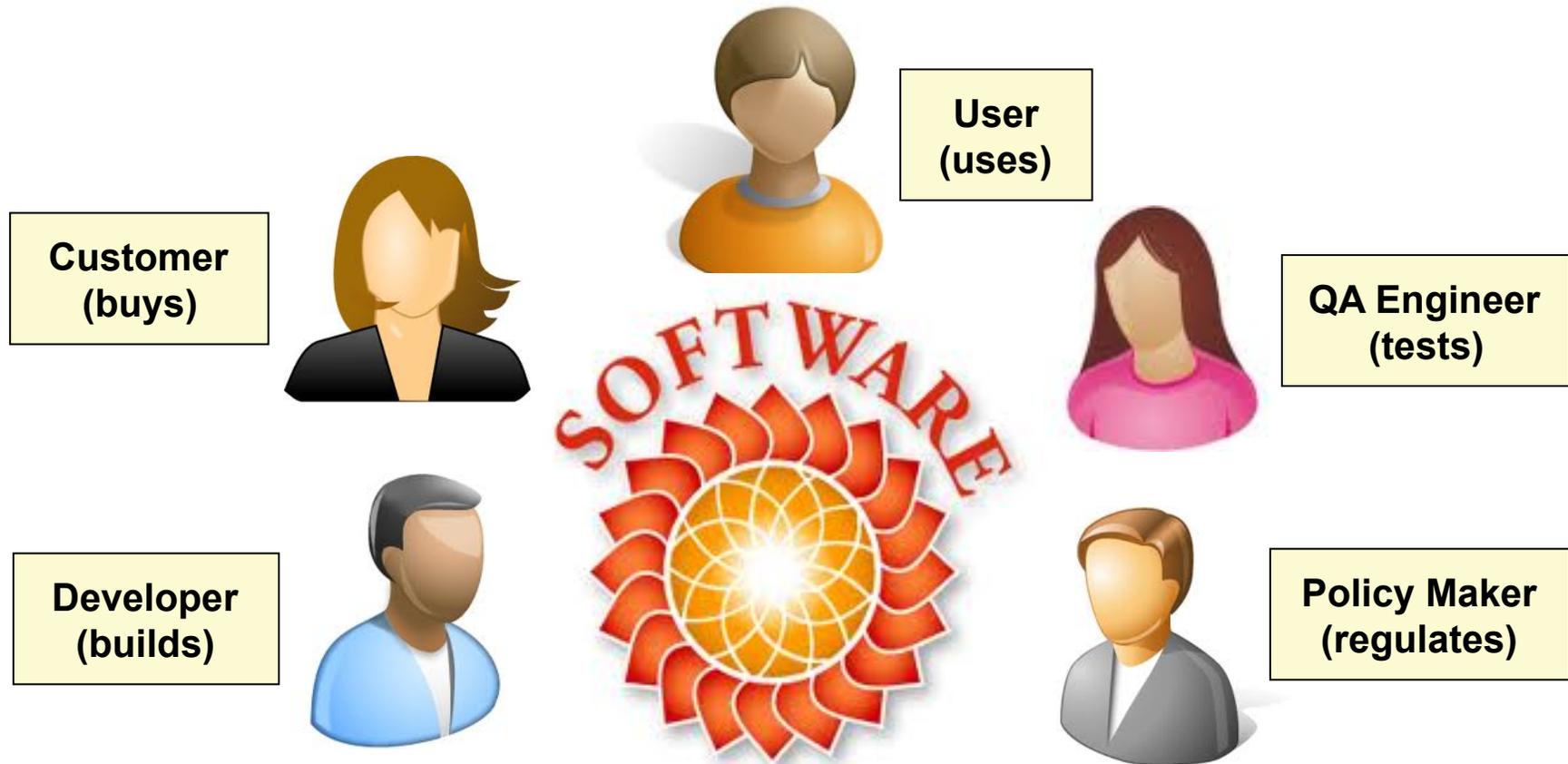
Software \approx Peopleware

[DeMarco and Lister, 1987]



Software \approx Peopleware

[DeMarco and Lister, 1987]



“Software is not limited by physics, but by properties of people.”

