Sketches in Software Engineering

Sebastian Baltes

University of Trier, Germany

@s_baltes

s.baltes@uni-trier.de

Talk in context of:

TRR 161

Transregional Collaborative Research Center
Quantitative Methods for Visual Computing

Outline

Past research:

"Sketches and Diagrams in Practice" FSE 2014

"Linking Sketches and Diagrams to Source Code Artifacts" FSE 2014

"Navigate, Understand, Communicate:
How Developers Locate Performance Bugs" esem 2015

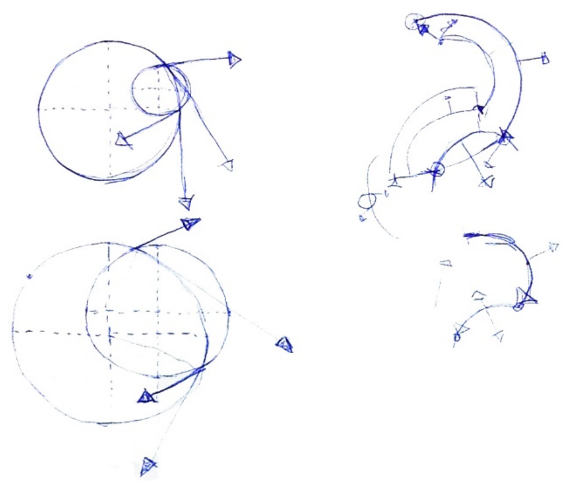
Future Research:

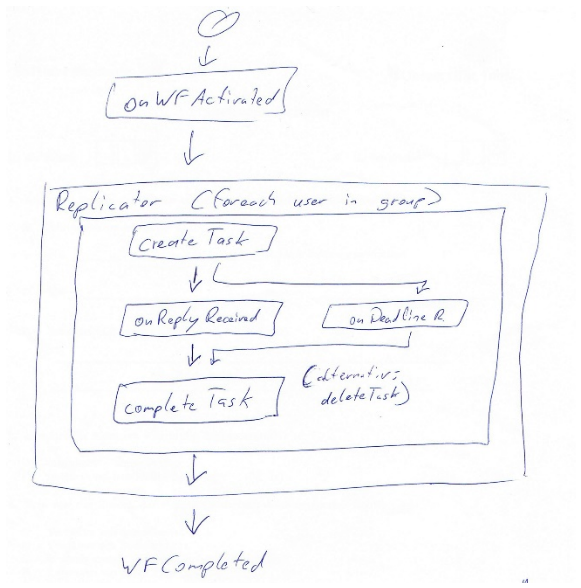
Do we need a Visual Literacy Curriculum for Developers?

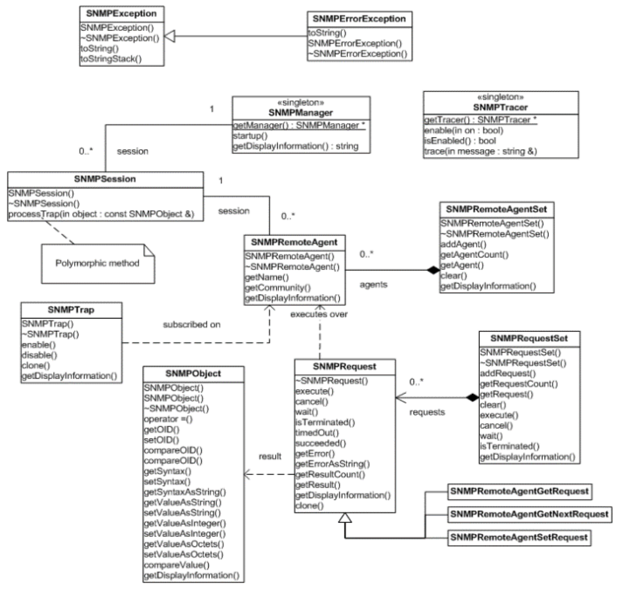
Sketches and Diagrams in Practice

Sebastian Baltes and Stephan Diehl

Sketches and Diagrams







informal ⟷ **formal**

Sketch

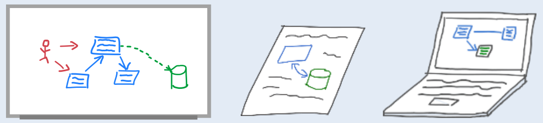
Diagram

Introduction

Past studies:
Sketches and diagrams important in daily work of software developers

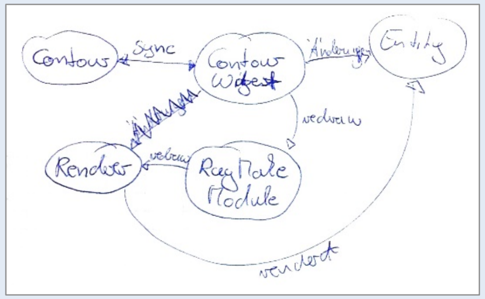
Purpose: Understanding, designing, communicating
[Cherubini07]

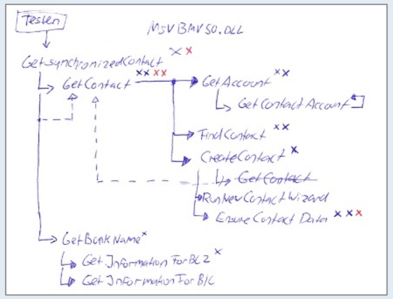
Depict **mental model** of software
[LaToza06]



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

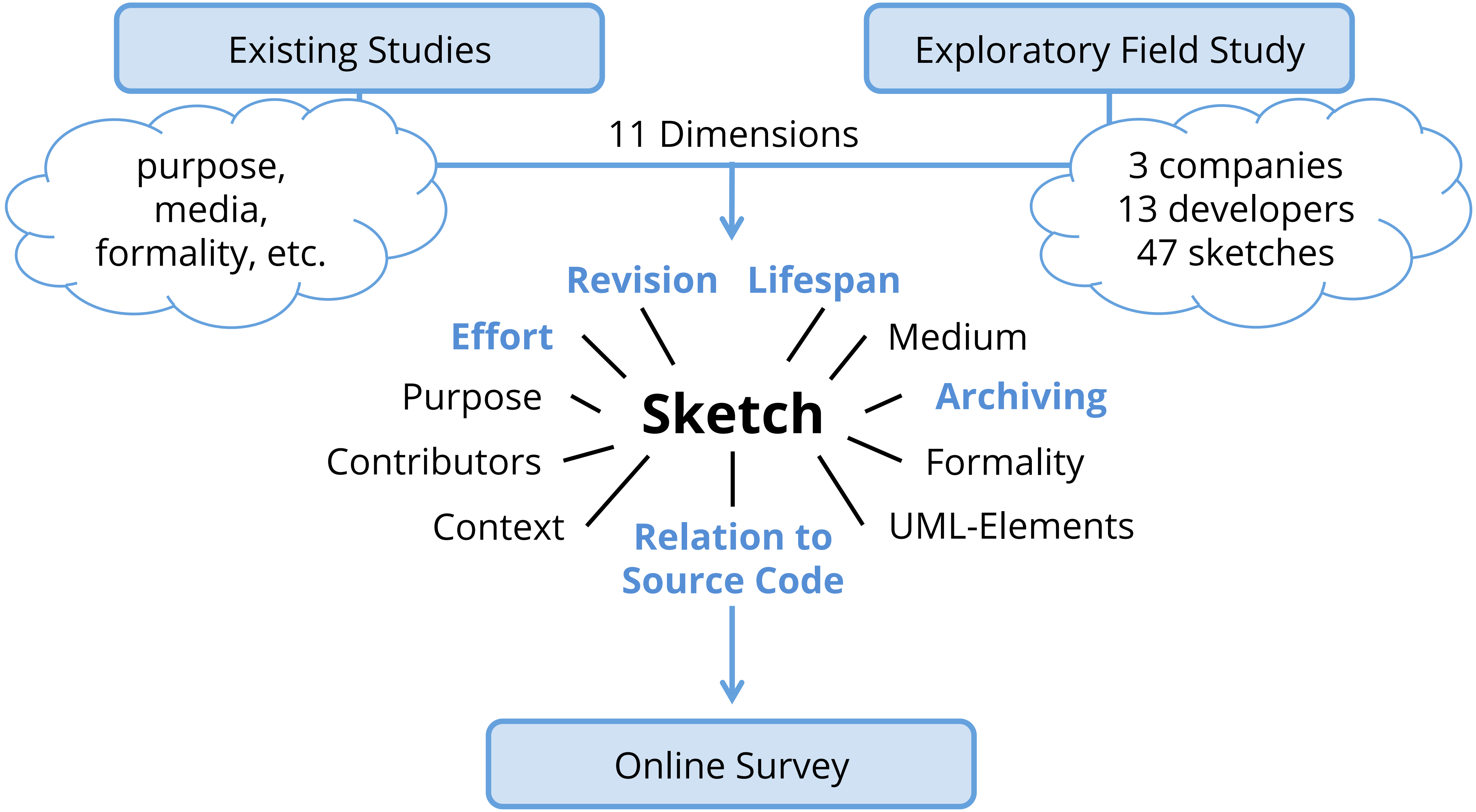
Our goal: Thorough description of how sketches and diagrams are used in software engineering practice

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

Research Design

How to describe sketches and diagrams in SE practice?

