On the Usability of Domain-Specific Languages

Vasco Amaral

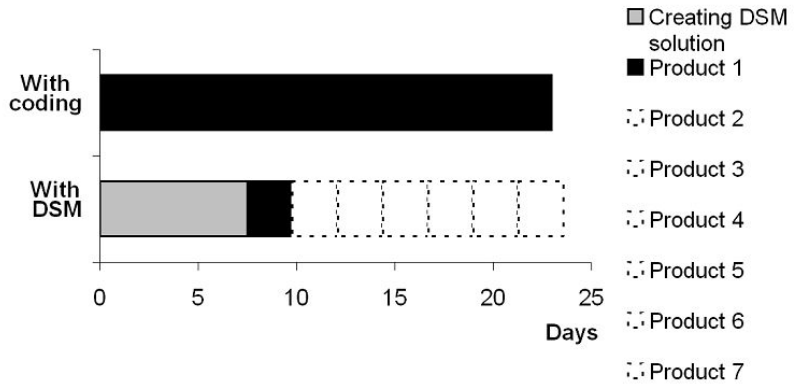
Lecture at:
Faculty of Technical Sciences
University of Novi Sad
21/3/2019

DSLs are the Silver Bullet

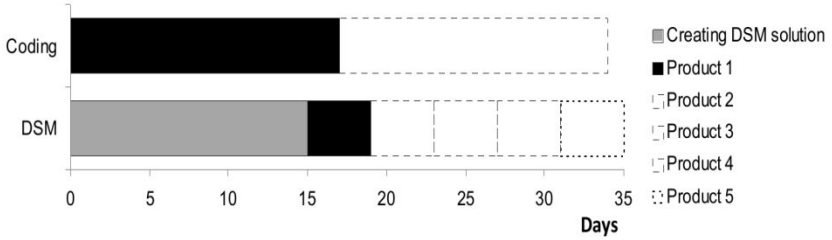
- Why?
  - Increase productivity
- How?
  - Minimizing the accidental complexity
    - Bridging the gap between problem and solution domain
    - Empowering domain users to work at the right abstraction level, with the most adequate formalism
    - Minimizing the errors by constraining the design space

Anecdotal evidences of return of investment

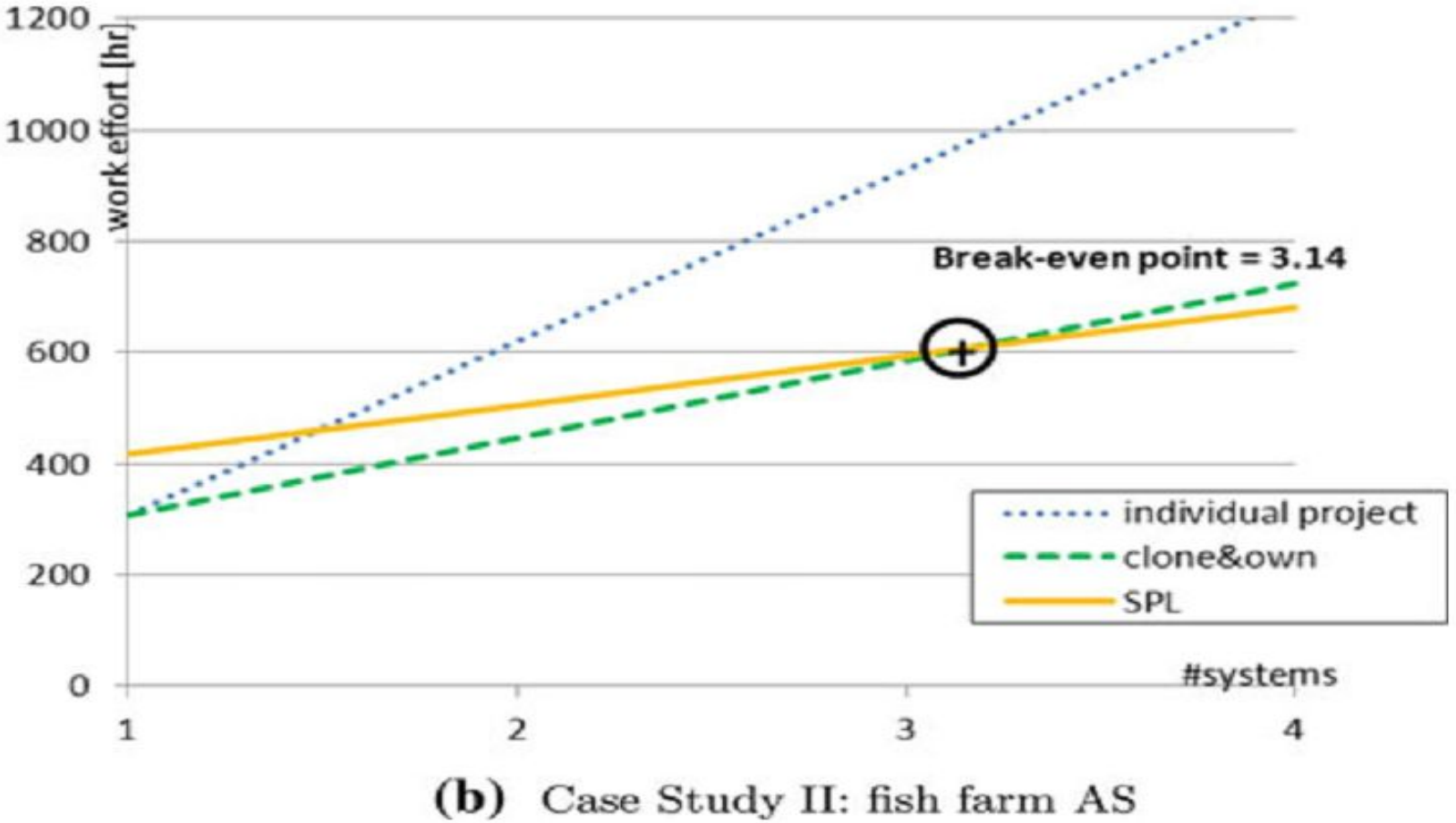
POLAR DSL



Panasonic touchscreen DSL

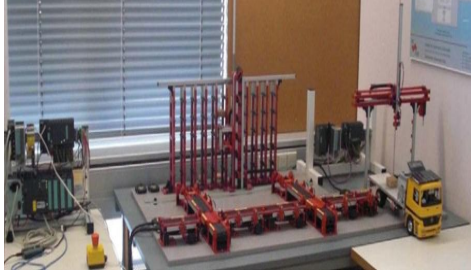


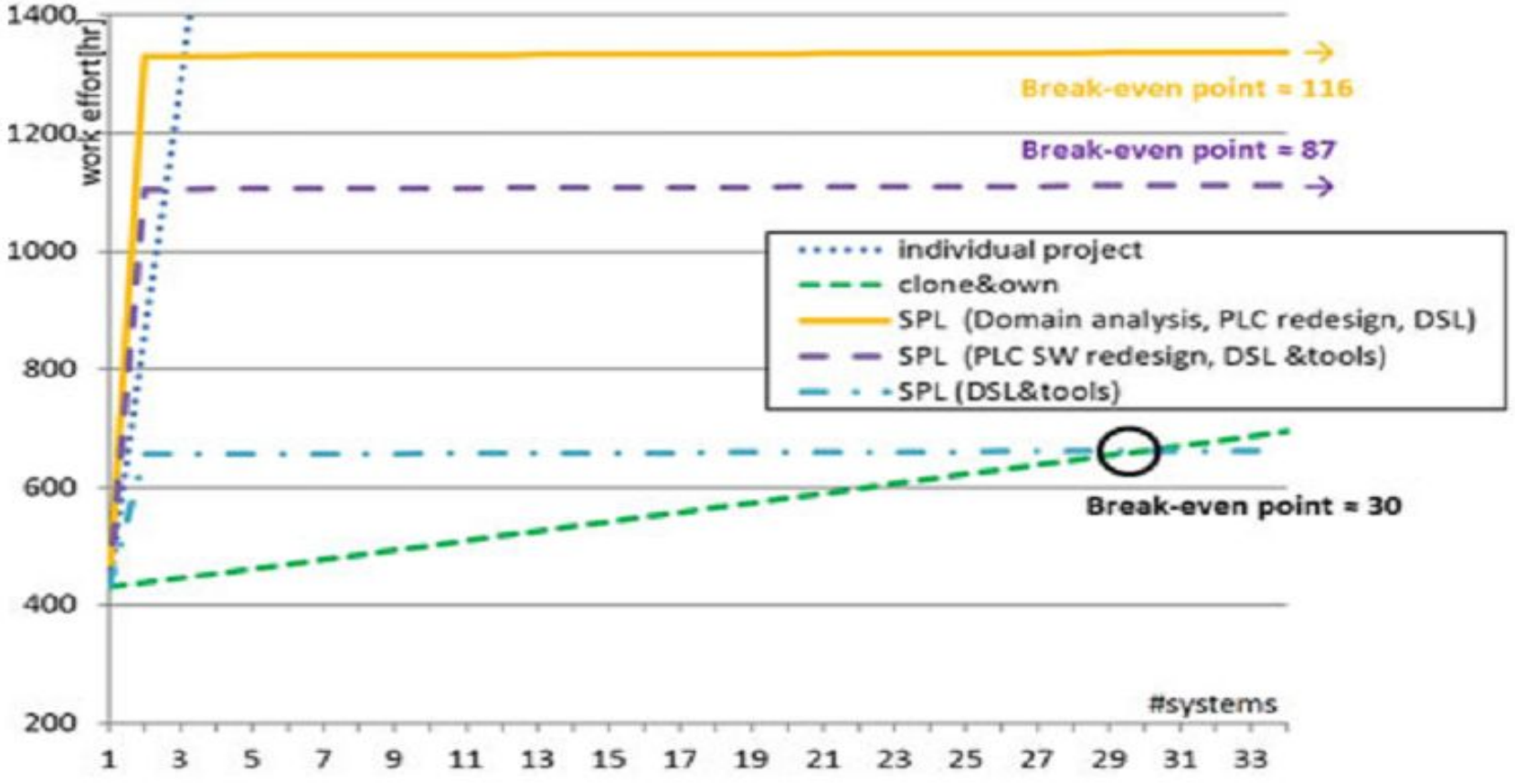
On this one, our break-even point is just 4 products away…



(b) Case Study II: fish farm AS

Oooops...





(a) Case Study I: inventory AS

What are the DSML cost drivers?

