Sketches in Software Engineering

Sebastian Baltes
University of Trier, Germany

@s_baltes
s.baltes@uni-trier.de

November 13, 2015
Outline

Past research:

"Sketches and Diagrams in Practice"

"Linking Sketches and Diagrams to Source Code Artifacts"

"Navigate, Understand, Communicate: How Developers Locate Performance Bugs"

Future Research:

Developing a Visual Literacy Curriculum for Developers?
Sketches and Diagrams in Practice

Sebastian Baltes and Stephan Diehl
Sketches and Diagrams

informal  Sketch  Diagram  formal
Past studies: Sketches and diagrams important in daily work of software developers

**Purpose:** Understanding, designing, communicating

[Cherubini07]

**Depict mental model of software**

[LaToza06]

**Medium:** Whiteboard, paper, computer

[Cherubini07, Walny11]

**Psychology:** Sketching augments information processing, sketches are sources of creativity

[Goldschmidt03, Tversky03]

Teams **improvise** representations, sketches/diagrams often **informal**

[Dekel07, Petre13]
Our Goal

Existing studies:
- Concentrated on certain aspects
- Single companies
- Academic environment
- Some had small number of participants

Our goal: Thorough description of how sketches and diagrams are used in software engineering practice

Better tool support for integrating sketches and diagrams into software development process
Research Design

How to describe sketches and diagrams in SE practice?
Research Design

Existing Studies
- purpose, media, formality, etc.

Exploratory Field Study
- 3 companies
- 13 developers
- 47 sketches

11 Dimensions
- Revision
- Lifespan
- Effort
- Purpose
- Contributors
- Context
- Relation to Source Code
- Archiving
- Medium
- Formality
- UML-Elements

Online Survey
Online Survey

• **Target population:** "software practitioners"

• **Concise:**
  • ~10 minutes to complete
  • 28 questions, 15 about last sketch

• **Recruiting:**
  • Network of colleagues and contacts
  • Social networks
  • IRC channels and online communities
  • Directly contacted software companies
  • Article on major German IT news website
Participants

- n=394
- 32 countries
  - 54% Germany
  - 15% North America
- 52% software developers, 22% software architects
- Time spent developing software: 80% (median)
- Professional work experience: 10 years (median)
- Software projects from various application areas
Results
When did you create your last sketch/diagram?

When did you use the last sketch/diagram created by someone else?

- Looked at, modified, extended

Takeaway 1:
Creating own sketches and using sketches created by others are frequent tasks among software practitioners.
What medium did you use to create the sketch/diagram?

- **Whiteboard (40%)**
- **Paper (18%)**
- **Computer (39%)**
- **E-Whiteboard (1.5%)**

**Analog (58%)**

**Digital (42%)**

**Tablet (0.8%)**
The sketch/diagram helped me to...
(multiple answers possible)

...design a new architecture (52%)
...design new features (48%)
...explain an issue to someone else (46%)
...analyze requirements (45%)
...understand an issue (44%)
The sketch/diagram helped me to...
(multiple answers possible)

...design a new architecture (52%)
...design new features (48%)
...explain an issue to someone else (46%)
**...analyze requirements (45%)**
...understand an issue (44%)
Effort and Revision

How much effective work time went into the creation and revision of the sketch/diagram up to now?

- 68% less than 1 hour
- 93% less than 8 hours

Revision:
15% revised once, 74% multiple times

Takeaway 2:
Most sketches are created in less than one hour and are revised at least once.
Please try to estimate the lifespan of the sketch/diagram (how long did/will you use it)?

- 30% less than one day
- 65% at least several days
- 48% at least several weeks

**Takeaway 3:**
Almost half of the sketches are used for at least several weeks.
**Formality and UML**

**Formality:** Please try to specify the formality of your sketch/diagram.
(6-point Likert scale item (0-5) from "very informal" to "very formal")

**UML:** To which degree does the sketch/diagram contain UML elements?
(6-point Likert scale item (0-5) from "no UML elements" to "only UML elements")

![Formality and UML Diagram]

- **68% informal**
- **40% no UML**
- **48% some UML**
- **9% solely UML**

**Takeaway 4:**
The majority of sketches and diagrams are **informal**. If UML is used, it is often mixed with other notations.
**Relation to Source Code**

**Help self:** Do you think that the sketch/diagram could help you in the future to understand the related source code artifact(s)?

**Help others:** ... help other software developers ...

(6-point Likert scale item (0-5) from "It will definitely not help " to "It will definitely help")

Takeaway 5: About **half of the sketches are rated as helpful** to understand the related source code artifact(s) in the future.

52% and 47% Likert 4-5
Archiving

Three questions:

1. Has the sketch/diagram been archived or will it be archived?

58% archived

2. If the sketch has been archived or will be archived, why do you want to keep it?

3. If the sketch has not been archived and won't be archived, why do you not want to keep it?

- Coded answers (open coding)
- Grouped why/why not answers each in four categories
- One category for archiving practice
Archiving - Media

**Takeaway 6:**
Most digital sketches, but also more than one third of the analog sketches, **are archived**.

- **Analog**
  - Archived (38%)
  - Not archived (62%)

- **Digital**
  - Archived (94%)
  - Not archived (6%)

- **Media**
  - Whiteboard (40%)
  - Paper (18%)
  - Computer (39%)
  - Tablet (0.8%)
  - E-Whiteboard (1.5%)
If the sketch has been archived or will be archived, why do you want to keep it?