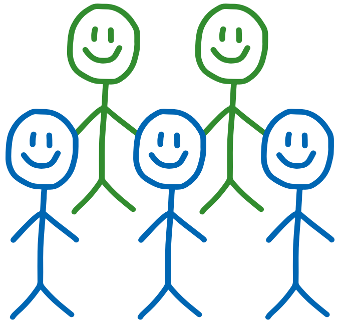
Research Design



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



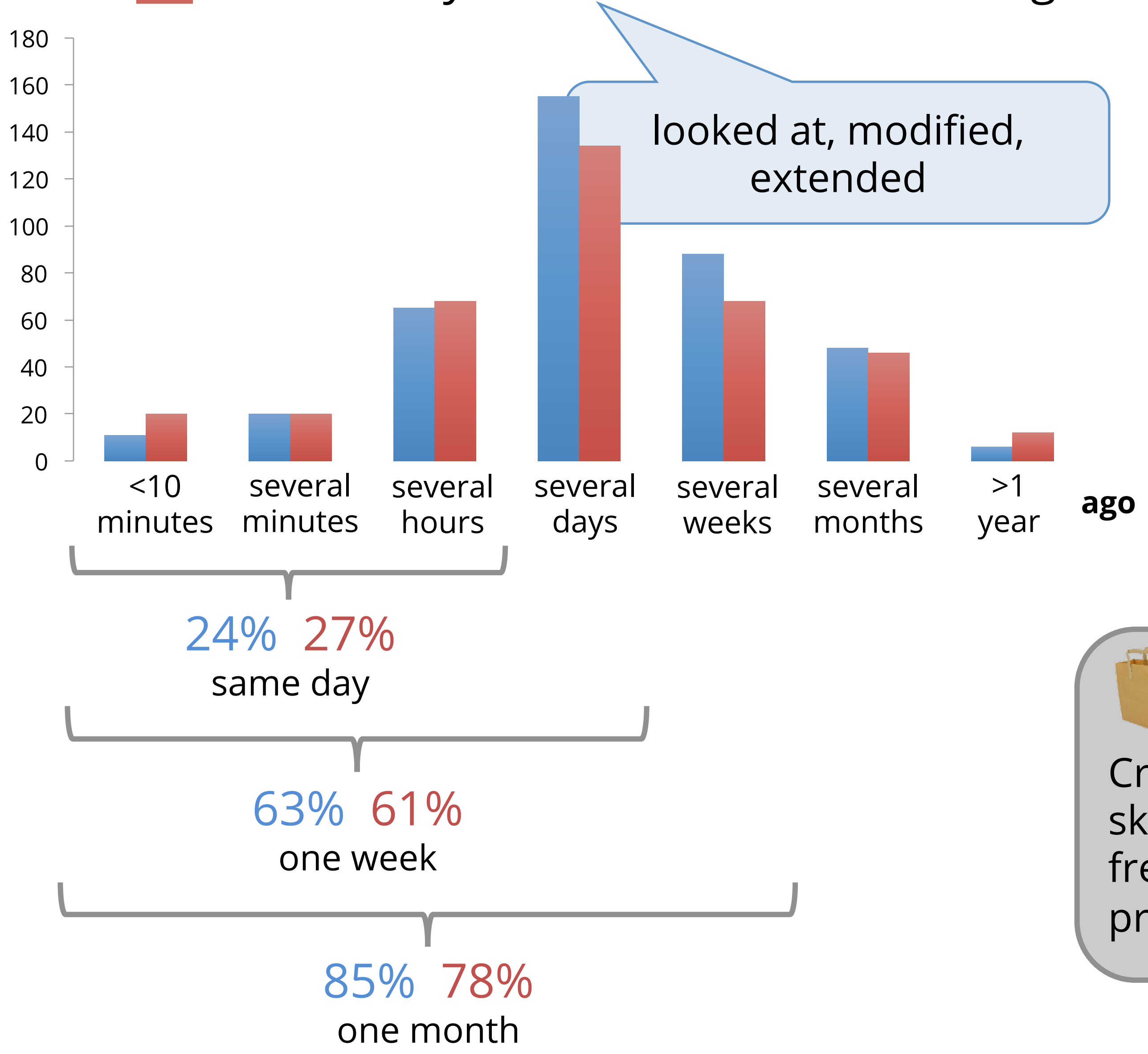
- **n=394**
- **32 countries**
  - 54% Germany 15% North America
- 52% software developers, 22% software architects
- Time spent developing software: **80%** (median)
- Professional work experience: **10 years** (median)
- Software projects from various **application areas**

Results

Creation and Usage

🟦 When did you create your last sketch/diagram?

🟥 When did you use the last sketch/diagram **created by some else**?

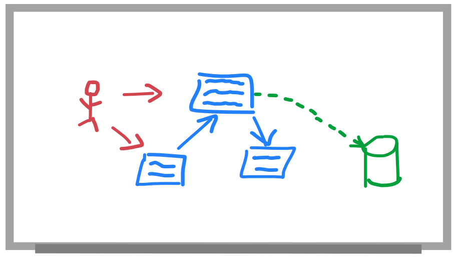


Takeaway 1:

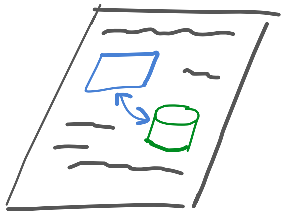
Creating own sketches **and using** sketches created by others are frequent tasks among software practitioners.

Media

What medium did you use to create the sketch/diagram?

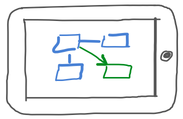


Whiteboard (40%)

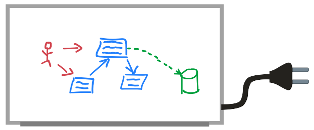


Paper (18%)

Computer (39%)



Tablet (0.8%)



E-Whiteboard (1.5%)

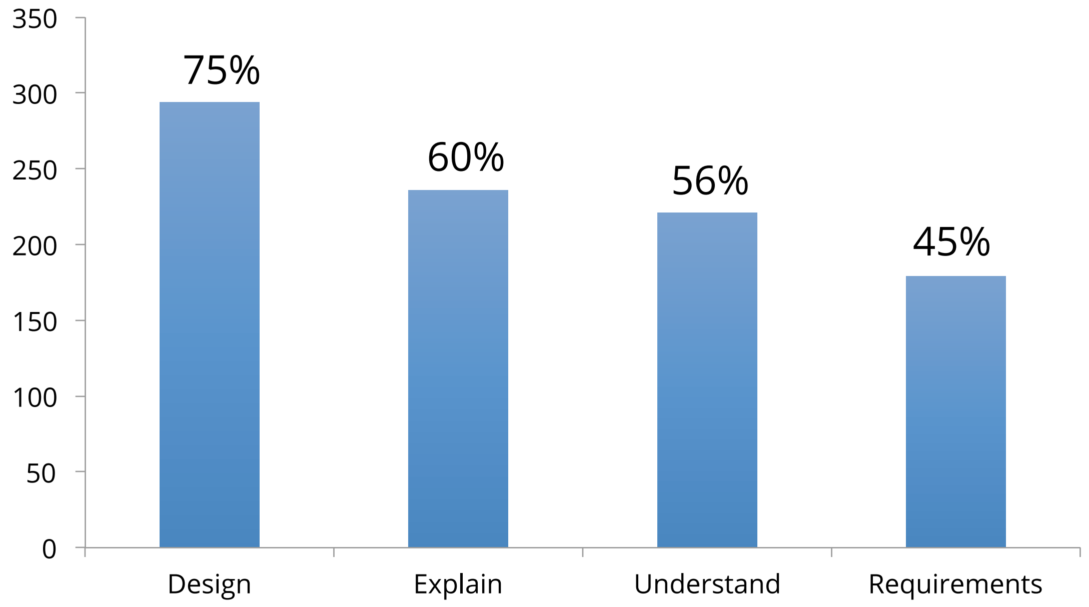
Analog (58%)

Digital (42%)

Purpose

- The sketch/diagram helped me to...
  (multiple answers possible)