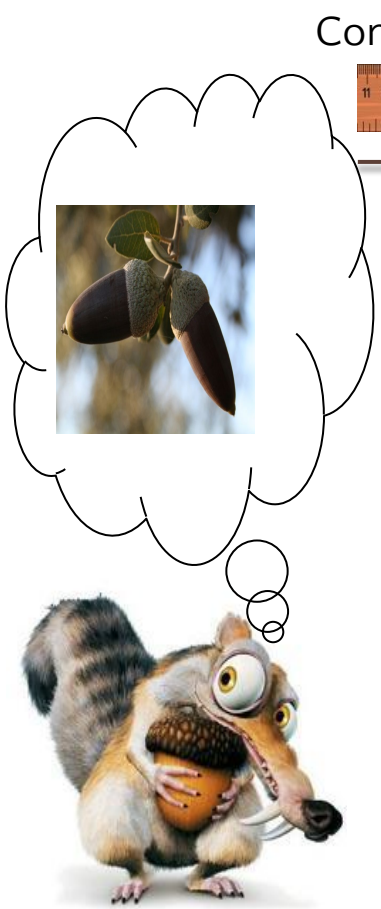
Technical cost drivers

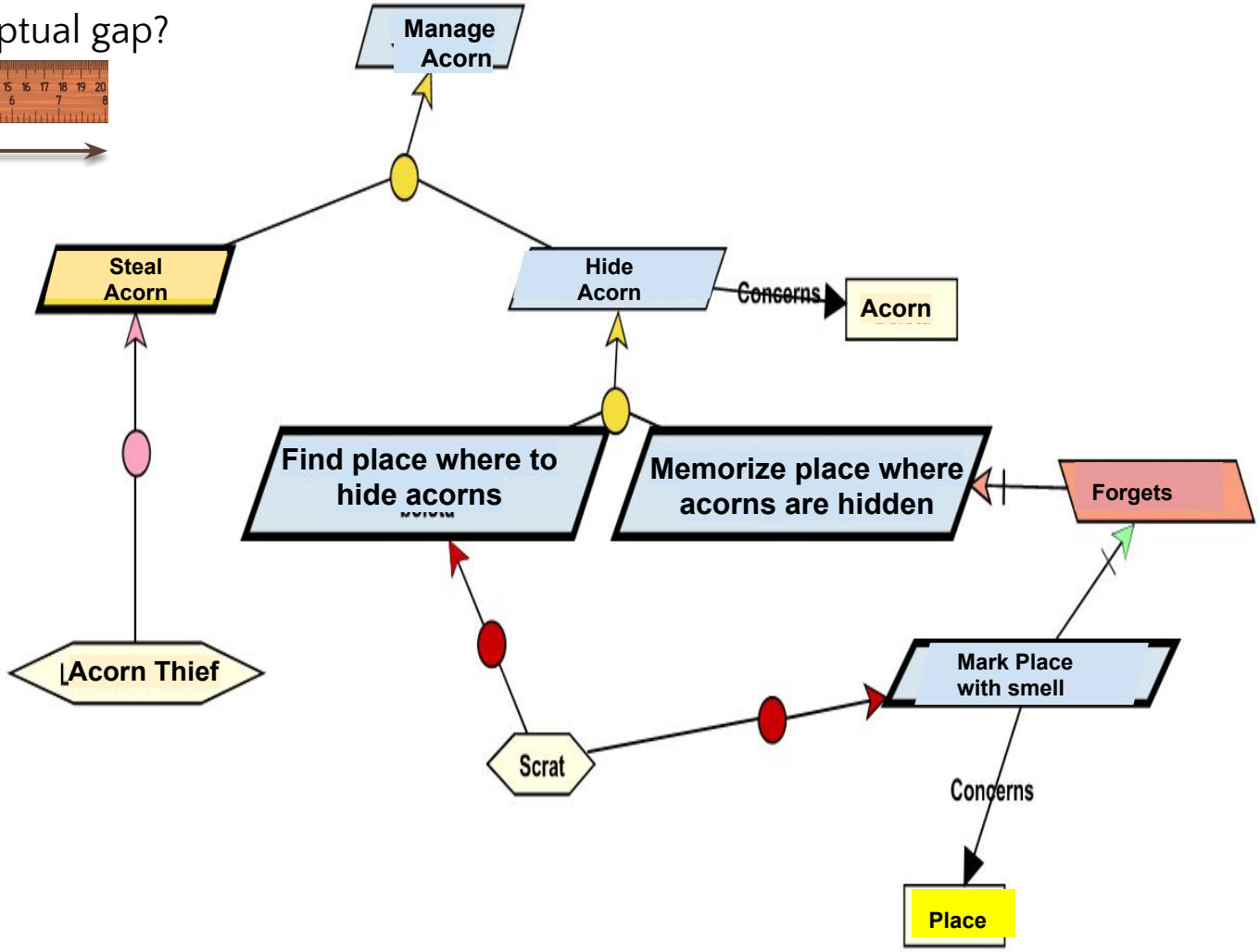
- Domain engineering
- Application engineering

Non-technical cost drivers

- Organization
- Social
- Ownership

The language has to empower its users... or they will end up using something else





Bad language?