Ralph Johnson

Human as User

Templates

[Create & Edit](#)

Design

[Preferences](#)

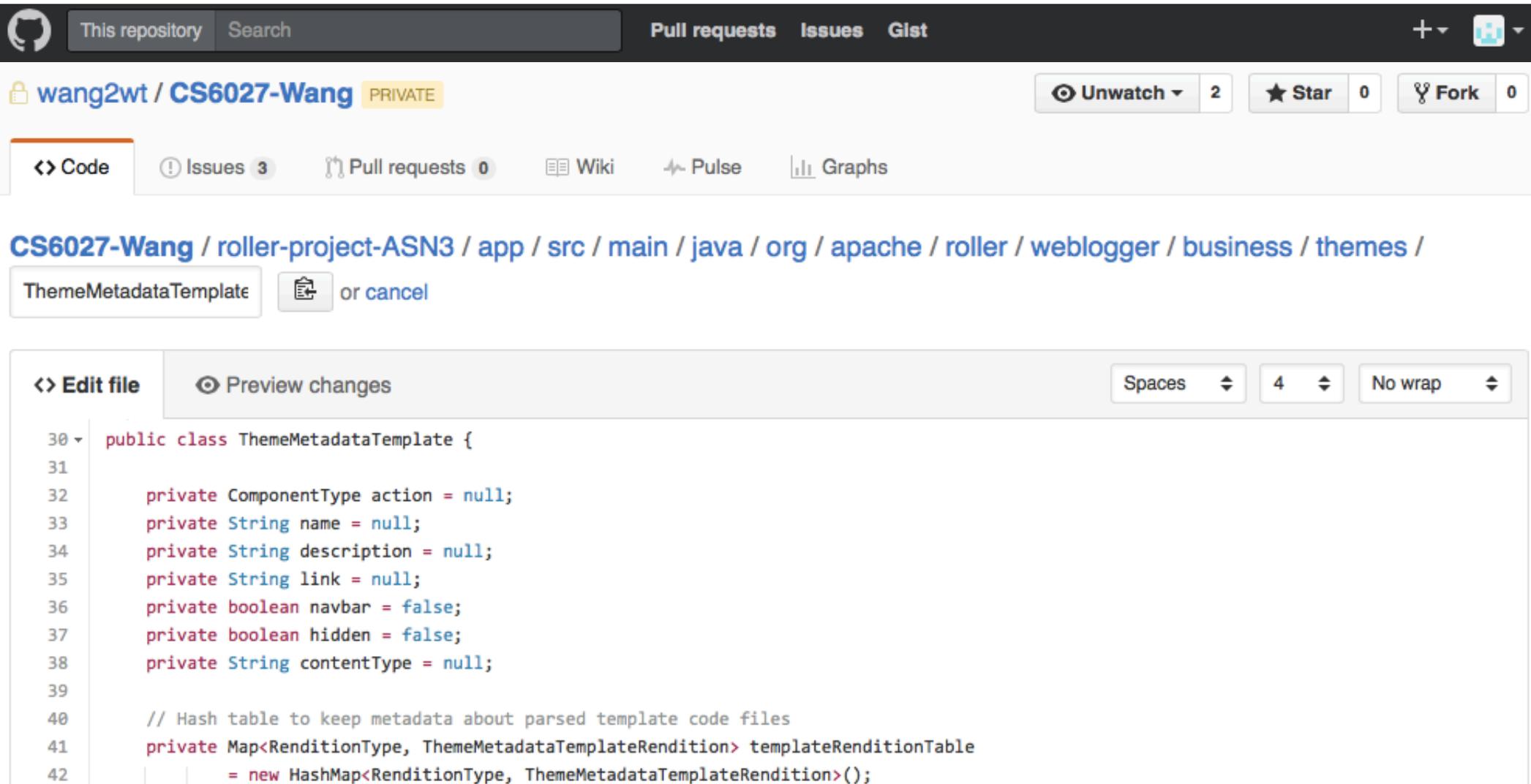
[Theme](#) | [Stylesheet](#) | **Templates**

Edit template **Archives** in weblog **admin**

You can edit this template to change what it generates. Refer the the Roller Template Guide for comfortable with HTML, you might want to leave your templates alone.