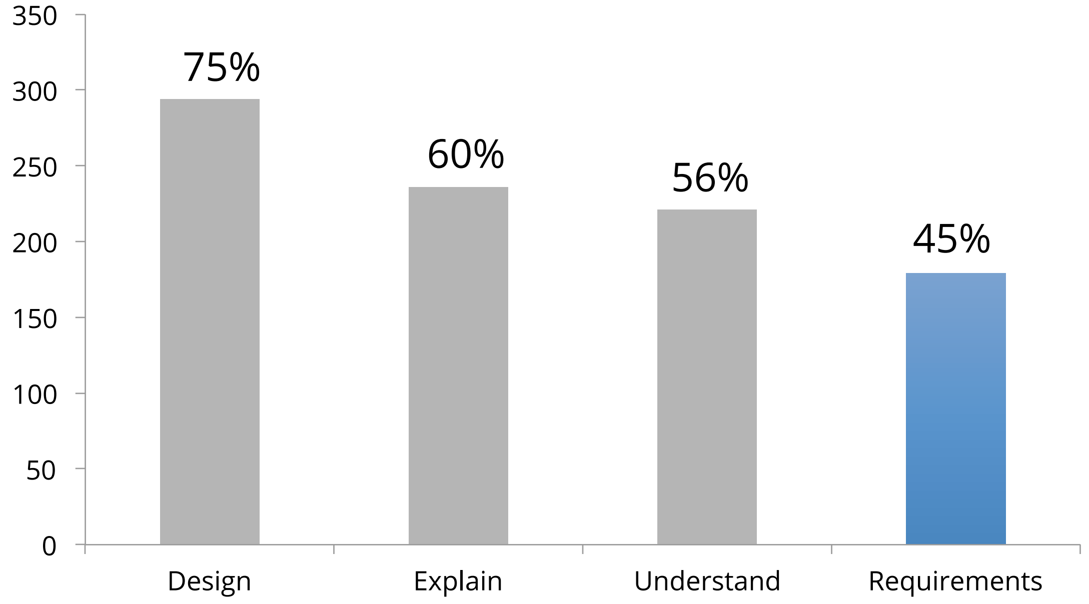
  ...design a new architecture (52%)
  ...design new features (48%)
  ...explain an issue to someone else (46%)
  ...analyze requirements (45%)
  ...understand an issue (44%)



Purpose

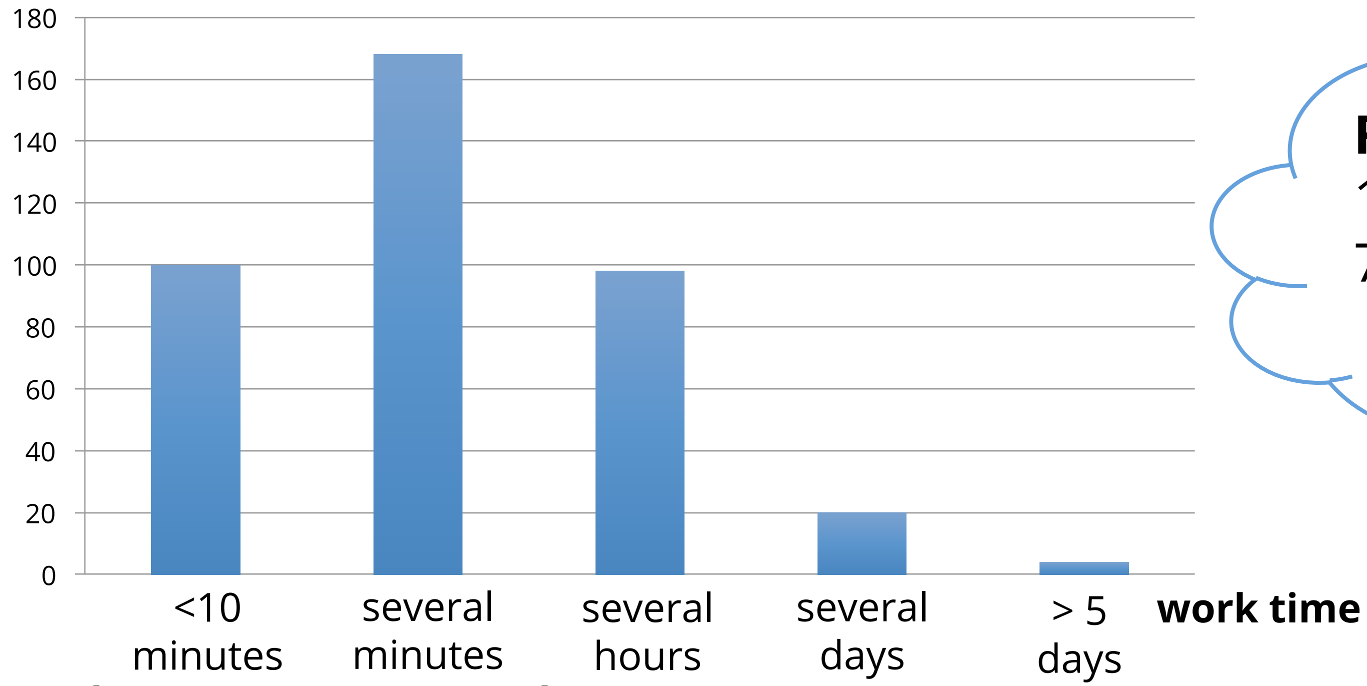
- The sketch/diagram helped me to…
  (multiple answers possible)

  …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

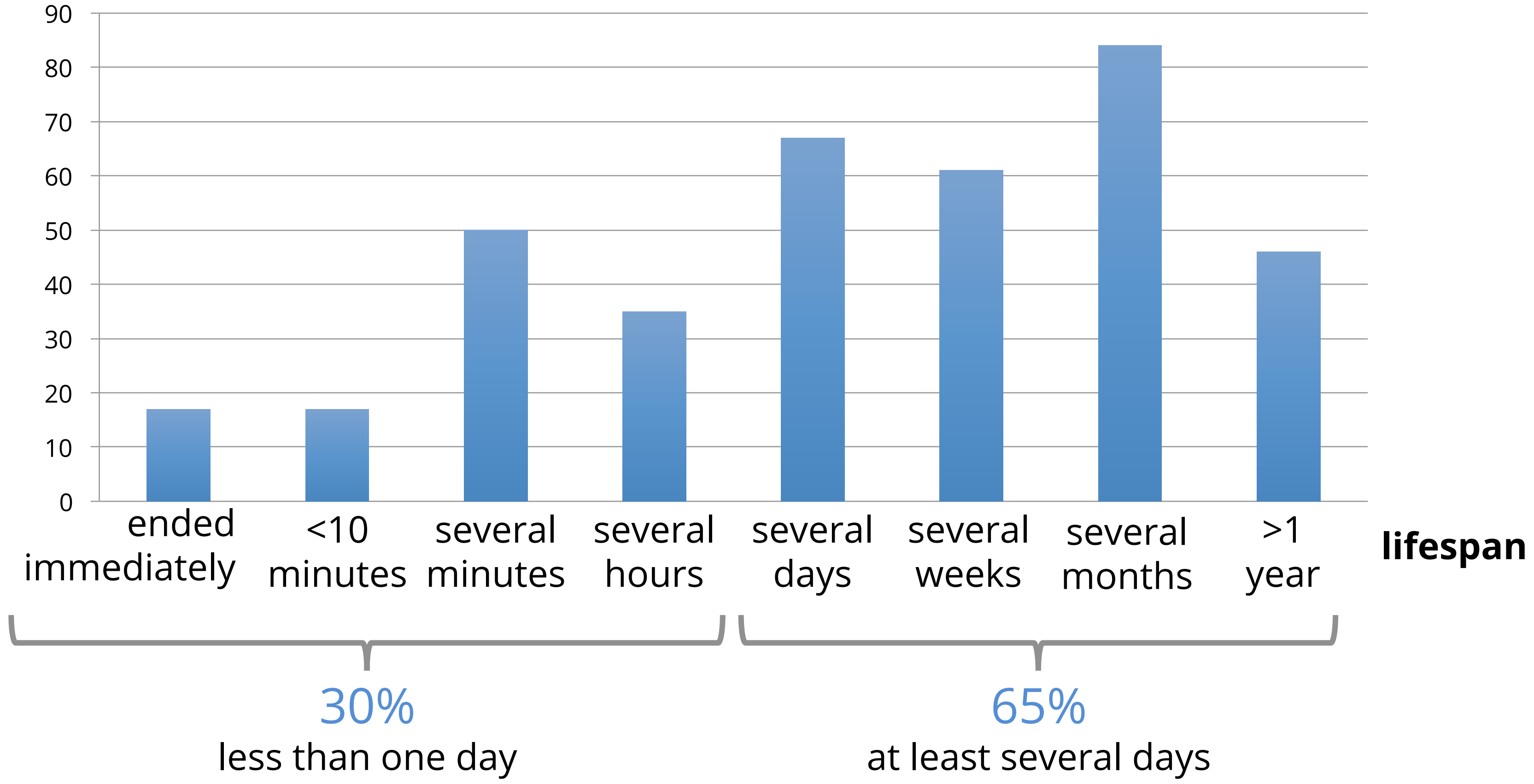
Revision:
15% revised once,
74% multiple times

Takeaway 2:

Most sketches are created in **less than one hour** and are **revised** at least once.