Language named Brainf*ck

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..+++.>>.<-.<.+++.
------.--------.>>+.>++.
```

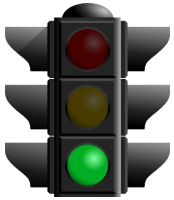
Outputs “HELLO WORLD”

Character	Meaning
`>`	increment the data pointer (to point to the next cell to the right).
`<`	decrement the data pointer (to point to the next cell to the left).
`+`	increment (increase by one) the byte at the data pointer.
`-`	decrement (decrease by one) the byte at the data pointer.
`.`	output the byte at the data pointer.
`,`	accept one byte of input, storing its value in the byte at the data pointer.
`[`	if the byte at the data pointer is zero, then instead of moving the instruction pointer forward to the next command, jump it *forward* to the command after the *matching* `]` command.
`]`	if the byte at the data pointer is nonzero, then instead of moving the instruction pointer forward to the next command, jump it *back* to the command after the *matching* `[` command.

Before we start...
an important disclaimer

- The remainder of this discussion should be placed in a context where the usability of a DSL and its **impact on the productivity** of its users are considered relevant
- No free lunch – engineering usability into your DSL **requires an investment**; make sure it will be returned, in the long run!

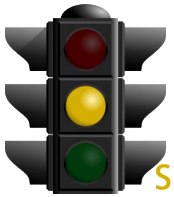
DSL usability evaluations are particularly relevant when:



- You are developing or adopting a DSL to be used by **other domain** users
  - As opposed to a DSL only to be used by its authors

- You are **claiming usability** improvements of your DSL compared to some existing alternative

DSL usability evaluations may not be so relevant when:



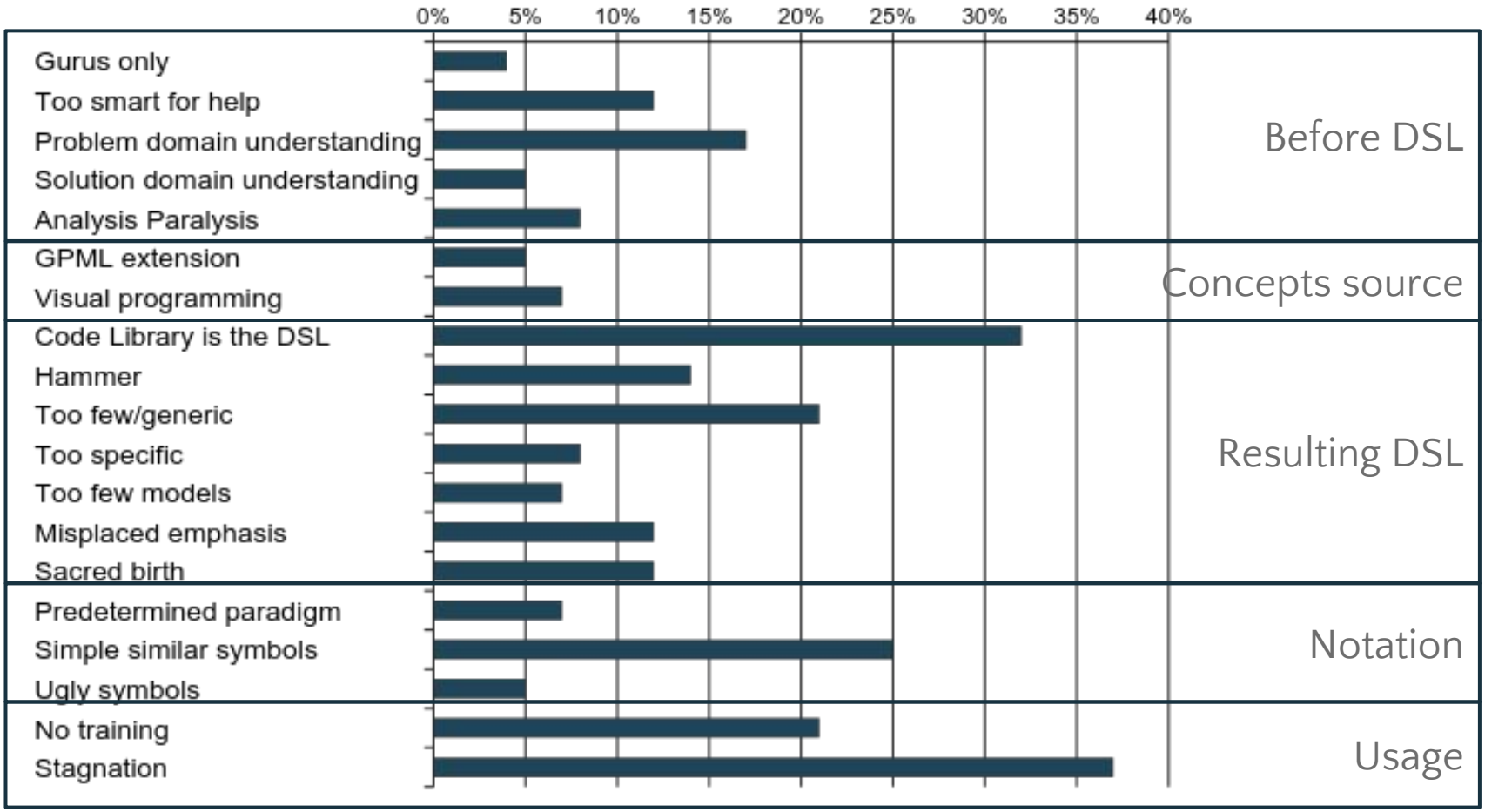
- You are developing (or adopting) a DSL
  - For personal/internal use
  - As part of a proof of concept
  - In other scenarios, where the actual usability of the DSL is not regarded as a priority concern (and it is also not being claimed)

What could possibly go wrong?

How **not** to build a DSL

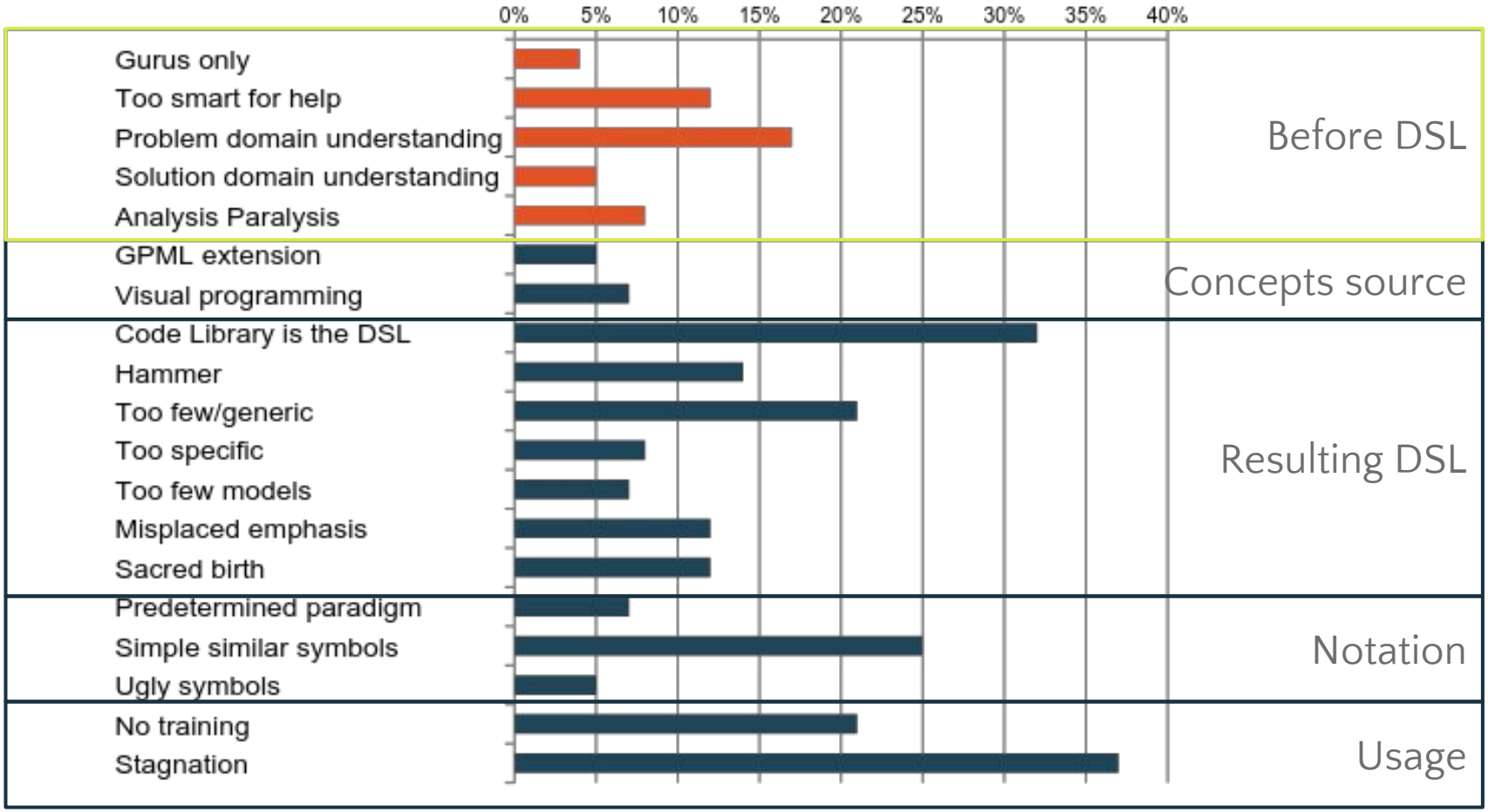
Common mistakes in DSMLs development and adoption

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)



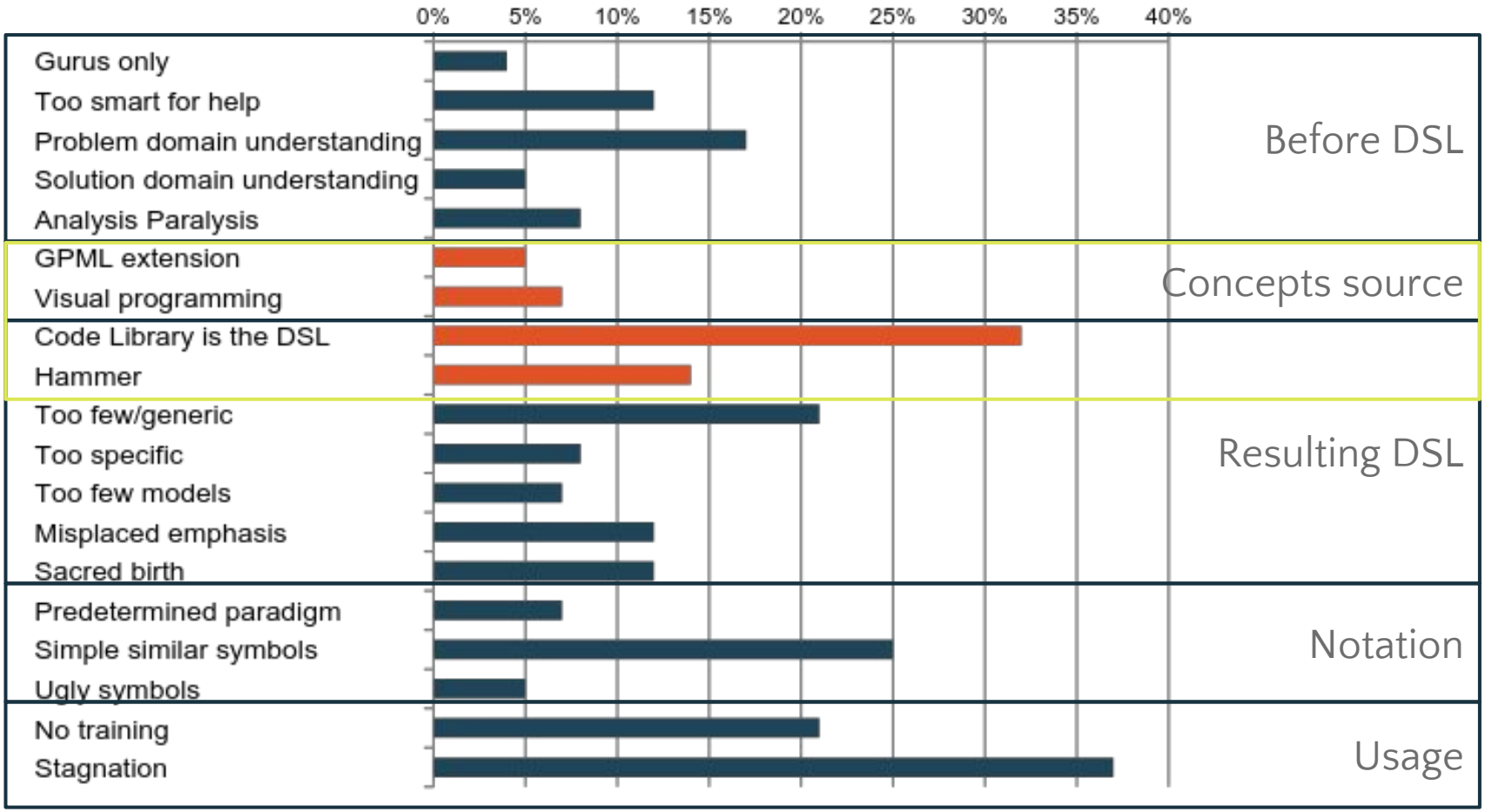
