Software Navigation

Thomas Zimmermann
Oral exams

☐ All slides
☐ List of ten papers
☐ Guest lectures
☐ Your course notes
Concept assignment problem

Identify how high-level concepts are implemented in source code.

For software modification: understand the concepts that are associated to the task.
Software navigation
Landmarks
Landmarks
Beacons
Beacons
Landsmarks and beacons

```java
public static void bubbleSort(int[] array) {
    for (int i = array.length - 1; i > 0; i--) {
        for (int j = 0; j < i; j++) {
            if (array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}
```

What would be a landmark and what would be a beacon?
Waypoints

LONGITUDE: 7.045454746632007
LATITUDE: 49.25755151296508
Learning an environment

- Landmark knowledge
- Route knowledge
- Survey knowledge
Route knowledge

Route knowledge allows to navigate from one point to another.
Route knowledge allows to navigate from one point to another.
Route knowledge allows to navigate from one point to another.
Route knowledge

Route knowledge allows to navigate from one point to another.

Navigational actions

Landmarks
Route knowledge
Route knowledge
Survey knowledge is analogous to a map. except that it allows the navigator to adopt the most convenient perspective. Spatial properties, but distortions can occur.
Elements in cognitive maps

**Paths:** channels for navigator movement  
street, canal, transit line

**Edges:** indicate district limits  
fence, river

**Districts:** reference points  
neighborhood

**Nodes:** focal points for travel  
town square, buildings

**Landmarks:** “special” reference points  
statue, tower
What makes a landmark?
What makes a landmark?

Memorizable

Memory for location
What makes a landmark?

- Significant height
- Visible
- **Memorizable**
- Complex shape
- Large, visible signs

Memory for location
What makes a landmark?

- Significant height
- Memorizable
- Complex shape
- Large, visible signs
- Bright exterior
- Unique color, texture
- Memory for location
- Visible
- Expensive materials
Navigation in codespace

Many similarities to physical world:

Manifestation (can be viewed and manipulated)

Identifiable constructs (objects)

Various levels of abstraction (hierarchies)

Direction

Distance (transit techniques)
Swiss army knife hypothesis
Swiss army knife hypothesis

Common subset of skills is used in each world (codespace + physical world).
Swiss army knife hypothesis

Common subset of skills is used in each world (codespace + physical world).

This subset is augmented and modified to deal with the unique aspects.
Software navigation

Where do I start?
Software navigation

Where do I start?

Where am I?

(Self location)
Software navigation

Where do I start?

When's the next turn?
(Where do I have to look/change next?)

(Self location)
Where am I?
Software navigation

Where do I start?

When's the next turn?
(Where do I have to look/change next?)

Where am I?
(Self location)

Have I reached?
(Have I completed my task?)
NavTracks

Keep track of navigation history to support browsing

Singer et al. (ICSM 2005)
Requirements

**Non-disruptive.** Should not disrupt nor affect the performance of developers.

**Current patterns.** Prefer locally available data over historical traces (e.g., version archives).

**Approximate but efficient.** Like Google, be efficient and accurate most of the time.
Architecture

1. Navigation Event
2. Event Director
3. Event Filter Chain
4. Association Engine
5. Recommendations

NavTracks Plugin

Association Repository
Record events

Architecture

1. Navigation Event
2. Event Director
3. Event Filter Chain
4. Association Engine
5. Recommendations
Architecture

Record events

Filter jitter and duplication
Architecture

1. Record events
2. Filter jitter and duplication
3. Compute associations
Associations

Event Step

Step 1:

Event Window
(size n=4)

Associations Detected
(minimum cycle length k=3)

No Cycles
Associations

Event Step | Event Window (size n=4) | Associations Detected (minimum cycle length k=3)
--- | --- | ---
Step 1: | No Cycles |  
Step 2: | No Cycles |  

- Step 1: Event Window with one event labeled A.
- Step 2: Event Window with two events labeled A and C.
# Associations

<table>
<thead>
<tr>
<th>Event Step</th>
<th>Event Window (size n=4)</th>
<th>Associations Detected (minimum cycle length k=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1:</td>
<td>A</td>
<td>No Cycles</td>
</tr>
<tr>
<td>Step 2:</td>
<td>A, C</td>
<td>No Cycles</td>
</tr>
<tr>
<td>Step 3:</td>
<td>A, C, B</td>
<td>No Cycles</td>
</tr>
</tbody>
</table>
Associations