Lifespan

- Please try to estimate the lifespan of the sketch/diagram (how long did/will you use it)?



Takeaway 3:

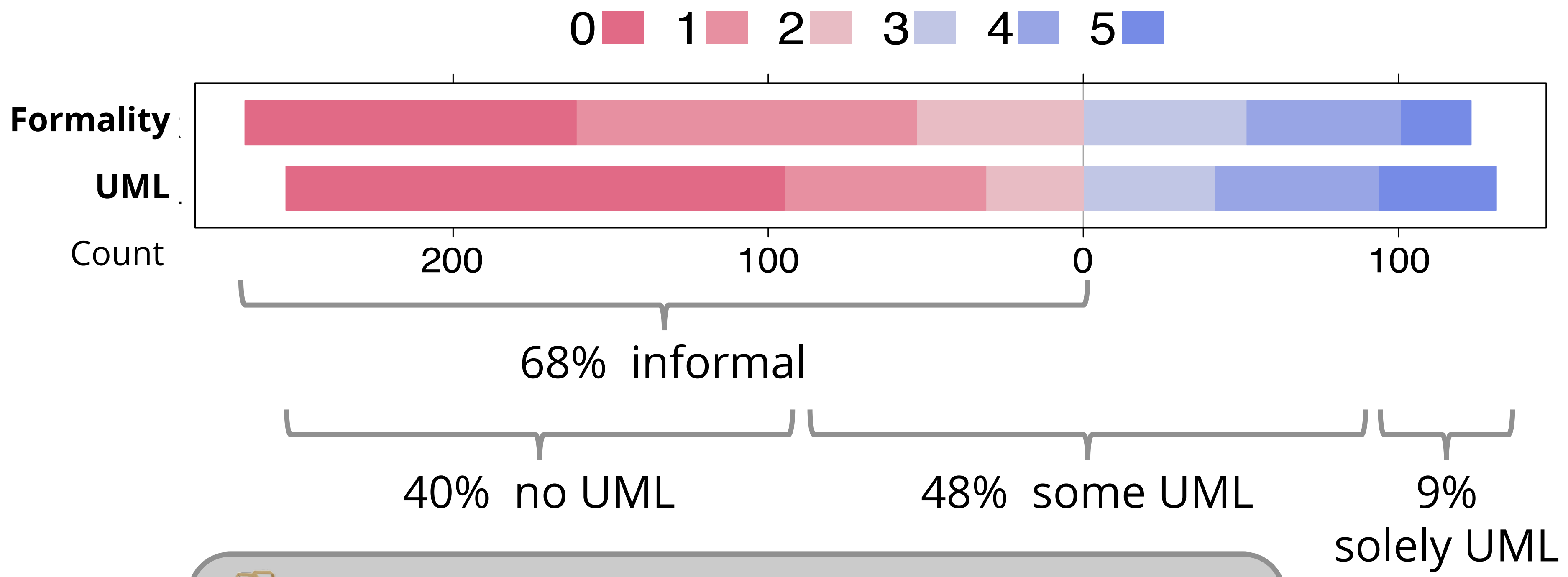
Almost half of the sketches are used for **at least several weeks**.

48%
at least several weeks

Formality and UML

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

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

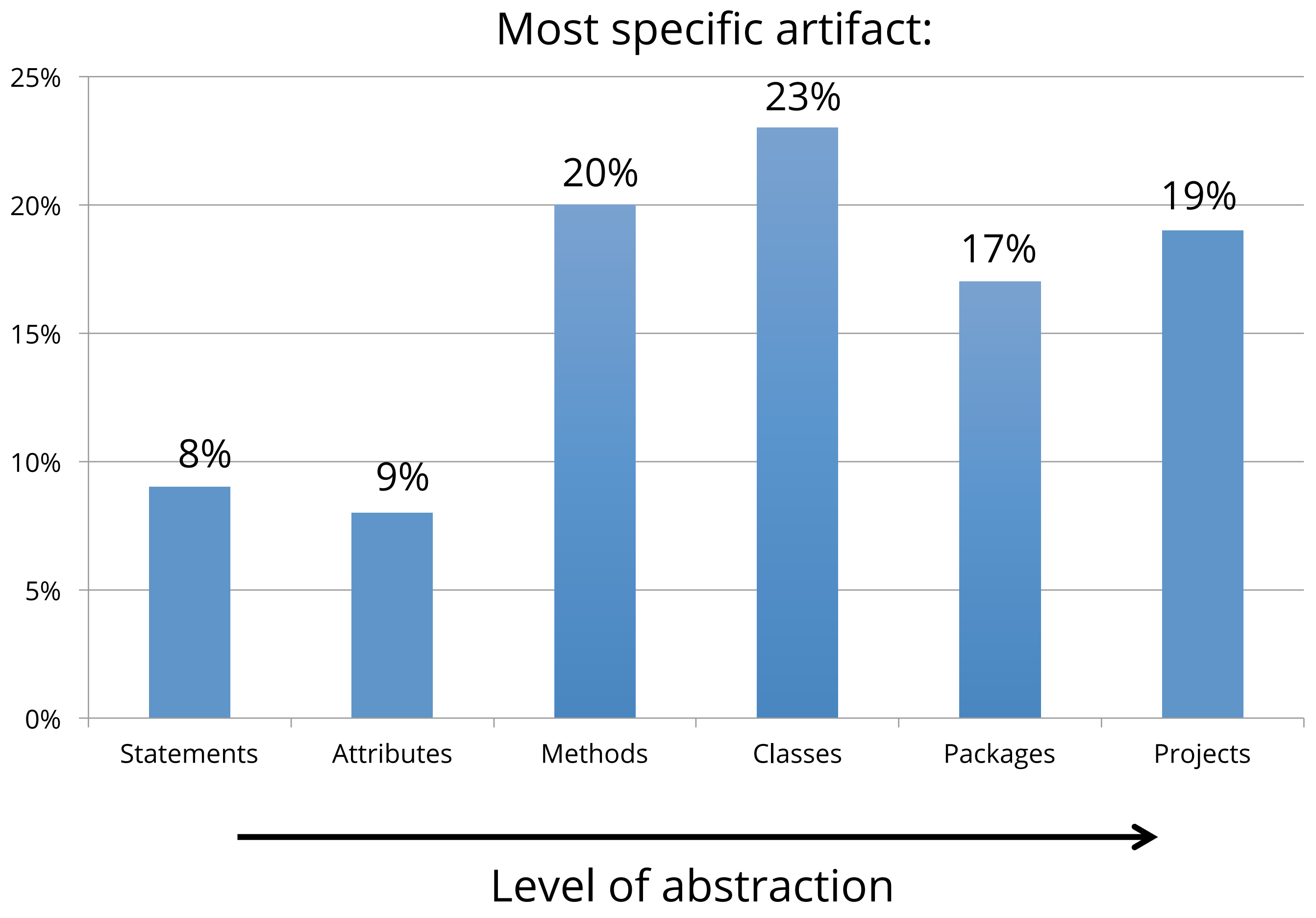


Takeaway 4:

The majority of sketches and diagrams are **informal**.
If UML is used, it is often mixed with other notations.

Relation to Source Code

- Please select the software artifact(s) to which the content of the sketch/diagram is related?
  (multiple answers or no answer possible)

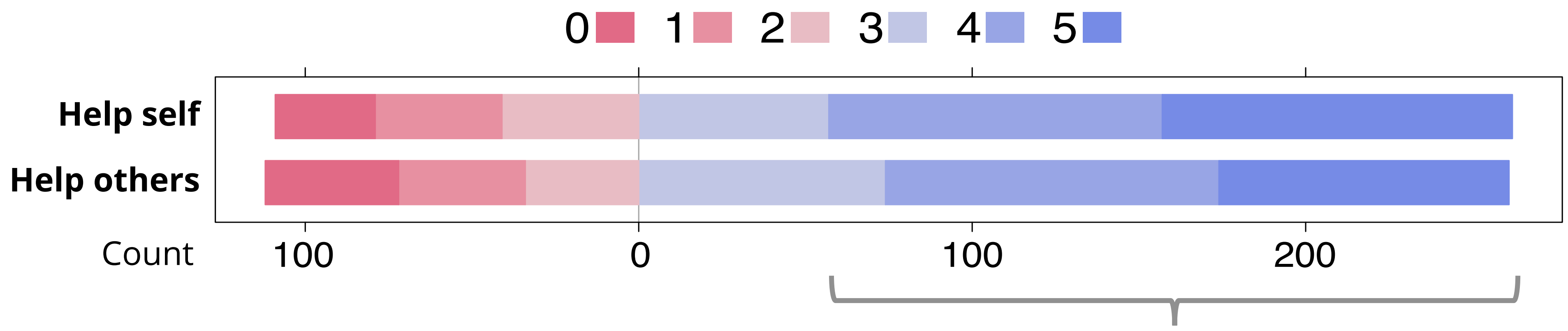


Relation to Source Code

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 (0-5) from "It will definitely not help " to "It will definitely help")