Ignore domain users as much as possible

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)



Use the wrong abstractions

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)

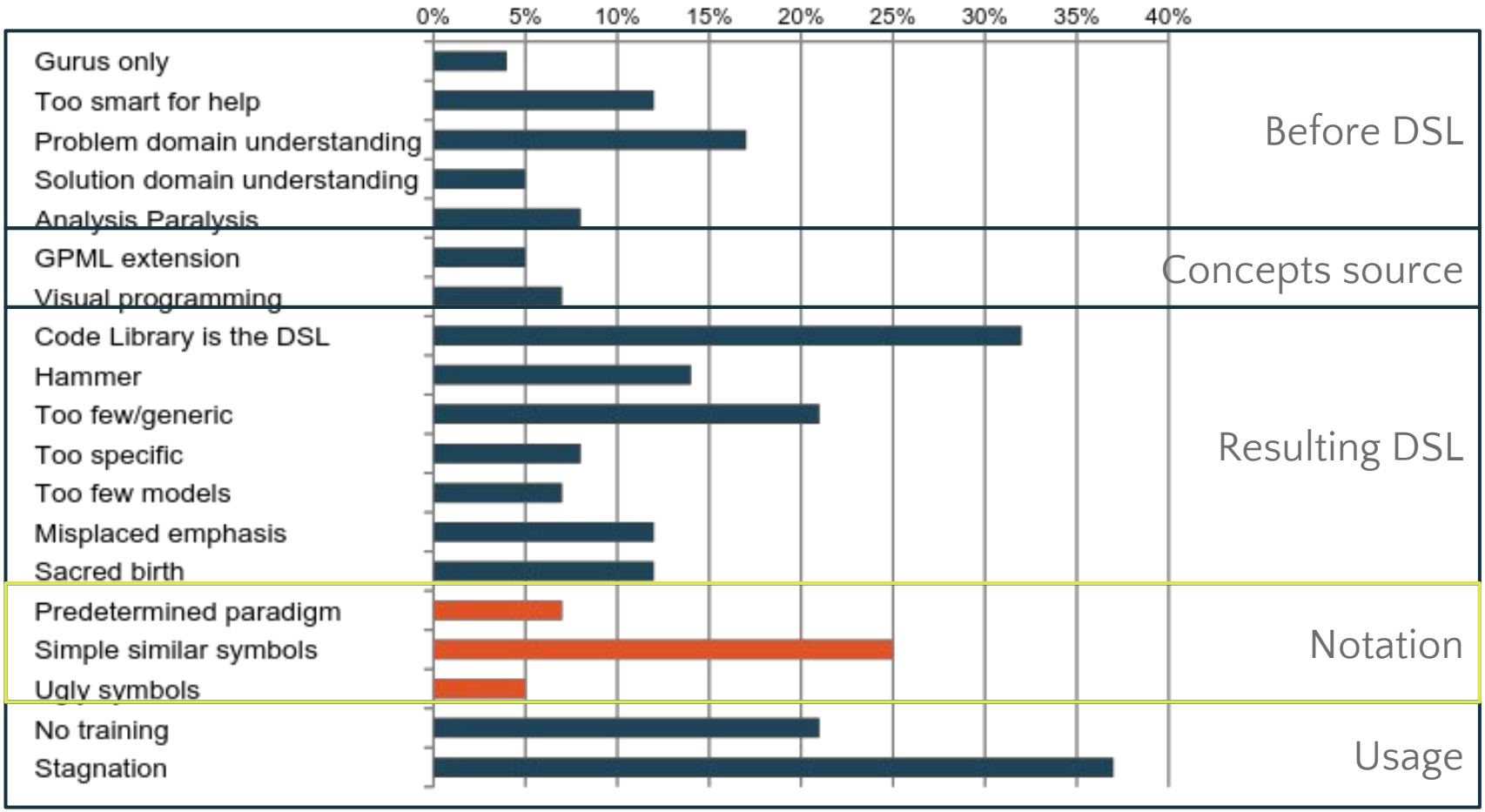


Miss the adequate abstraction level

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)

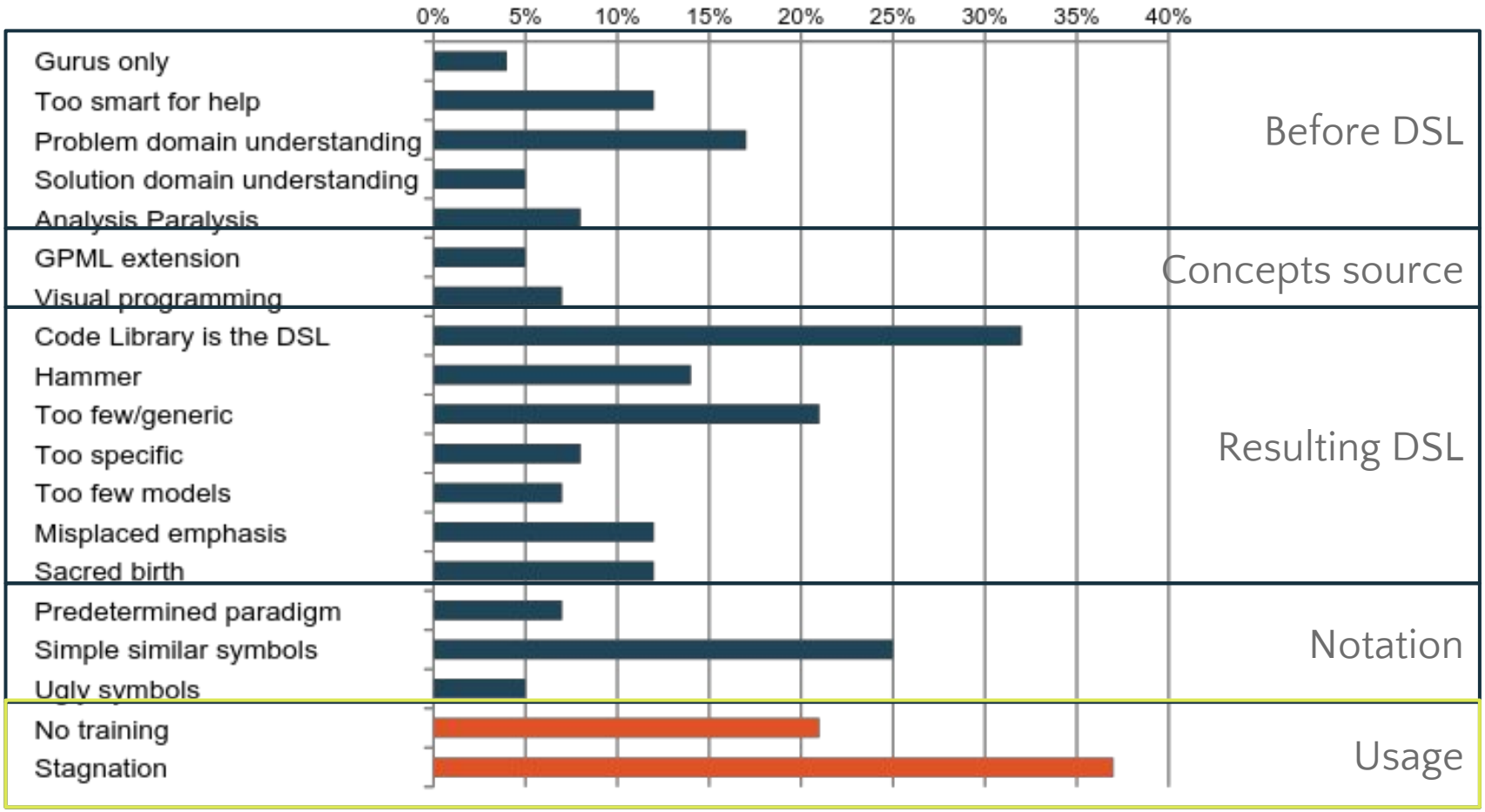
Create a bad concrete syntax

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)



Keep ignoring domain users

Steven Kelly, Risto Pohjonen, "Worst Practices for Domain-Specific Languages", IEEE Software, vol.6, n.4, 2009 (22-29)



Are users **the** problem?

Usability of DSLs

"The usability of the languages and libraries that developers use have a strong impact on their ability to successfully complete a set of development tasks."

Steven Clarke, 2004

Why does DSL usability matter?

Loosely adapted from: Pamela Fox, "The developer Experience", Atlassian Summit, 2012

Bad experience with DSL

- Bare minimum usage
- Low barrier for leaving
- Detractor within the DSL users community

Good experience with DSL

- Innovative usage
- Strong incentive for successful adoption
- Champion within the DSL users community

Poor usability can easily destroy an otherwise technically sound DSL!
Usability encompasses both language and tool support.

So, how often is DSL usability studied?

(highlights from a Systematic Literature Review)

Gabriel, Goulão and Amaral. "Do Software Languages Engineers Evaluate their Languages?", CIbSE'2010, April 2010.

- Very few papers report any sort of evaluation
  - Even those provide too few details
  - Too much tacit knowledge: virtually impossible to replicate evaluations and perform meta-analysis
- Predominance of toy examples
  - Unsubstantiated claims to the merits of DSLs
- Poor characterization of subjects involved in validations
  - Unknown representativeness

So, how often is DSL usability studied?

(highlights from a Systematic Literature Review)

Gabriel, Goulão and Amaral. "Do Software Languages Engineers Evaluate their Languages?", CIbSE'2010, April 2010.

- No evidence of widely adopted DSL validation strategies, with respect to usability
  - This does not necessarily mean practitioners do not assess their DSLs or that claims are fake