Event Step

Event Window
(size n=4)

Associations Detected
(minimum cycle length k=3)

Step 3:

A  C  B

No Cycles
Associations

Event Step | Event Window (size n=4) | Associations Detected (minimum cycle length k=3)
---|---|---
Step 3: | A C B | No Cycles
Step 4: | A C B A | AC, AB
Associations

**Event Step** | **Event Window** | **Associations Detected**
---|---|---
**Step 3:** |  | No Cycles
| A | C | B |  
**Step 4:** |  | AC, AB
| A | C | B | A |  
**Step 5:** |  | BA
| A | C | B | A | B |
Continuous evaluation

Continuous evaluation and training process on the events of three developers: 36%, 35%, 16%

<table>
<thead>
<tr>
<th>Event Class</th>
<th>D3</th>
<th>D1</th>
<th>D2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-6 occurrences</td>
<td>25%</td>
<td>23%</td>
<td>17%</td>
<td>22%</td>
</tr>
<tr>
<td>7-11 occurrences</td>
<td>52%</td>
<td>50%</td>
<td>32%</td>
<td>45%</td>
</tr>
<tr>
<td>11-16 occurrences</td>
<td>46%</td>
<td>51%</td>
<td>31%</td>
<td>43%</td>
</tr>
<tr>
<td>12-21 occurrences</td>
<td>60%</td>
<td>31%</td>
<td></td>
<td>45%</td>
</tr>
<tr>
<td>&gt; 21 occurrences</td>
<td>43%</td>
<td>43%</td>
<td></td>
<td>43%</td>
</tr>
</tbody>
</table>
User study

Experiences and feedback from 5 developers.

**Newcomer & New system development use.** Tool provides a memory aid for related files.

**Wanderer use.** Tool did not help for refactoring (repetitive copy, paste, and modify cycle).

**Navigation use.** Tool supported contextually different navigation paths.
Team Tracks

Team Tracks is used at Microsoft

Team Tracks = NavTracks on Teams

DeLine et al. (VL/HCC 2005, SOFTVIS 2005)
Navigation and changes
Navigation and changes
Navigation and changes

interesting

navigated locations
Navigation and changes

interesting

changed locations

navigated locations
Navigation and changes

Logging  NavTracks
TeamTracks
Navigation and changes

Logging  NavTracks  TeamTracks

Change History

Order is lost
TagSea

---

```java
/**
 * Simple utility class for TagSEA.
 * @author Del Myers
 */

//@tag tagsea utilities: basic utilities for TagSEA
public class TagSEAUUtils {

/**
 * Return code for a valid tag name.
 */
 public static final int TAG_NAME_VALID = 0;

/**
 * Return code for a tag name with a syntax error.
 */
 public static final int TAG_NAME_SYNTAX_ERROR = 1;
```
TagSea = Waypoints + Social tagging
Suade

Suade addresses the concept assignment problem.

Guide developers towards interesting sections of code by analyzing the topology of a program’s structural dependencies.

Robillard (ACM Distinguished Paper @ ESEC/FSE 2005)
The approach

Set of interest

fuzzy set
The approach

Set of interest -> Suade -> Suggestion set

fuzzy set  ->  fuzzy set
The approach

Set of interest $\xrightarrow{fuzzy\ set} \text{Suade} \xrightarrow{Structural\ dependencies} \text{Suggestion set}^{fuzzy\ set}$
The approach

Set of interest \[\text{fuzzy set}\] → Suade → Suggestion set \[\text{fuzzy set}\]

Structural dependencies

Specifi city Reinforcement
Specificity

Set of interest

A ⊆ B

B has no other calls!