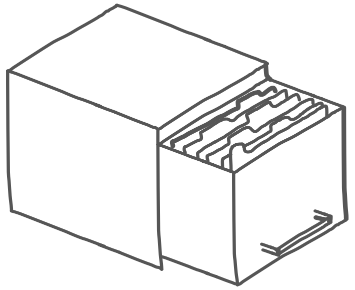


52% and 47% Likert 4-5

Takeaway 5:

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

Archiving



Three questions:

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

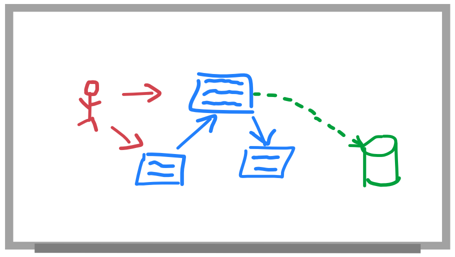
58% archived

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

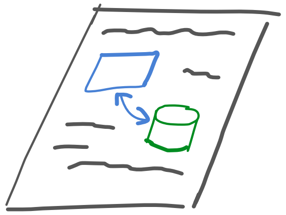
3. If the sketch has not been archived and won't be archived, **why do you not want to keep it**?

- Answers analyzed using an approach based on open coding
- Extracted four categories for the answers to each question
- One category for archiving practice

Archiving

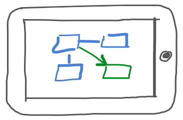


Whiteboard (40%)

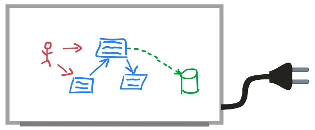


Paper (18%)

Computer (39%)



Tablet (0.8%)

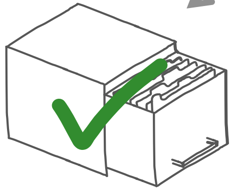


E-Whiteboard (1.5%)

Analo **42%)**

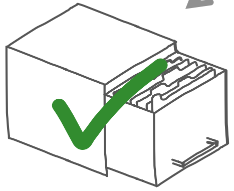
Takeaway 6:

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



Archived (38%)

Not archived (62%)



Archived (94%)

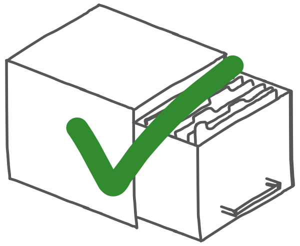
Not archived (6%)

Archiving – Why?

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

Documentation

Future Use



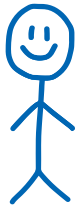
Understanding

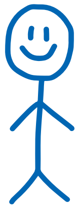
Visualization

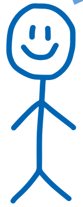
"It will be difficult to understand the code without the diagram."

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

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





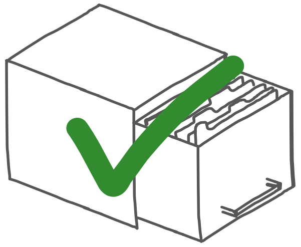


Archiving – Why?

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

Documentation

Understanding



Future Use

Visualization

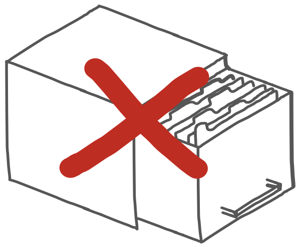
Takeaway 7:

Sketches are kept, because they **document** software, **visualize** it, and support its **understanding**.

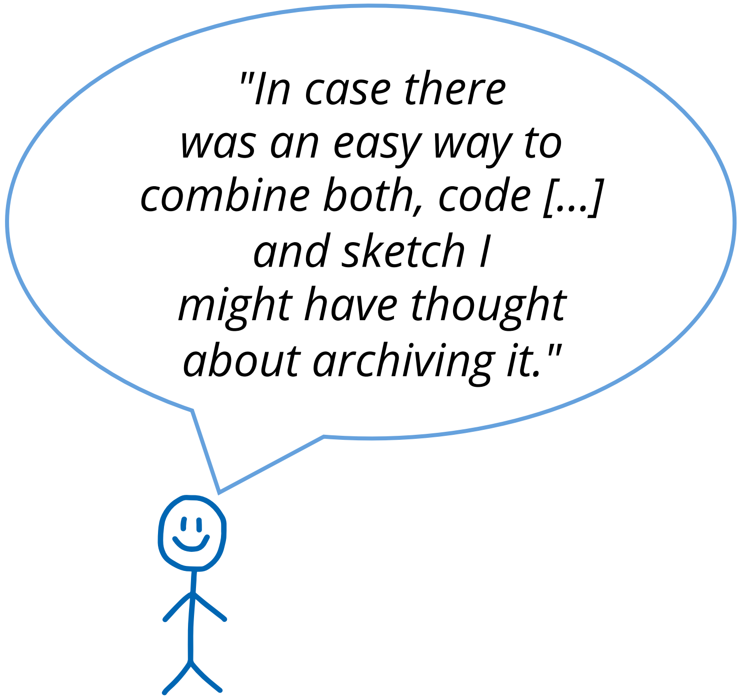
Archiving – Why not?

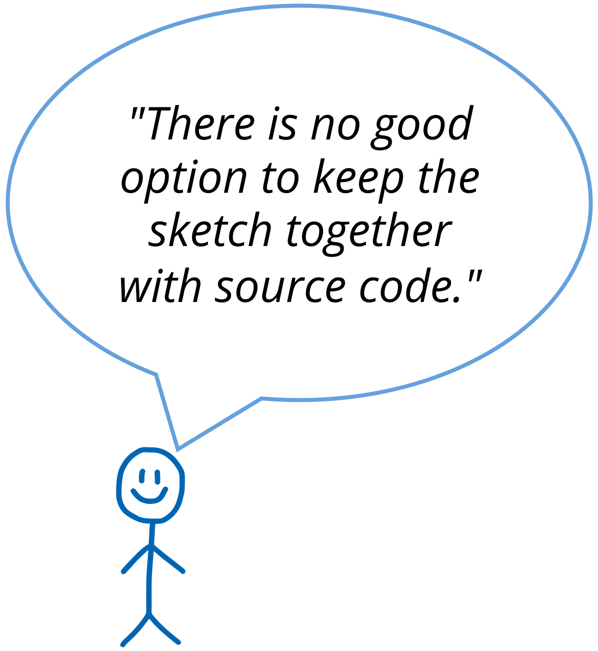
If the sketch has not been archived and won't be archived, why do you not want to keep it?

- Served its purpose
- Outdated
- Substituted
- 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?

Summary

Takeaways

1. Creating own sketches/diagrams **and using** sketches/diagrams created by others are frequent tasks among software practitioners
2. Most sketches/diagrams are **created in less than one hour** and are **revised** at least once after creation
3. Almost half of the sketches/diagrams are **used for at least several weeks**
4. Majority of sketches/diagrams are **informal**
5. About half of the sketches/diagrams are rated as **helpful** to understand the related source code artifact(s) in the future
6. Most digital sketches/diagrams, but also more than one third of the analog ones, are **archived**
7. Sketches/diagrams **document** the implementation, visualize it, and support its understanding

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`

Artifact ★ Consistent ★ Complete ★ Well Documented ★ Easy to Reuse ★ Evaluated ★ FSE ★ AEC

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

FSE 2014

SketchLink

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 a 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
- Interviews
- Profiling and analysis tool (list and in-situ)