- **Documentation**
  - "It will be difficult to understand the code without the diagram."

- **Future Use**
  - "[The code] can be quickly understood due to the visual representation without hours of digging through complex source code."

- **Understanding**
  - "[The sketch] shows concepts that are not directly visible from code."

- **Visualization**
If the sketch has been archived or will be archived, why do you want to keep it?

**Takeaway 7:**

Sketches are kept, because they document software, visualize it, and support its understanding.
Archiving - Why not?

If the sketch has not been archived and won't be archived, why do you not want to keep it?

- Served its purpose
- Substituted
- Outdated
- Technical Issue

"I do want to keep the sketch, but I have no way to archive whiteboard drawings."

"In case there was an easy way to combine both, code [...] and sketch I might have thought about archiving it."

"There is no good option to keep the sketch together with source code."
Archiving – How?

- MediaWiki
- Subversion
- git
- JIRA
- Bugzilla
Conclusion

• **Software documentation** is frequently *poorly written* and out of date
  [Forward02, Lethbridge03]

• Sketches and diagrams could serve as a *supplement* to conventional documentation

• Software practitioners are *willing to keep* their sketches and diagrams

• **Better tool support needed** for archiving and retrieving sketches/diagrams related to source code artifacts

• Tools should support *evolution* of sketches/diagrams (and software)

Survey data and questionnaire available at:
  [http://st.uni-trier.de/survey-sketches](http://st.uni-trier.de/survey-sketches)
Our Goal

Existing studies:
- Concentrated on certain aspects
- Single companies
- Academic environment
- Some had small number of participants

Our goal: Thorough description of how sketches and diagrams are used in software engineering practice

Better tool support for integrating sketches and diagrams into software development process
Linking Sketches and Diagrams to Source Code Artifacts

Sebastian Baltes, Peter Schmitz, and Stephan Diehl
Linking Sketches and Diagrams to Source Code Artifacts

Video available online:
https://www.youtube.com/watch?v=3IuLKZx7Wbs
Future Work

- What **distinguishes** helpful from not **helpful sketches**?

- What **context information** is required to understand sketches later?

- Do (informal) visualizations for certain source code artifacts share **common characteristics**?

**Recommendations** on how to create, augment, or annotate sketches so that they can serve as valuable software documentation.
Navigate, Understand, Communicate: How Software Developers Locate Performance Bugs

Sebastian Baltes, Oliver Moseler, Fabian Beck, and Stephan Diehl
Summary

Objective:
Investigate how developers, when locating performance bugs:
• **Navigate** through the source code
• **Understand** the program
• **Communicate** detected issues

Method:
• **Qualitative** user study
• Observed **12 developers** fixing documented performance bugs in open source projects (Apache Commons Collections and Google Guava Libraries)
• Pair programming setting
• Interviews, data collection in IDE, recorded sketching
• Profiling and analysis tool (list and in-situ)
Objective:
Investigate how developers, when locating performance bugs:
- **Navigate** through the source code
- **Understand** the program

Method:
- Qualitative user study
- Observed 12 developers fixing documented performance bugs in open source projects (Apache Commons Collections and Google Guava Libraries)
- Pair programming setting
- Interviews, data collection in IDE, recorded sketching
- Profiling and analysis tool (list and in-situ)
Study Setup

Introductory Video and Slides

IDE

Camera

Microphone

Sketching
RQ2.2: Could sketches help to understand and communicate a performance bug?

→ Sketches visualizing **data structures** and **algorithms** turned out to be valuable for **externalizing and communicating** the comprehension process for complex bugs.

Dynamic Behavior:

Alternatives:
Results from cross-case analysis of interview answers:

"If and how much sketching occurs depends on the sketching experience of the developers." (4/6 teams)

"A common sketch vocabulary is needed in the team." (3/6 teams)

→ Many developers had problems to visually express their thoughts
A Visual Literacy Curriculum for Developers?
Visual Literacy

Term first coined 1969, many different definitions exist, e.g.:

"Visual literacy can be defined as a group of skills which enable an individual to understand and use visuals for intentionally communicating with others." (Ausburn and Ausburn, 1978)

"Visual literacy is the ability to understand (read) and use (write) images and to think and learn in terms of images, i.e., to think visually." (Hortin, 1983)
Future Work

• **Research** in visual literacy **often focuses on reading** and interpreting visuals or visualizations

• Not much work on "**production literacy**" (Messaris, 1994)

**Our goal:** Develop a lightweight curriculum to teach software developers how to produce simple visuals for communicating their ideas.

**NOT UML!**
**Example**

**the 5 Basic Elements**

- Circle
- Square
- Triangle
- Line
- Dot

**EVERYTHING YOU WANT TO DRAW CAN BE CREATED WITH THESE 5 ELEMENTS.**

Can you identify the 5 basic elements in these simple drawings?

**ONCE YOU REALIZE** how the objects around you are made from these 5 elements, it becomes easier to draw all sorts of things.
Inspirations

- **Psychology**: Research on perception, visual thinking, sketching, etc.
- **Semiotics**: Icons, symbols, etc.
- Non-scientific literature on **sketchnote**, visual thinking, graphic facilitation
Inspirations

- **Psychology:** Research on perception, visual thinking, sketching, etc.
- **Semiotics:** Icons, Symbols, etc.
- Non-scientific literature on sketchnoting, visual thinking, graphic facilitation

Questions?

@s_baltes
s.baltes@uni-trier.de