Name	<input type="text" value="Archives"/>
Edit Template	<input type="text" value="CUSTOM"/>
Link	<input type="text" value="archives"/> http://localhost:8080/roller/admin/page/archives [launch]
Description	<input type="text" value="Calendar browser and links to latest entries"/>

Human as Builder



The screenshot shows a GitHub repository page for 'wang2wt / CS6027-Wang'. The repository is private and has 2 watchers, 0 stars, and 0 forks. The current file being viewed is 'ThemeMetadataTemplate' in the path 'CS6027-Wang / roller-project-ASN3 / app / src / main / java / org / apache / roller / weblogger / business / themes /'. The code editor shows the following Java code:

```
30 public class ThemeMetadataTemplate {
31
32     private ComponentType action = null;
33     private String name = null;
34     private String description = null;
35     private String link = null;
36     private boolean navbar = false;
37     private boolean hidden = false;
38     private String contentType = null;
39
40     // Hash table to keep metadata about parsed template code files
41     private Map<RenditionType, ThemeMetadataTemplateRendition> templateRenditionTable
42     = new HashMap<RenditionType, ThemeMetadataTemplateRendition>();
```

What is Engineering?

- ⇒ Creating cost-effective solutions
 - ⇒ Not just about solving problems, but doing so with economical use of all resources
- ⇒ by applying scientific knowledge
 - ⇒ Solving problems in a particular way: by applying science, mathematics, and design analysis
- ⇒ to build things
 - ⇒ Emphasizing the solutions, which are usually tangible artifacts
- ⇒ in the service of mankind
 - ⇒ Not only serving the immediate customer, but eng. also develops technology and expertise supporting the society

What is Engineering?

⇒ Creating cost-effective solutions

⇒ Not just about solving problems, but doing so with economical use of all resources

by applying scientific knowledge

Let's look at Computer Science

⇒ to build things

⇒ Emphasizing the solutions, which are usually tangible artifacts

⇒ in the service of mankind

⇒ Not only serving the immediate customer, but eng. also develops technology and expertise supporting the society

Building Blocks for Software

- ⇒ Church-Turing Thesis: Any Turing-complete machine can compute anything that is computable.
- ⇒ Implies that code running on any computer can (theoretically) fulfill any (computable) functional requirements.
- ⇒ Problem: Within the space of what's computable, limitations come from our own limited capacities
 - ⇒ What should be the requirements?
 - ⇒ What languages & abstractions can we dream up?
 - ⇒ What are our limitations and how can we compensate for them?
 - ⇒ How can we act together in a coordinated way?

What is Engineering?

⇒ Creating cost-effective solutions

⇒ Not just about solving problems, but doing so with economical use of all resources

by applying scientific knowledge

My view: “a human-centric perspective”

⇒ to build things

⇒ Emphasizing the solutions, which are usually tangible artifacts

⇒ in the service of mankind

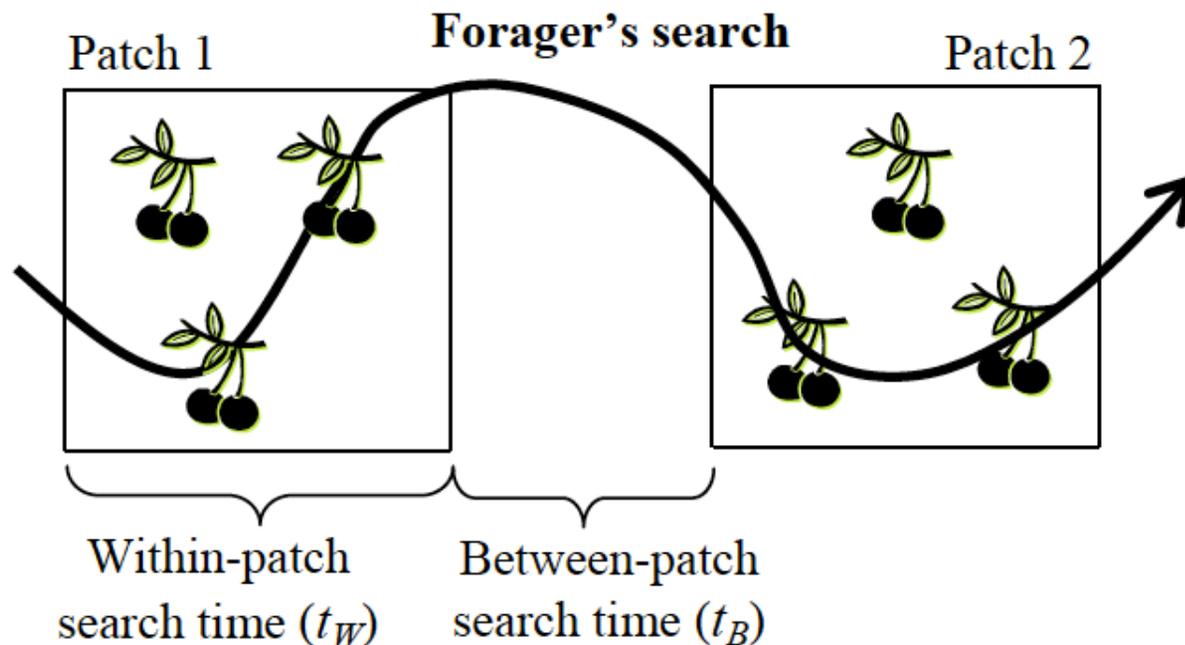
⇒ Not only serving the immediate customer, but eng. also develops technology and expertise supporting the society