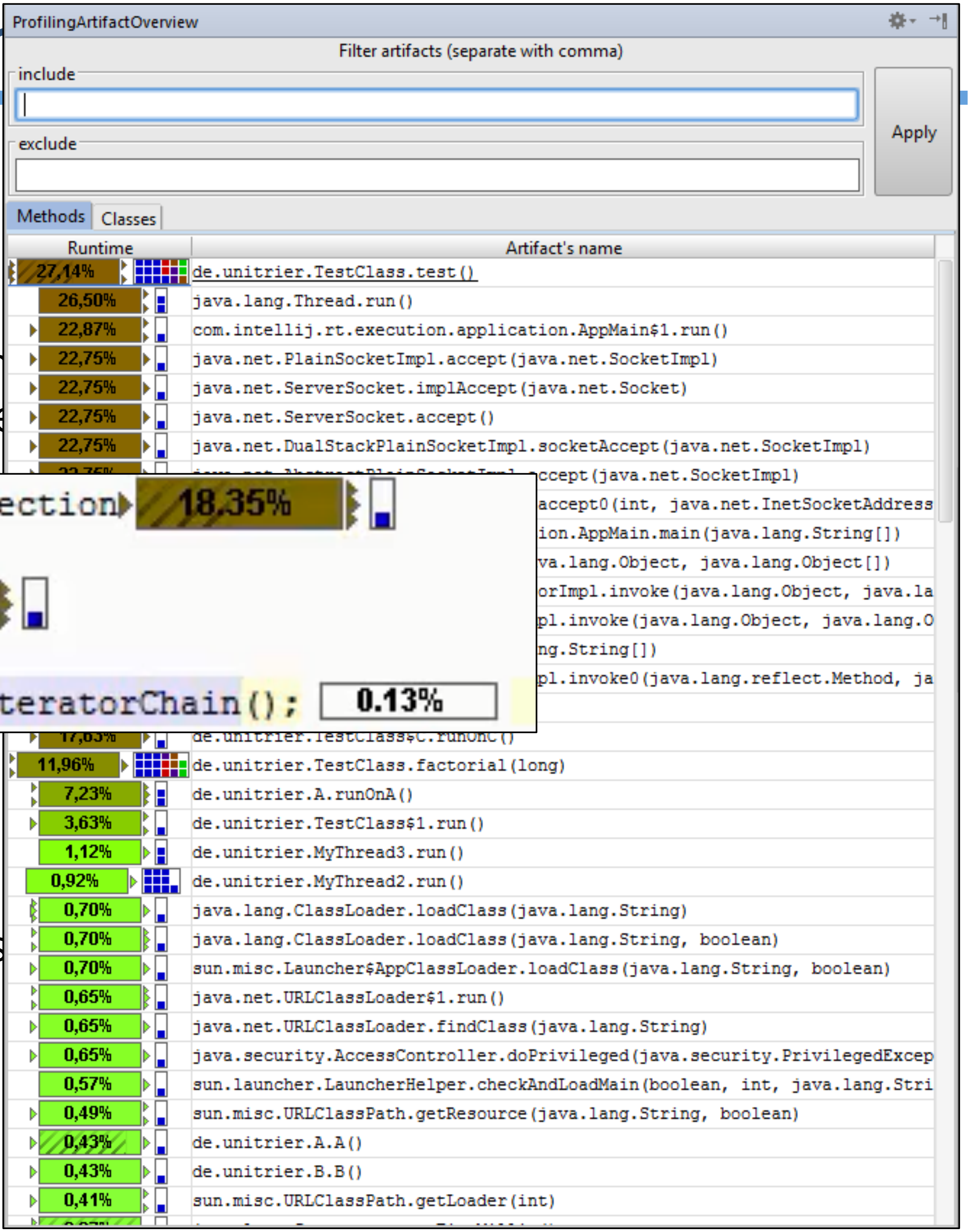
Sum

Objective:

Investigate how developers, when

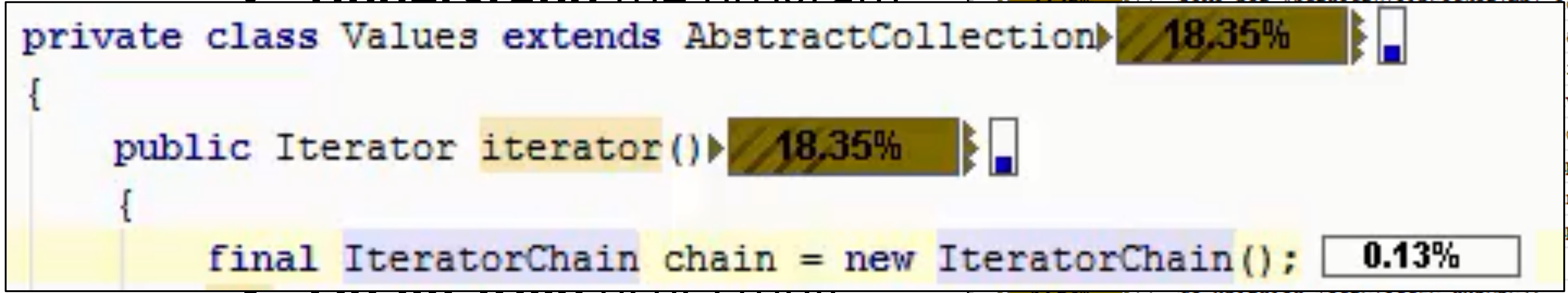
- **Navigate** through the source
- **Understand** the program

- **Qualitative** user study
- Observed **12 developers** fixi
  open source projects
- Interviews
- Profiling and analysis tool (lis



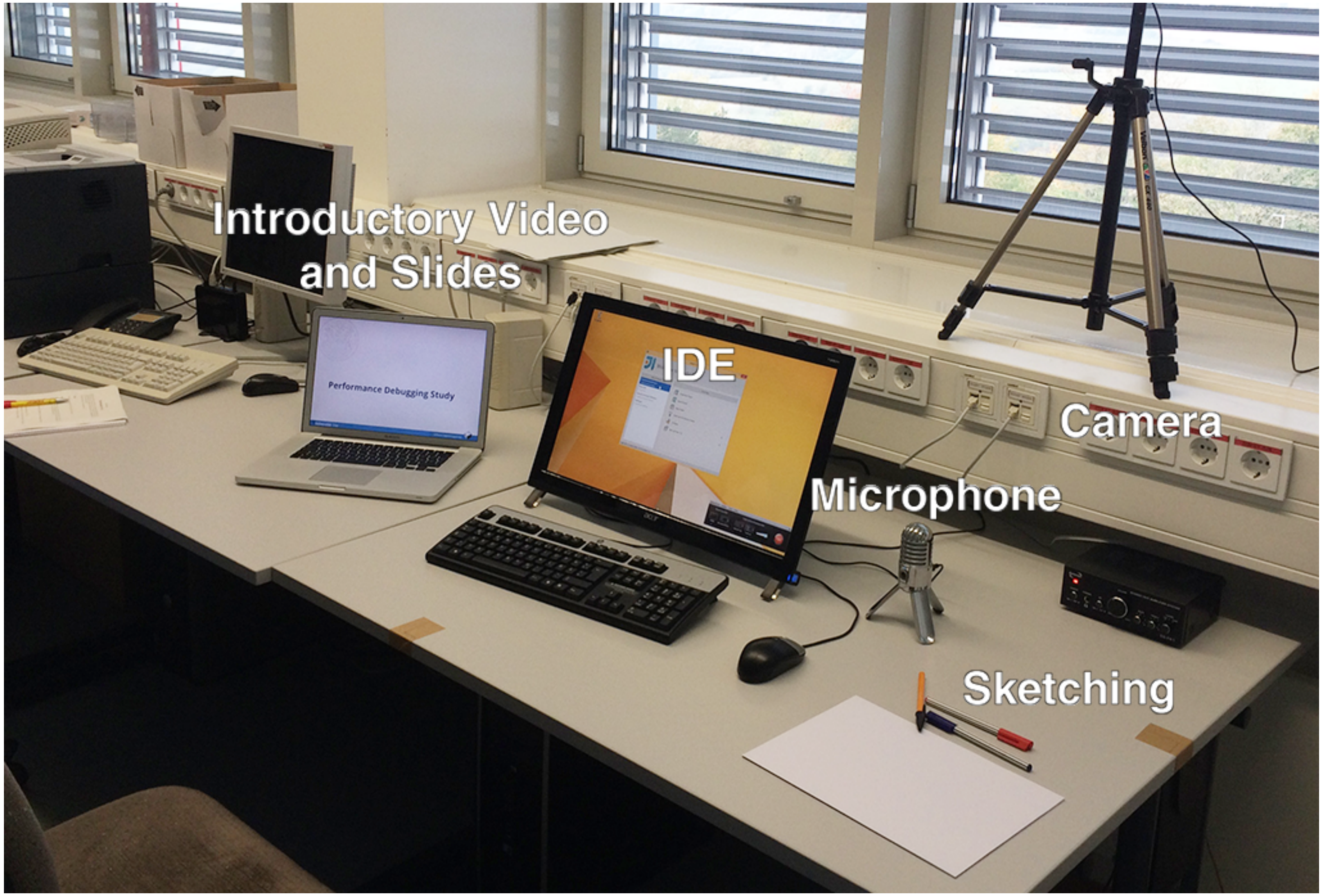
ProfilingArtifactOverview
Filter artifacts (separate with comma)
include
exclude
Apply
Methods
Classes