  - It does mean usability/productivity claims are not supported by evidence made available to the software development community

Wrong message is being sent to (*naïve*) DSL engineers:

Increased usability (or, to be more precise, its impact on DSL users' productivity while developing solutions with a DSL, when compared to an existing baseline) is assumed to be a given property of DSL based development.

So, let us build usability into DSLs!

How do we know we didn't miss the target? How do we evaluate?

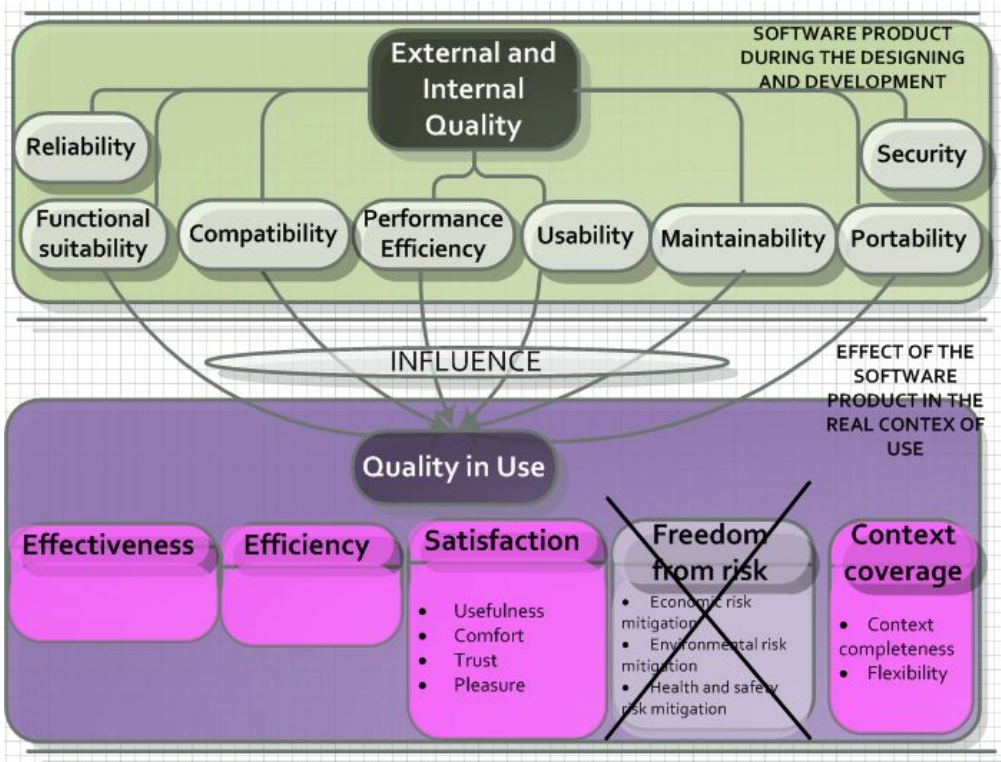
Usability (or Quality in Use) according to ISO 9126

Usability is defined as the degree to which a product or system can be used by specified users to achieve specific goals with:

- **effectiveness**
- **efficiency**
- **satisfaction**

in a specified context

of use.



What strategies are available to us?

- Constructive approaches
  - Our own expertise and common sense
  - Usability heuristics such as the “Physics of notations”
  - Use the wisdom of the crowds
- Evaluation-based (Empirical Studies) approaches
  - “Traditional” DSL usability evaluations
  - User monitoring while using the DSL

Use our expertise and common sense?

Sure. But that is probably not enough, otherwise we would not have this problem.

Why is our expertise not enough?

- Like with any other software, neglecting the importance of DSL **users** is a recipe for disaster
- The DSL is usually defined with the inputs of **domain experts**
- The DSL is used by **domain users**
- **Domain users** **<u>are not necessarily</u>** **domain experts**
- If **domain users** are **<u>not</u>** committed in the DSL development process, important usability shortcomings may be detected too late!

The DSL must be usable by its "normal" <u>users</u>, rather than by experts, or by language engineers!

What strategies are available to us?

- Constructive approaches
  - Our own expertise and common sense
  - Usability heuristics such as the “Physics of notations”
  - Use the wisdom of the crowds
- Evaluation-based (Empirical Studies) approaches
  - “Traditional” DSL usability evaluations
  - User monitoring while using the DSL

Use language design heuristics?