Optimal Foraging Theory

[Stephens and Krebs, 1986]

⇒ Basics

- ⇒ Animals adapt, among other reasons, to increase their rate of energy intake
- ⇒ Optimality here refers to the strategy that maximizes the gain per unit cost



$$R = \frac{G}{t_B + t_W}$$

Patchy Environment

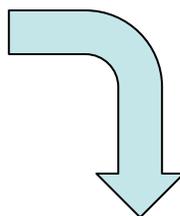
Social Foraging Theory

[Giraldeau and Caraco, 2000]

⇒ Hints are exchanged in group foraging

⇒ e.g., individual performance affected by group size

$$R = \frac{G}{t_B + t_W}$$

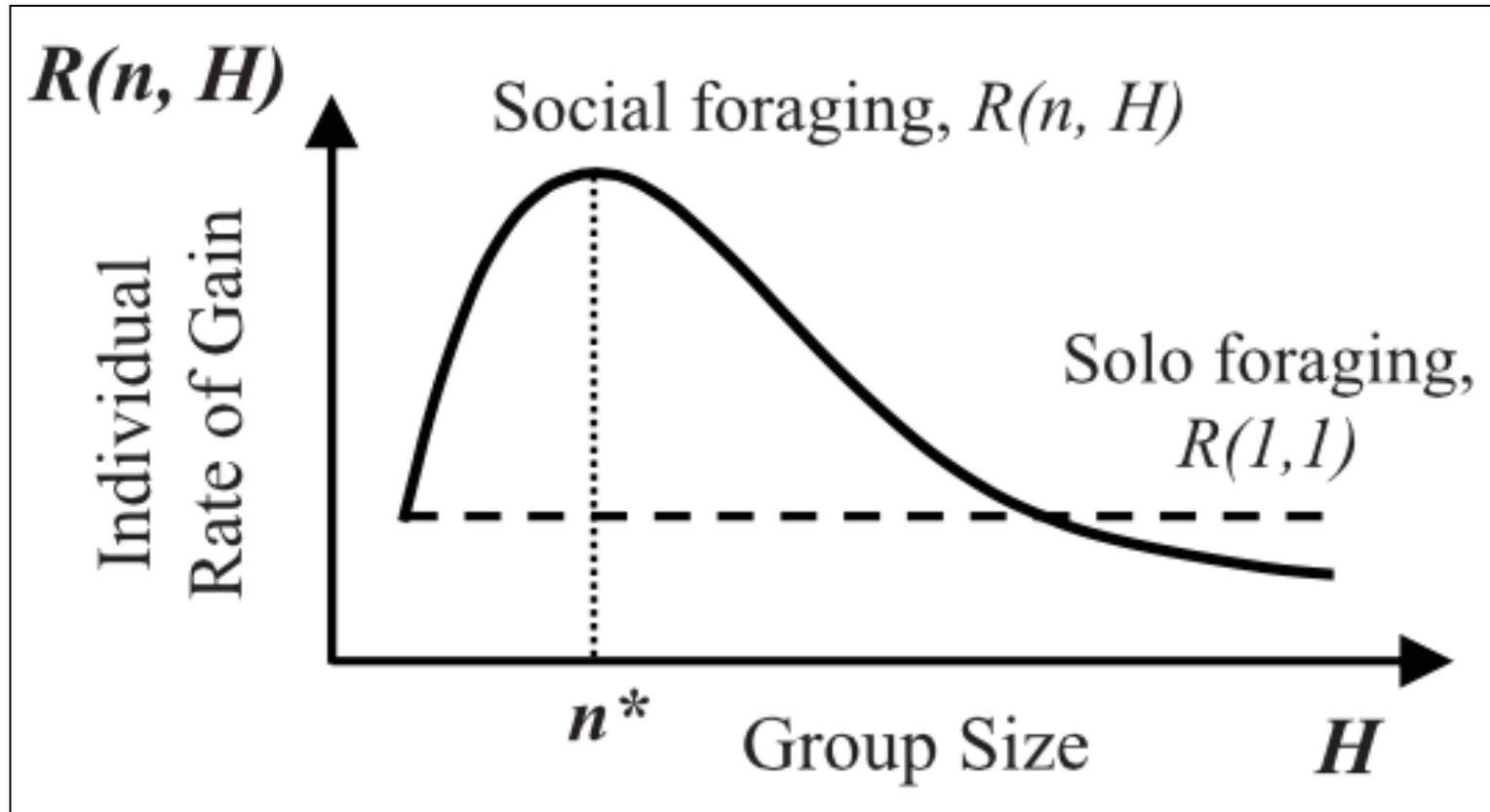


$$\begin{aligned} R(n, H) &= \frac{G/n}{t_I + t_B + t_W} = \frac{G/n}{\frac{1}{n \cdot \lambda(n)} + \frac{\lambda(H)}{n \cdot \lambda(n)} + \frac{\tau(n)}{n \cdot \lambda(n)}} \\ &= \frac{\lambda(n) \cdot G}{1 + \lambda(H) + \tau(n)}. \end{aligned}$$

Social Foraging Theory

[Giraldeau and Caraco, 2000]

- ⇒ Hints are exchanged in group foraging
 - ⇒ e.g., individual performance affected by group size



Group Size for OSS Change Tasks

[Bhowmik *et al.*, 2016]

	Firefox	Mylyn
Domain	Web browsing	Task management
Source	mozilla.org/firefox	eclipse.org/mylyn
Programming Language	C/C++, JavaScript	Java
# of Source Code Files	1968 (C/C++)	2321
Analysis Begin Date	2004-07-06	2006-12-05
Analysis End Date	2011-06-20	2011-02-28