Runtime	Artifact's name
27,14%	de.unitrier.TestClass.test()
26,50%	java.lang.Thread.run()
22,87%	com.intellij.rt.execution.application.AppMain$1.run()
22,75%	java.net.PlainSocketImpl.accept(java.net.SocketImpl)
22,75%	java.net.ServerSocket.implAccept(java.net.Socket)
22,75%	java.net.ServerSocket.accept()
22,75%	java.net.DualStackPlainSocketImpl.socketAccept(java.net.SocketImpl)
22,75%	ccept(java.net.SocketImpl)
	accept0(int, java.net.InetSocketAddress
	ion.AppMain.main(java.lang.String[])
	va.lang.Object, java.lang.Object[])
	orImpl.invoke(java.lang.Object, java.la
	pl.invoke(java.lang.Object, java.lang.O
	ng.String[])
	pl.invoke0(java.lang.reflect.Method, ja
17,63%	de.unitrier.TestClass$C.runOnC()
11,96%	de.unitrier.TestClass.factorial(long)
7,23%	de.unitrier.A.runOnA()
3,63%	de.unitrier.TestClass$1.run()
1,12%	de.unitrier.MyThread3.run()
0,92%	de.unitrier.MyThread2.run()
0,70%	java.lang.ClassLoader.loadClass(java.lang.String)
0,70%	java.lang.ClassLoader.loadClass(java.lang.String, boolean)
0,70%	sun.misc.Launcher$AppClassLoader.loadClass(java.lang.String, boolean)
0,65%	java.net.URLClassLoader$1.run()
0,65%	java.net.URLClassLoader.findClass(java.lang.String)
0,65%	java.security.AccessController.doPrivileged(java.security.PrivilegedExcep
0,57%	sun.launcher.LauncherHelper.checkAndLoadMain(boolean, int, java.lang.Stri
0,49%	sun.misc.URLClassPath.getResource(java.lang.String, boolean)
0,43%	de.unitrier.A.A()
0,43%	de.unitrier.B.B()
0,41%	sun.misc.URLClassPath.getLoader(int)



```
private class Values extends AbstractCollection 18,35%
{
    public Iterator iterator() 18,35%
    {
        final IteratorChain chain = new IteratorChain(); 0.13%
```

Study Setup

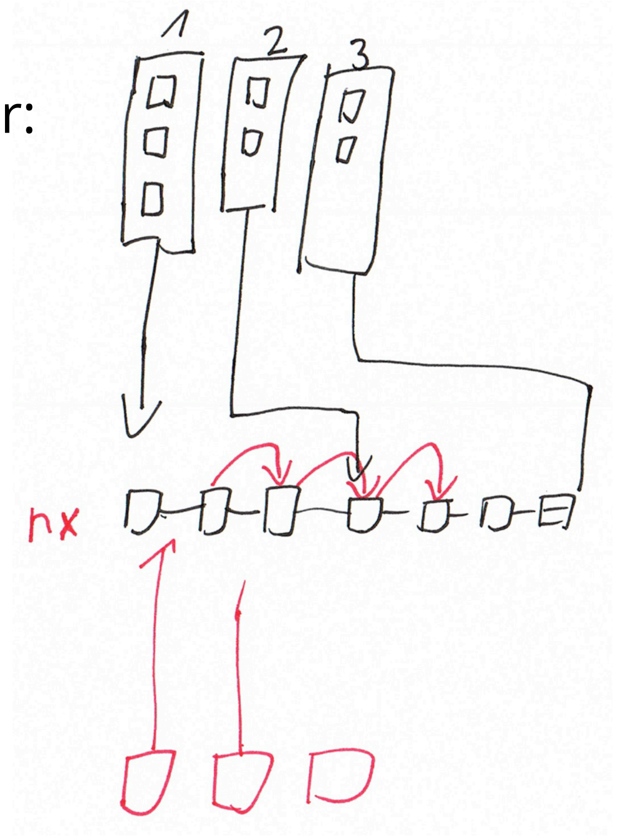

Introductory Video and Slides
Performance Debugging Study
IDE
Camera
Microphone
Sketching

Sketches and Performance Bugs

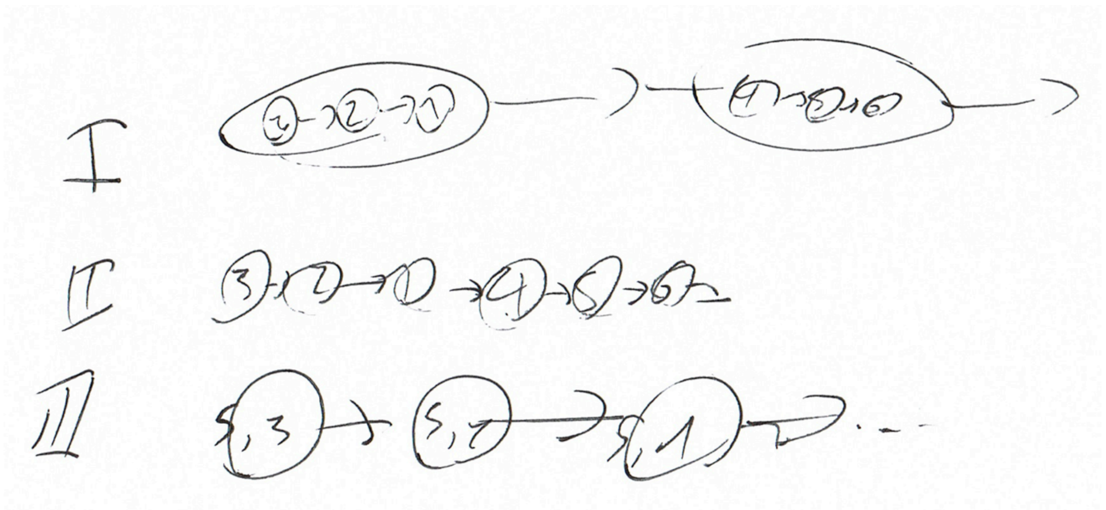
RQ2.2:
Could sketches help to understand and communicate a performance bug?

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

Dynamic Behavior:



Alternatives:



But, ...

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)

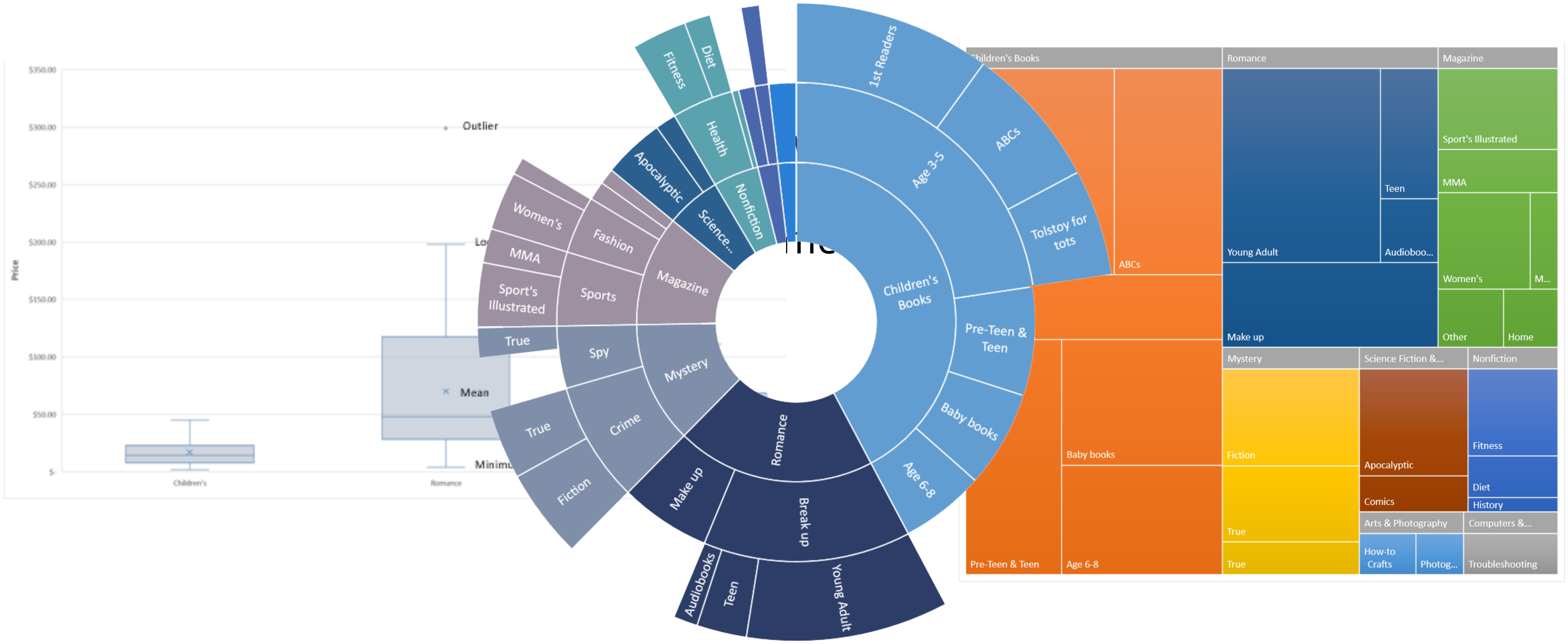
Visualization Literacy

"[Visualization literacy is] the ability to confidently **use** a given data visualization **to translate** questions specified in the data domain **into visual queries** in the visual domain, as well as **interpreting** visual patterns in the visual domain as properties in the data domain." (Box et al., 2014)