Caire, Genon, Heymans, & Moody, "Visual notation design 2.0: towards user comprehensible requirements engineering notations". Proceedings of the *21st IEEE International Requirements Engineering Conference (RE), 2013*. (pp. 115–124).

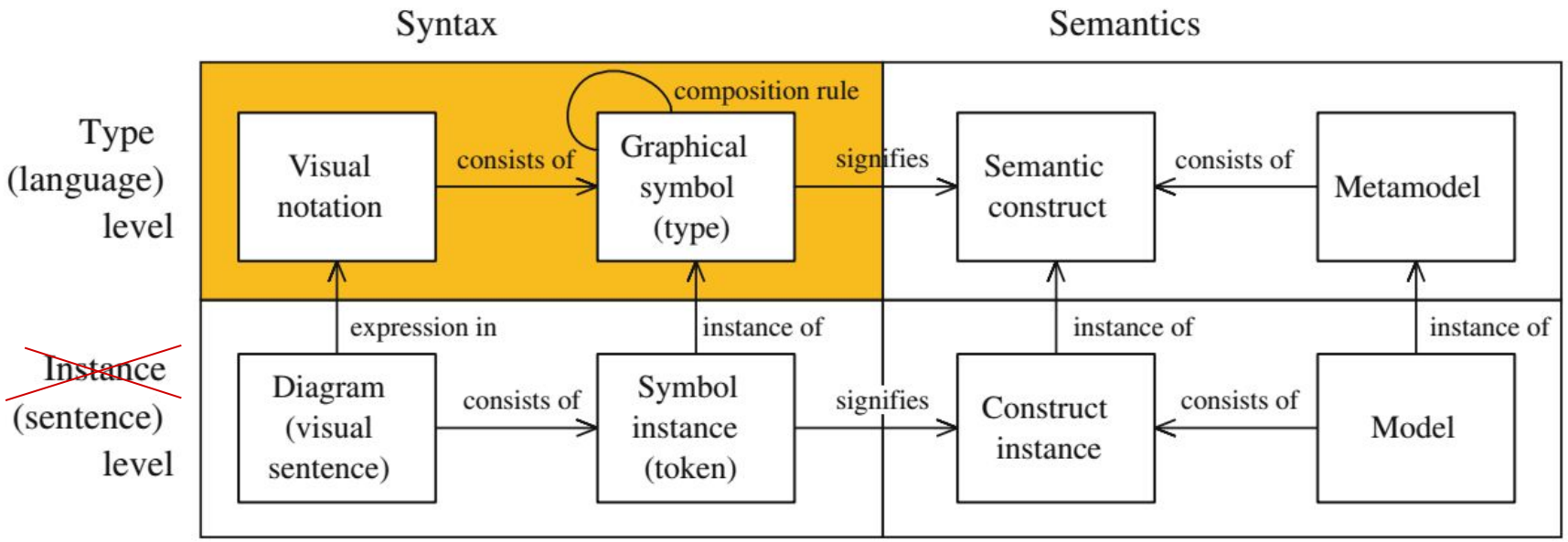
The anatomy of a visual notation

The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering

Daniel L. Moody, *Member*, *IEEE*

Abstract—Visual notations form an integral part of the language of software engineering (SE). Yet historically, SE researchers and notation designers have ignored or undervalued issues of visual representation. In evaluating and comparing notations, details of visual syntax are rarely discussed. In designing notations, the majority of effort is spent on semantics, with graphical conventions largely an afterthought. Typically, no design rationale, scientific or otherwise, is provided for visual representation choices. While SE has developed mature methods for evaluating and designing semantics, it lacks equivalent methods for visual syntax. This paper defines a set of principles for designing cognitively effective visual notations: ones that are optimized for human communication and problem solving. Together these form a design theory, called the Physics of Notations as it focuses on the physical (perceptual) properties of notations rather than their logical (semantic) properties. The principles were synthesized from theory and empirical evidence from a wide range of fields and rest on an explicit theory of how visual notations communicate. They can be used to evaluate, compare, and improve existing visual notations as well as to construct new ones. The paper identifies serious design flaws in some of the leading SE notations, together with practical suggestions for improving them. It also showcases some examples of visual notation design excellence from SE and other fields.

The anatomy of a visual notation

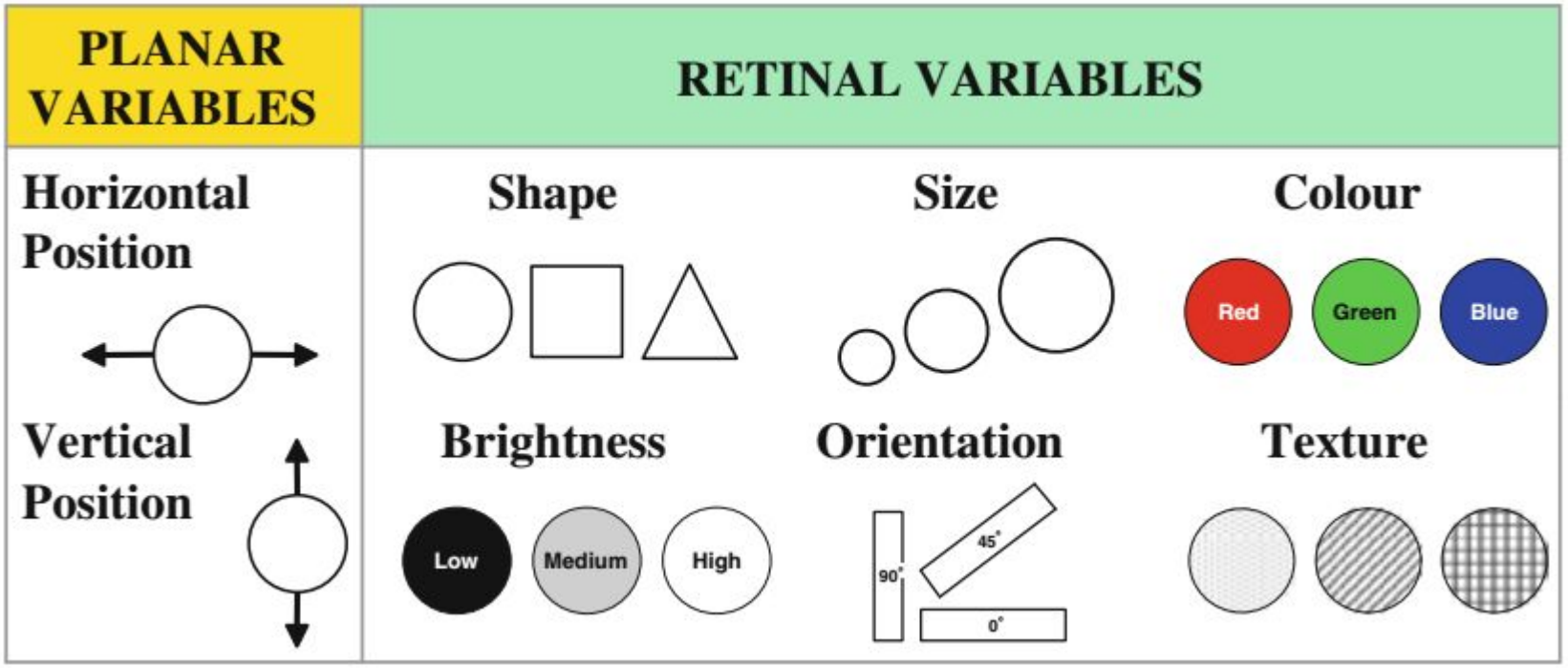


What makes a good visual notation?

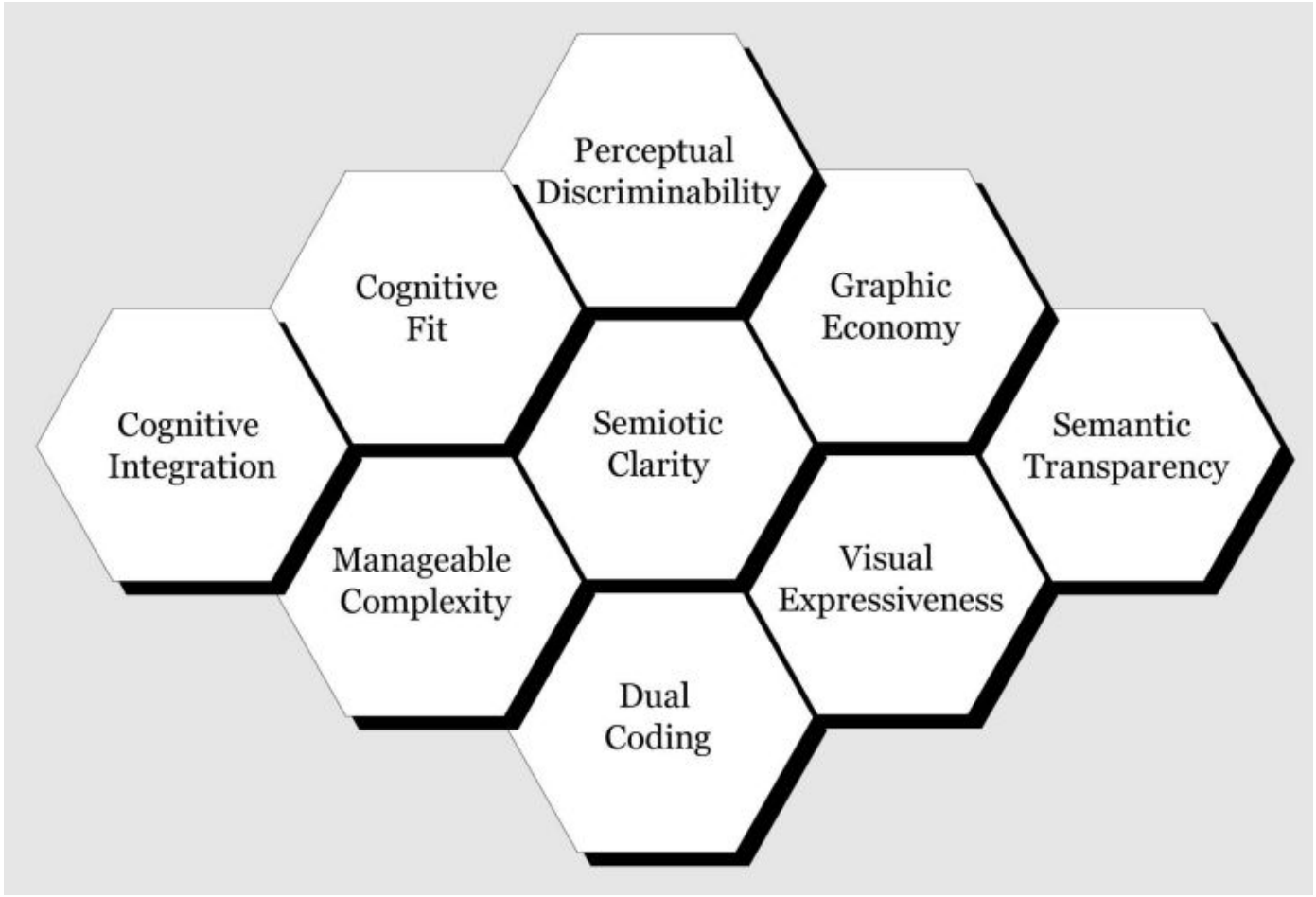
- **Cognitive effectiveness:**
  - the ***speed***, ***ease*** and ***accuracy*** with which a representation can be processed by the human mind
- Cognitive effectiveness should be **planned**, **implemented** and **evaluated**

Key design goal: to improve the cognitive effectiveness of the visual notation