⇒ B is specific to the set of interest
Reinforcement

Set of interest

⇒ B is reinforced
Example for specificity

Distinguish between direct and transpose relations.
Multiple relations

1: **Param:** $\bar{I}$: Set of interest
2: **Param:** $L$: Set of relation types to analyze
3: **Var:** $\bar{S} = \{\}$: Suggestion set
4: **Var:** $\bar{T} = \{\}$: Temporary set
5: **for all** $l \in L$ **do**
6: $\bar{T} = \text{analyzeRelation}(l, \bar{I})$
7: $\bar{S} = \bar{S} \cup \bar{T}$
8: **end for**
9: **return** $\bar{S}$
Multiple relations

1: **Param:** $\bar{I}$: Set of interest
2: **Param:** $L$: Set of relation types to analyze
3: **Var:** $\bar{S} = \{\}$: Suggestion set
4: **Var:** $\bar{T} = \{\}$: Temporary set
5: **for all** $l \in L$ **do**
6: $\bar{T} = \text{analyzeRelation}(l, \bar{I})$
7: $\bar{S} = \bar{S} \cup \bar{T}$
8: **end for**
9: **return** $\bar{S}$
Multiple relations

1: **Param:** \( \bar{I} \): Set of interest
2: **Param:** \( \bar{L} \): Set of relation types to analyze
3: **Var:** \( \bar{S} = \{ \} \): Suggestion set
4: **Var:** \( \bar{T} = \{ \} \): Temporary set
5: **for all** \( l \in \bar{L} \) **do**
6: \( \bar{T} = \text{analyzeRelation}(l, \bar{I}) \)
7: \( \bar{S} = \bar{S} \cup \bar{T} \)
8: **end for**
9: **return** \( \bar{S} \)

**Suade uses a special union operator!!!**

calls, called by, accesses, accessed by
## Analyze relations

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Assumes: $P = (E, R)$: A program</td>
</tr>
<tr>
<td>2</td>
<td>Param: $\bar{I} = {(x, \mu_{\bar{I}}(x)) \mid x \in E}$: Set of interest</td>
</tr>
<tr>
<td>3</td>
<td>Param: $l \in {r \mid (r, e_1, e_2) \in R}$: Relation type to analyze</td>
</tr>
<tr>
<td>4</td>
<td>Param: $0 \leq \alpha \leq 1$: A calibration parameter</td>
</tr>
<tr>
<td>5</td>
<td>Var: $S_b \in E$: A (crisp) set of program elements</td>
</tr>
<tr>
<td>6</td>
<td>Var: $S_f \in E$: A (crisp) set of program elements</td>
</tr>
<tr>
<td>7</td>
<td>Var: $\bar{Z}$: The (fuzzy) set to be returned</td>
</tr>
</tbody>
</table>
Analyze relations

8: for all $x \in \bar{I}$ do
9: \[ S_b = \{ y \mid (l, x, y) \in R \} \]
10: for all $s \in S_b$ do
11: \[ \text{if } s \notin \bar{I} \text{ then} \]
12: \[ S_f = \{ y \mid (l^\top, s, y) \in R \} \]
13: \[ \bar{Z} = \bar{Z} \cup \{(s, (\mu_{\bar{I}}(x) \cdot \deg(1, S_b, \bar{I}) \cdot \deg(0, S_f, \bar{I}))^\alpha)\} \]
14: \[ \text{end if} \]
15: \[ \text{end for} \]
16: \[ \text{end for} \]
17: return $\bar{Z}$
Analyze relations

8: for all $x \in \bar{I}$ do
9:   $S_b = \{y \mid (l, x, y) \in R\}$
10: for all $s \in S_b$ do
11:   if $s \notin \bar{I}$ then
12:       $S_f = \{y \mid (l^\top, s, y) \in R\}$
13:       $\bar{Z} = \bar{Z} \cup \{(s, (\mu_{\bar{I}}(x) \cdot \text{deg}(1, S_b, \bar{I}) \cdot \text{deg}(0, S_f, \bar{I}))^\alpha)\}$
14:   end if
15: end for
16: end for
17: return $\bar{Z}$

direct degree
Analyze relations

8: for all \( x \in \bar{I} \) do
9:     \( S_b = \{ y \mid (l, x, y) \in R \} \)
10:    for all \( s \in S_b \) do
11:        if \( s \notin \bar{I} \) then
12:            \( S_f = \{ y \mid (l^\top, s, y) \in R \} \)
13:            \( \bar{Z} = \bar{Z} \cup \{ (s, (\mu_{\bar{I}}(x)) \cdot deg(1, S_b, \bar{I}) \cdot deg(0, S_f, \bar{I})^\alpha) \} \)
14:        end if
15:    end for
16: end for
17: return \( \bar{Z} \)
Analyze relations