→ Rather **passive** role of the user
→ **But:** Today, users also need to know which visualization is suitable for their data
→ Many new visualizations, e.g. in Office 2016 Preview

Visualization Literacy

"[Visualization literacy is] the ability to confidently **use** a given data visualization **to translate** questions specified in the data domain **into visual queries** in the visual domain, as well as **interpreting** visual patterns in the visual domain as properties in the data domain." (Box et al., 2014)



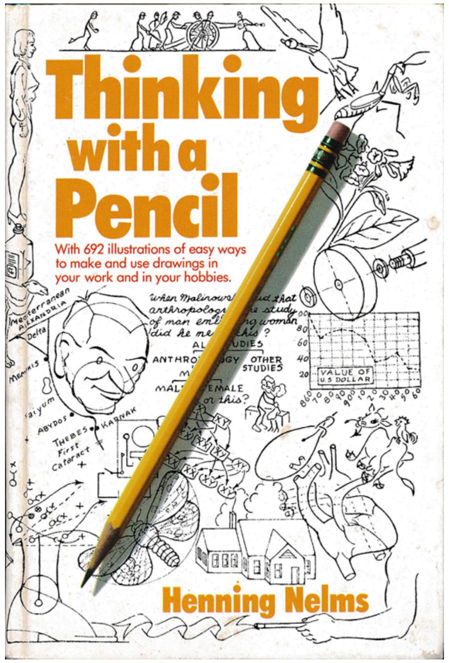
Gap?

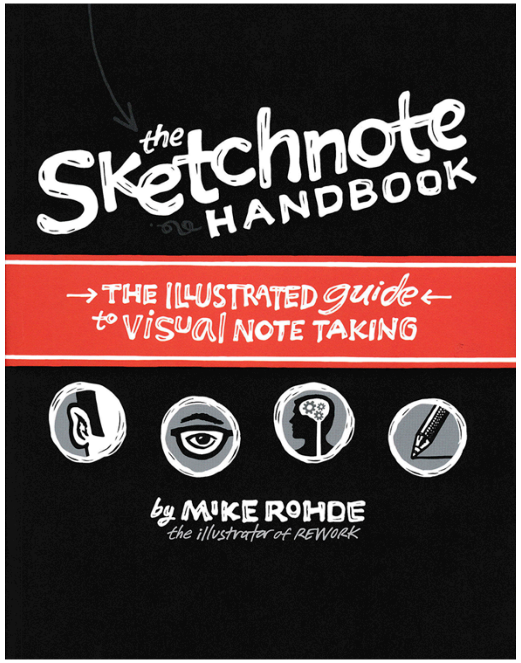
- **Research** in visual(ization) literacy **often focuses on reading** and interpreting visuals or visualizations
- Not much work on **"production literacy"** (Messaris, 1994)

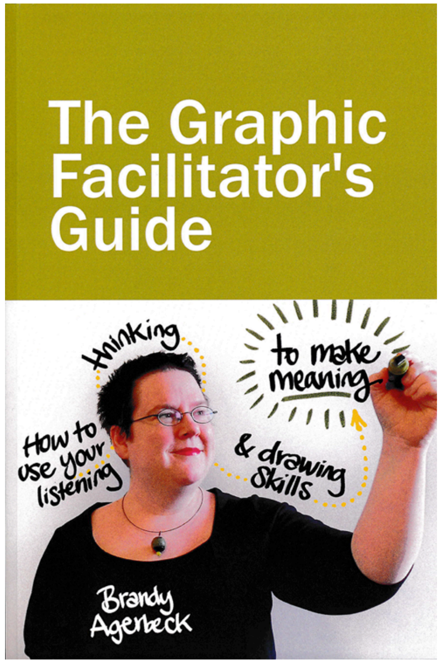
Our goal: Develop a lightweight curriculum to teach software developers how to produce simple visuals for communicating their ideas.

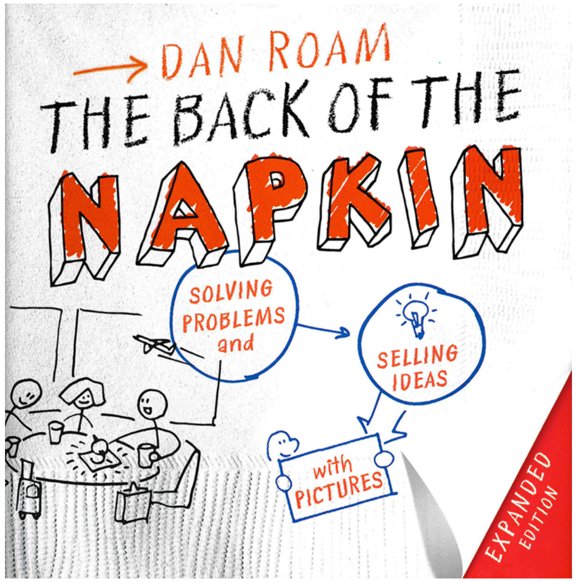
Inspirations

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


Thinking with a Pencil
With 692 illustrations of easy ways to make and use drawings in your work and in your hobbies.
Henning Nelms

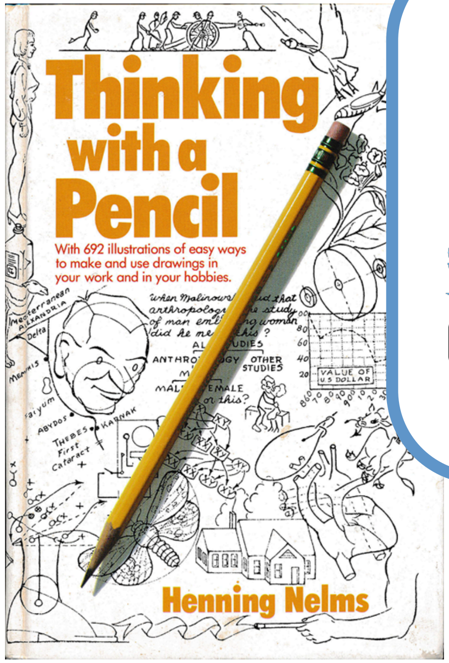

the Sketchnote HANDBOOK
→ THE ILLUSTRATED guide ← to visual NOTE TAKING
by MIKE ROHDE
the illustrator of REWORK

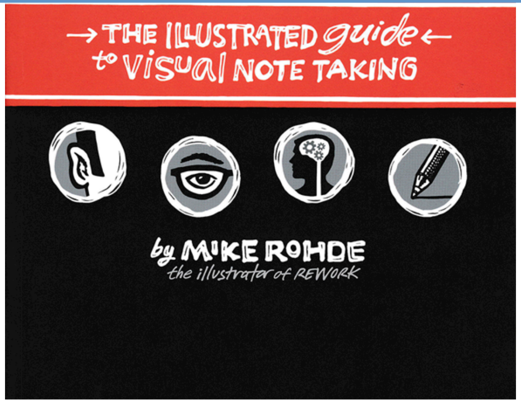


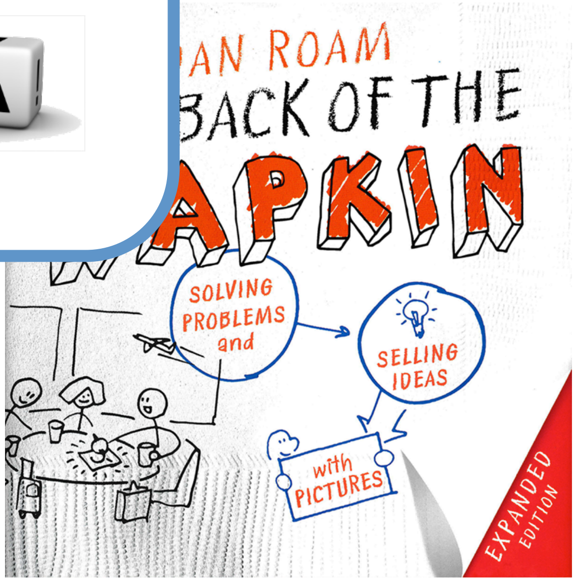


Inspirations

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


Thinking with a Pencil
With 692 illustrations of easy ways to make and use drawings in your work and in your hobbies.
Henning Nelms


THE ILLUSTRATED guide to visual NOTE TAKING
by MIKE ROHDE
the illustrator of REWORK



Questions?

@s_baltes

s.baltes@uni-trier.de