# of Unique Developer IDs	2569	149
# of Completed Tasks	2878	1898
# of Commits	18538	4908

Main Results

⇒ H1: There's no difference between n^* (optimal group size) and n (actual group size).

⇒ Rejected.

⇒ H2: The closer n is to n^* , the more productive the group members are.

⇒ Supported.

⇒ H3: As the OSS project evolves, the group size becomes more optimal.

⇒ Rejected.

⇒ H4: Developer's solo-social changes in evolution lead to productivity gain at the group level.

⇒ Initial evidence (by looking at high-profile developers)

Where's the Theory for Software Engineering?

Pontus Johnson, Mathias Ekstedt, and Ivar Jacobson

- ⇒ Description
- ⇒ Explanation
- ⇒ Prediction
- ⇒ Action



Where's the Theory for Software Engineering?

Pontus Johnson, Mathias Ekstedt, and Ivar Jacobson

- ⇒ Description
- ⇒ Explanation
- ⇒ Prediction
- ⇒ Action



Take-Away Messages

⇒ Software \approx Peopleware

⇒ Human as user & human as builder

⇒ Software engineering goes social

⇒ How to build the software to enable its users to behave nicely?

⇒ Foraging theory

⇒ What info. to hunt? How long to seek & handle it?
What to exchange & what not? To whom and how widely? ...

Studying the Social Aspects of Software Development

Nan Niu

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

Thank You!