8: for all $x \in \bar{I}$ do
9: \[ S_b = \{ y \mid (l, x, y) \in R \} \]
10: for all $s \in S_b$ do
11: \[ \text{if } s \notin \bar{I} \text{ then} \]
12: \[ S_f = \{ y \mid (l^\top, s, y) \in R \} \]
13: \[ \bar{Z} = \bar{Z} \cup \{ (s, (\mu_{\bar{I}}(x) \cdot \deg(1, S_b, \bar{I}) \cdot \deg(0, S_f, \bar{I})^\alpha) \} \]
14: end if
15: end for
16: end for
17: return $\bar{Z}$

\[
\deg(t, U, \bar{V}) = \frac{t + |U \cap \bar{V}|}{|U|}
\]
The Suade plug-in

Set of interest

Suggestion set
Quantitative evaluation

Size of suggestion sets.
On average 4.8 suggestions for a singleton set.
Largest set contained 86 elements.

Parameterization of the algorithm.
Most useful values of $\alpha$ for $0.1 < \alpha < 0.7$.
Degree should be viewed as relative measure.
Case study

File allocation concern of the Azureus system.
Set of interest: 4 members of DiskManagerImpl
Suggestion set: additional 54 elements

Evaluation by two experts (working as a team). Is this element relevant to understand how files are allocated? ⇒ “Yes”, “No”, and “Somewhat”
Case study
Mylar

Provide a task-focused UI that reduces information overload by monitoring the work activity of developers.

Kersten and Murphy (AOSD 2005, FSE 2006)
Information Overload

Hard to see the forest through the trees.

Repetitive scrolling, searching, navigating

taken from Mylar: a Task Focused UI for Eclipse (EclipseCon talk, Mar 2006)
taken from Mylar: a Task Focused UI for Eclipse (EclipseCon talk, Mar 2006)
Mylar
See only what you’re working on.

Aluminized film used to avoid blindness when staring at an eclipse.
Task Focused UI to avoid information blindness when staring at Eclipse.

*taken from Mylar: a Task Focused UI for Eclipse (EclipseCon talk, Mar 2006)*
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

**Interaction history**
- InteractionEvent stream
- Origin, handle, type, date

**Context (Core)**
- Degree-of-interest graph
Mylar’s context model

**Interaction history**
- InteractionEvent stream
- Origin, handle, type, date

**Context (Core)**
- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

Interaction history

- InteractionEvent stream
- Origin, handle, type, date

Context (Core)

- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph

taken from Mylar: a Task Focused UI for Eclipse (EclipseCon talk, Mar 2006)
Mylar’s context model

**Interaction history**
- InteractionEvent stream
- Origin, handle, type, date

**Context (Core)**
- Degree-of-interest graph
- Degree-of-separation scope

*taken from Mylar: a Task Focused UI for Eclipse (EclipseCon talk, Mar 2006)*
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
- Degree-of-separation scope
- Scaling factors

_taken_from_Mylar_a_Task_Focused_UI_for_Eclipse_EclipseCon_talk_Mar_2006_
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
- Degree-of-separation scope
- Scaling factors

Projections (UI)
- Views, editors, files

taken from *Mylar: a Task Focused UI for Eclipse* (EclipseCon talk, Mar 2006)
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
- Degree-of-separation scope
- Scaling factors

Projections (UI)
- Views, editors, files
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
- Degree-of-separation scope
- Scaling factors

Projections (UI)
- Views, editors, files

taken from *Mylar: a Task Focused UI for Eclipse* (EclipseCon talk, Mar 2006)
Mylar’s context model

Interaction history
- InteractionEvent stream
- Origin, handle, type, date

Context (Core)
- Degree-of-interest graph
- Degree-of-separation scope
- Scaling factors

Projections (UI)
- Views, editors, files
Summary

- Landmarks, beacons, and waypoints
- Route knowledge vs. survey knowledge
- Support software navigation by using
  - past navigation traces (NavTracks)
  - structural dependencies (Suade)
- Task-focus reduces information overload (Mylar)
Design guidelines for landmarks to support navigation in virtual environments. Norman Vinson.


Automatic generation of suggestions for program investigation. Martin P. Robillard.

Using task context to improve programmer productivity. Mik Kersten, Gail C. Murphy.