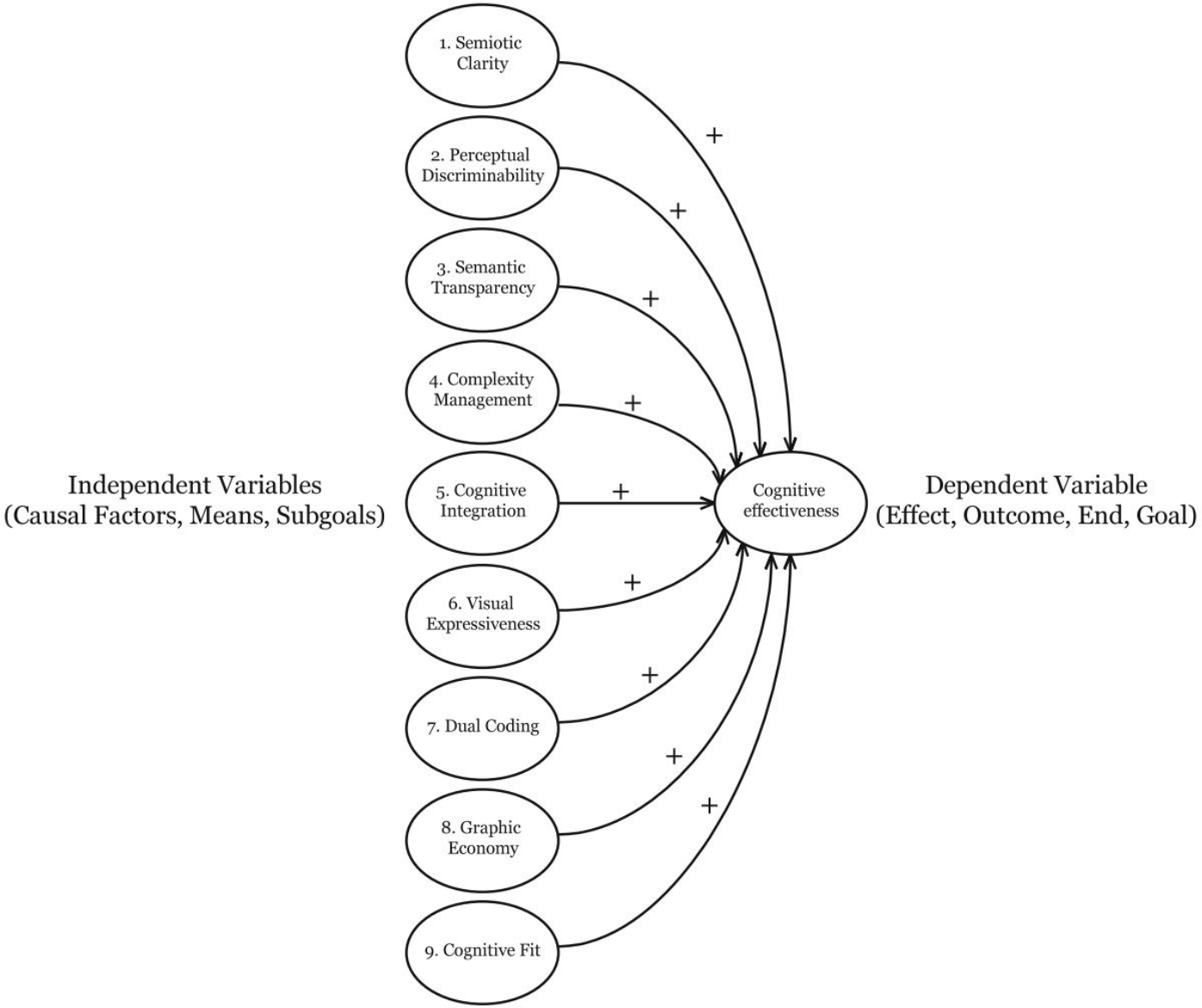
What is our graphic design space?



The principles supporting the cognitive effectiveness of visual languages



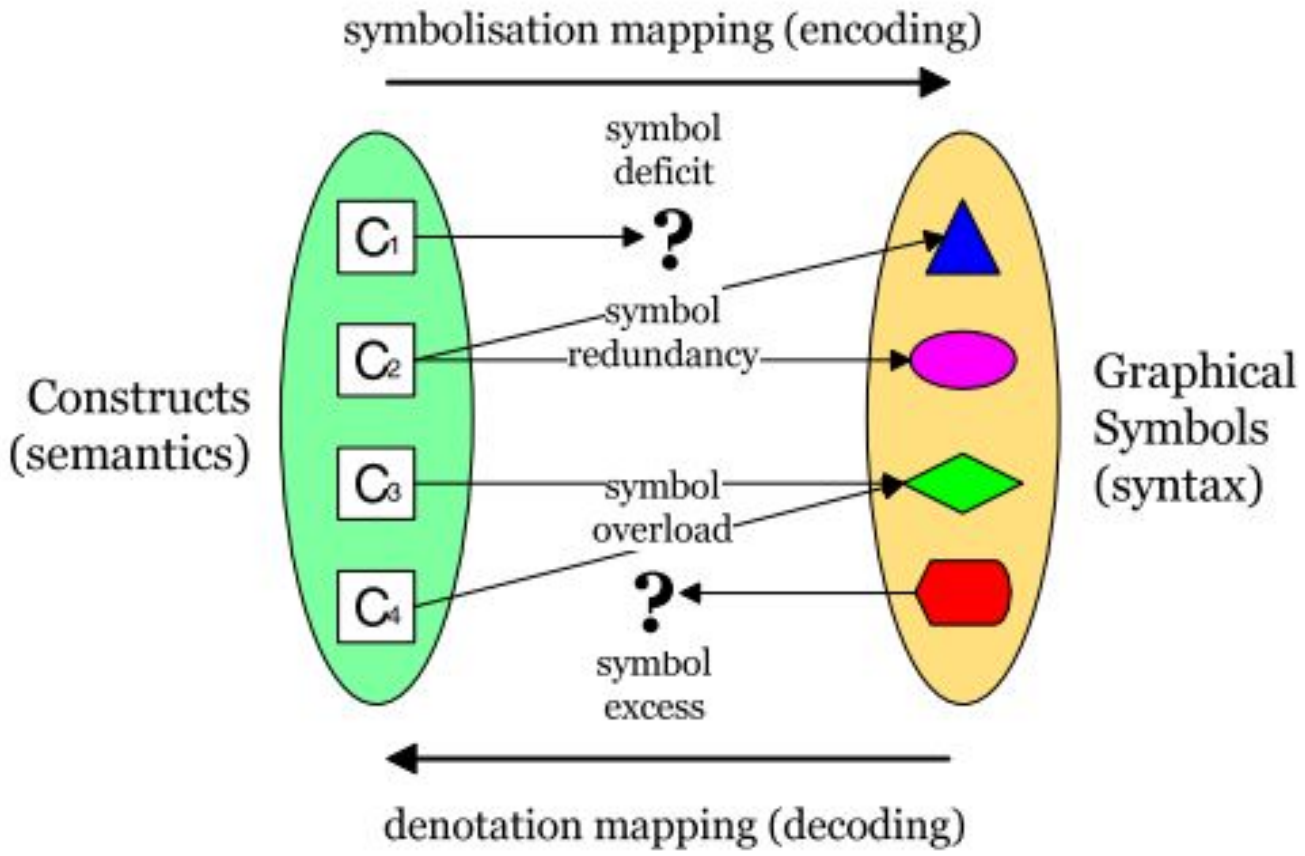
A causal model for cognitive effectiveness



Principles for the Physics of Notations

- 1. **Semiotic Clarity**: there should be a 1:1 correspondence between semantic constructs and graphical symbols
  2. **Perceptual Discriminability**: symbols should be clearly distinguishable from one another
  3. **Semantic Transparency**: use symbols whose appearance suggests their meaning
  4. **Complexity Management**: include explicit mechanisms for dealing with complexity
  5. **Cognitive Integration**: include explicit mechanisms to support integration of information from different diagrams
  6. **Visual Expressiveness**: use the full range and capacities of visual variables
  7. **Dual Coding**: use text to complement graphics
  8. **Graphic Economy**: keep the number of different graphical symbols cognitively manageable
  9. **Cognitive Fi**t: use different visual dialects for different tasks and/or audiences

Semiotic clarity: there should be a 1:1 correspondence between semantic constructs and graphical symbols

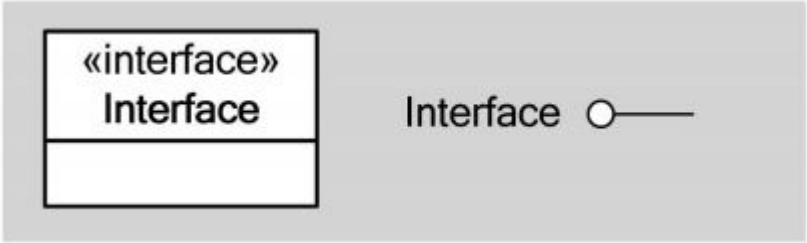


Semiotic clarity: there should be a 1:1 correspondence between semantic constructs and graphical symbols

- **Symbol deficit:** when a semantic construct is not represented by any symbol

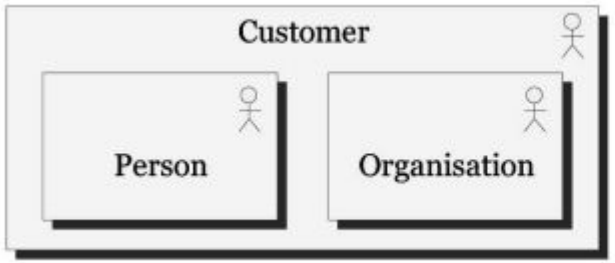
Semiotic clarity: there should be a 1:1 correspondence between semantic constructs and graphical symbols

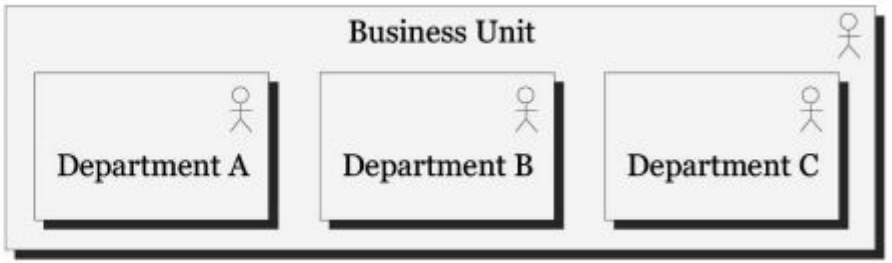
- **Symbol redundancy**: a semantic construct is represented by multiple symbols
  - Synographs: alternative representations (the equivalent to textual synonyms) for the same semantic construct



Semiotic clarity: there should be a 1:1 correspondence between semantic constructs and graphical symbols

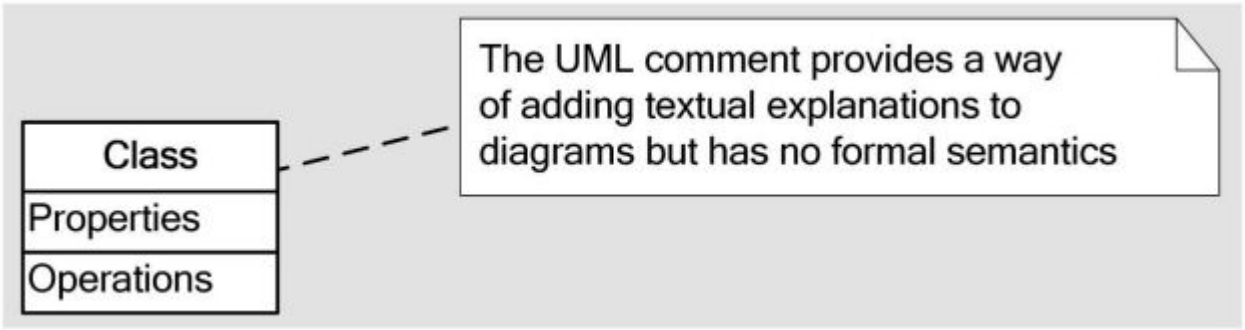
- **Symbol overload:** the same symbol is used to represent multiple constructs
  - Symbol overload (homographs) in ArchiMate: The same graphical convention can be used to represent different types of relationships:





Semiotic clarity: there should be a 1:1 correspondence between semantic constructs and graphical symbols

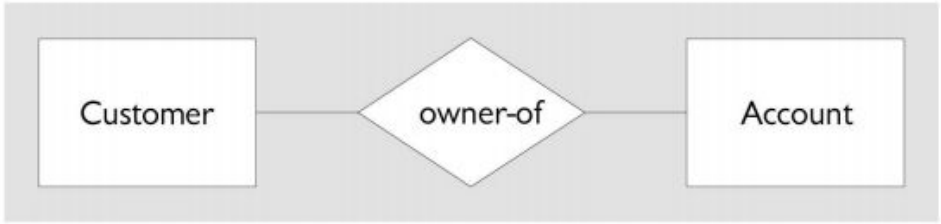
- **Symbol excess:** symbol does not represent any semantic construct
  - In UML a comment is a useful notational feature but has no formal semantics



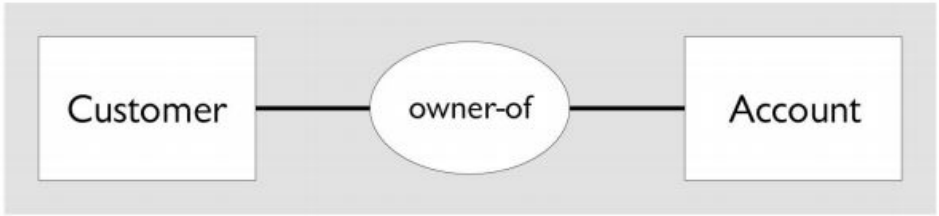
Perceptual Discriminability: symbols should be clearly distinguishable from one another

- **Visual distance:** number of visual variables which are different and the size of those differences

bad



better

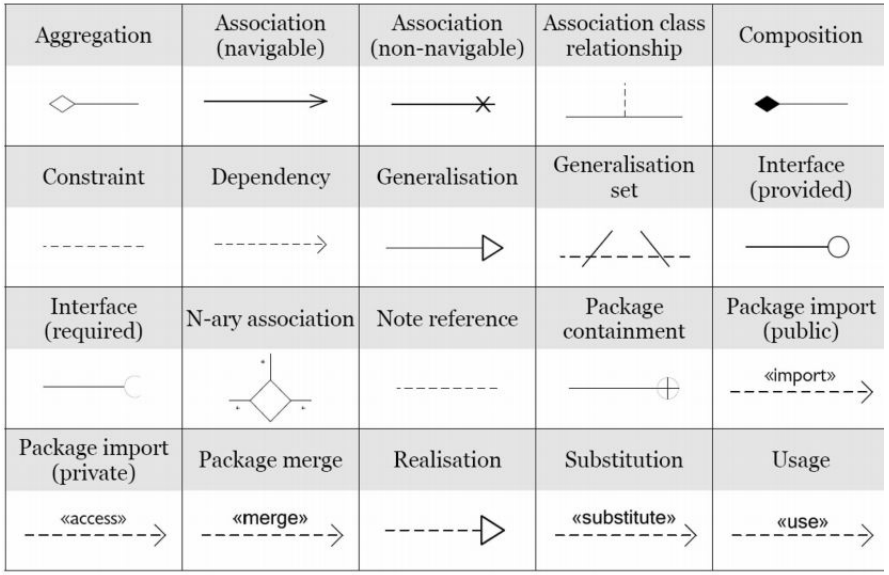


Perceptual Discriminability: symbols should be clearly distinguishable from one another

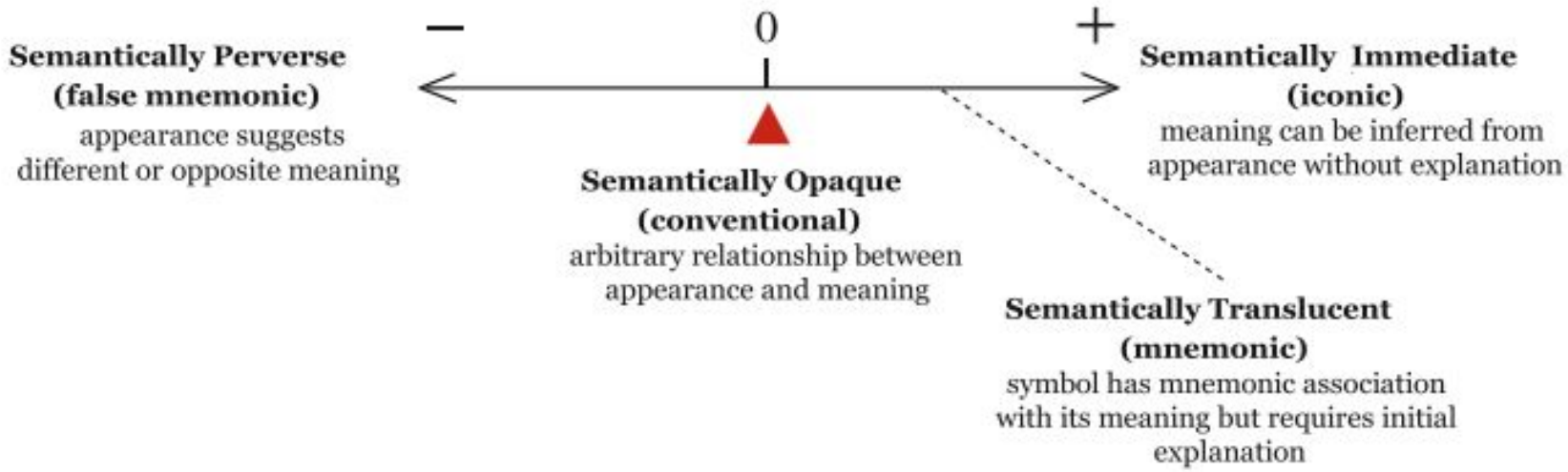
- **Redundant encoding: use multiple visual variables (e.g. shape and color)**

Perceptual Discriminability: symbols should be clearly distinguishable from one another

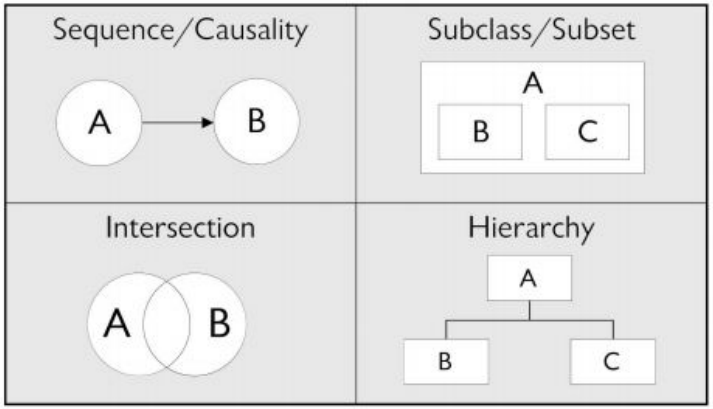
- **Perceptual pop out**: visual elements should be uniquely distinguishable by at least one visual property, rather than a combination of properties
- **Textual differentiation:** notations should not rely on text to distinguish among different symbols

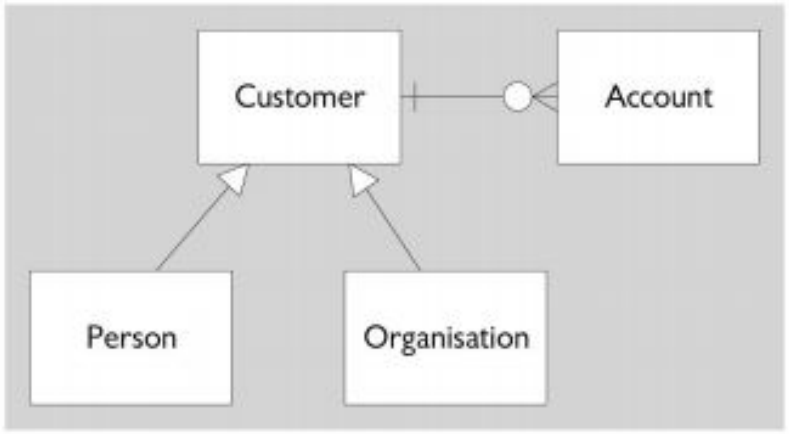


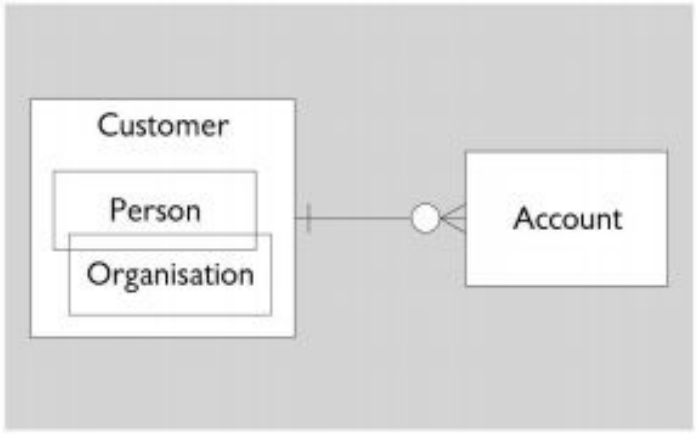
Semantic transparency: Use Visual Representations Whose Appearance Suggests Their Meaning



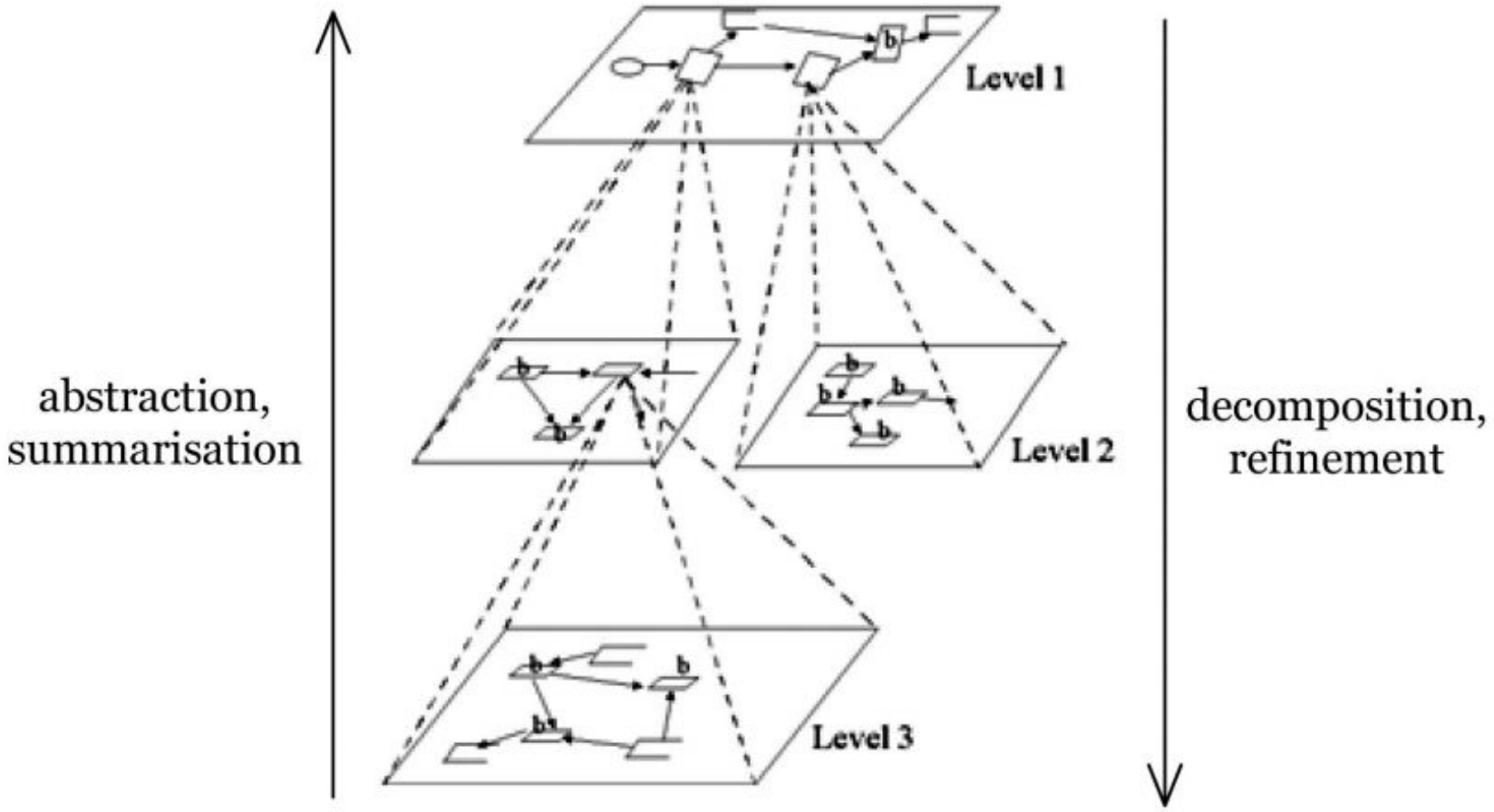
Semantic transparency: Use Visual Representations Whose Appearance Suggests Their Meaning



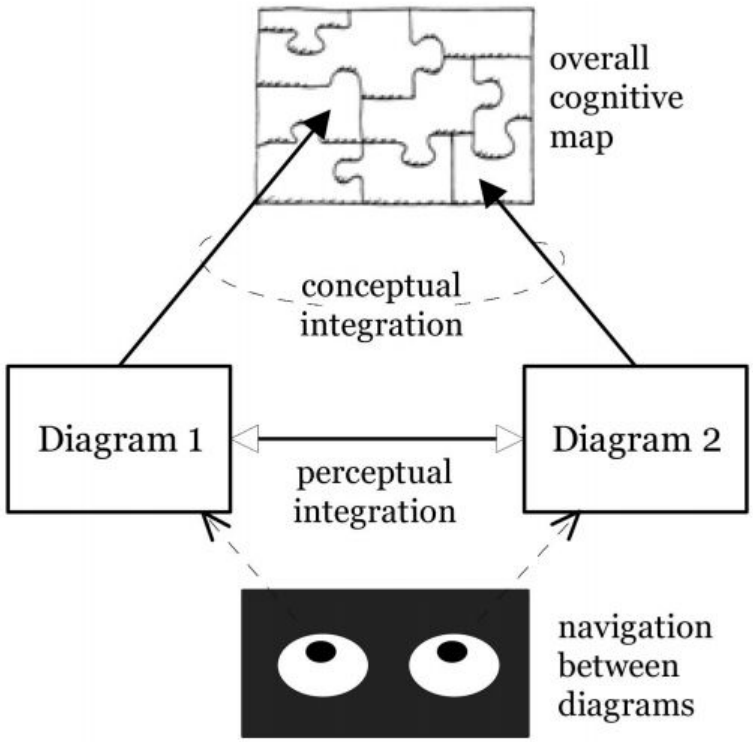


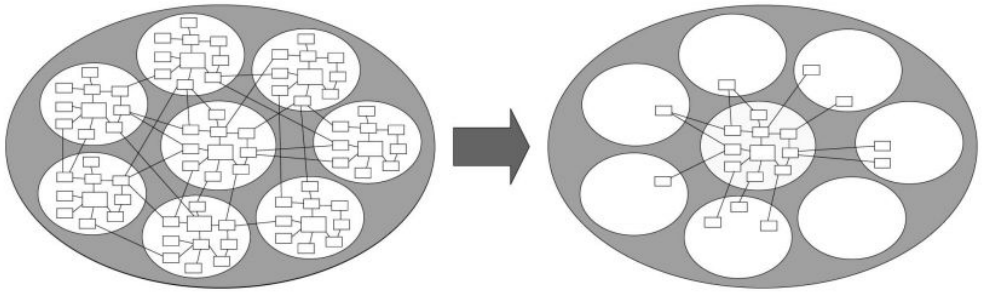


Principle of Complexity Management: Include Explicit Mechanisms for Dealing with Complexity



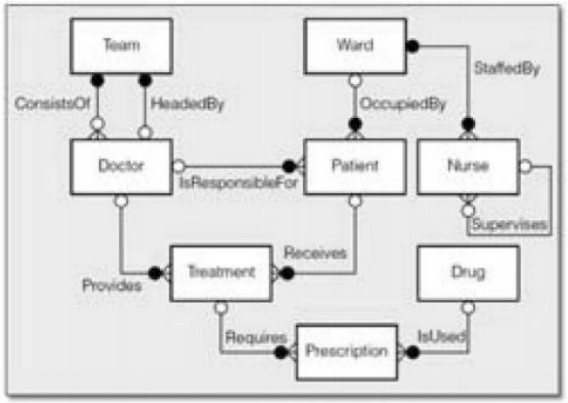
Principle of Cognitive Integration: Include Explicit Mechanisms to Support Integration of Information from Different Diagrams

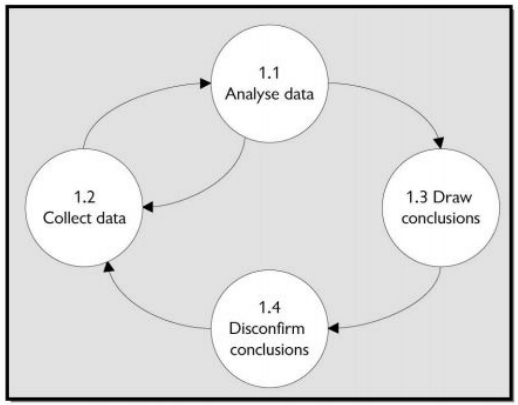


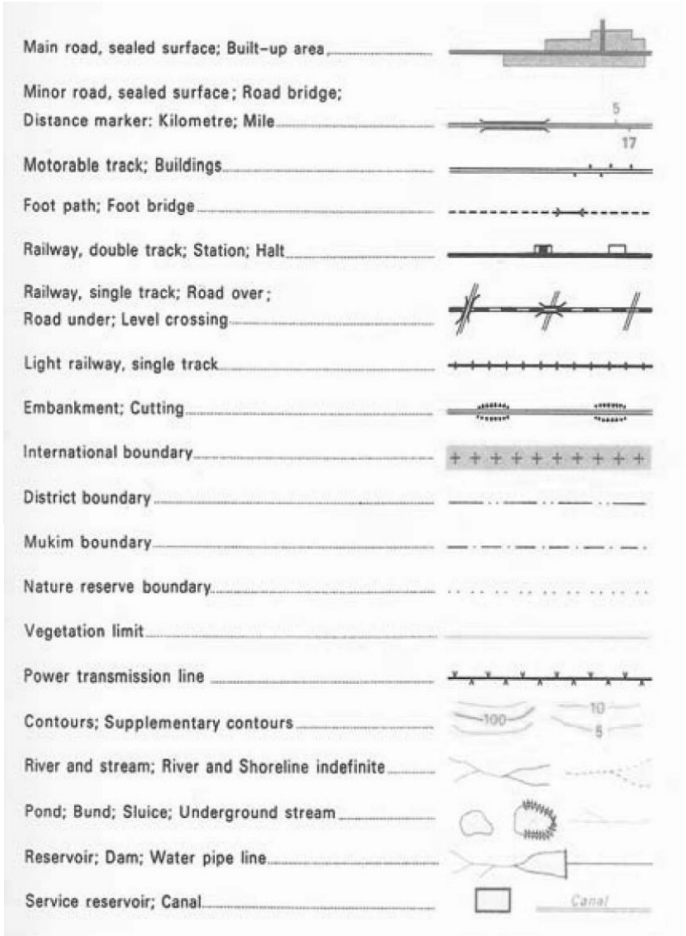


Principle of visual expressiveness: Use the Full Range and Capacities of Visual Variables

- Visual expressiveness is defined as the number of visual variables used in a notation







Principle of visual expressiveness: Use the Full Range and Capacities of Visual Variables

- Visual expressiveness is defined as the number of visual variables used in a notation

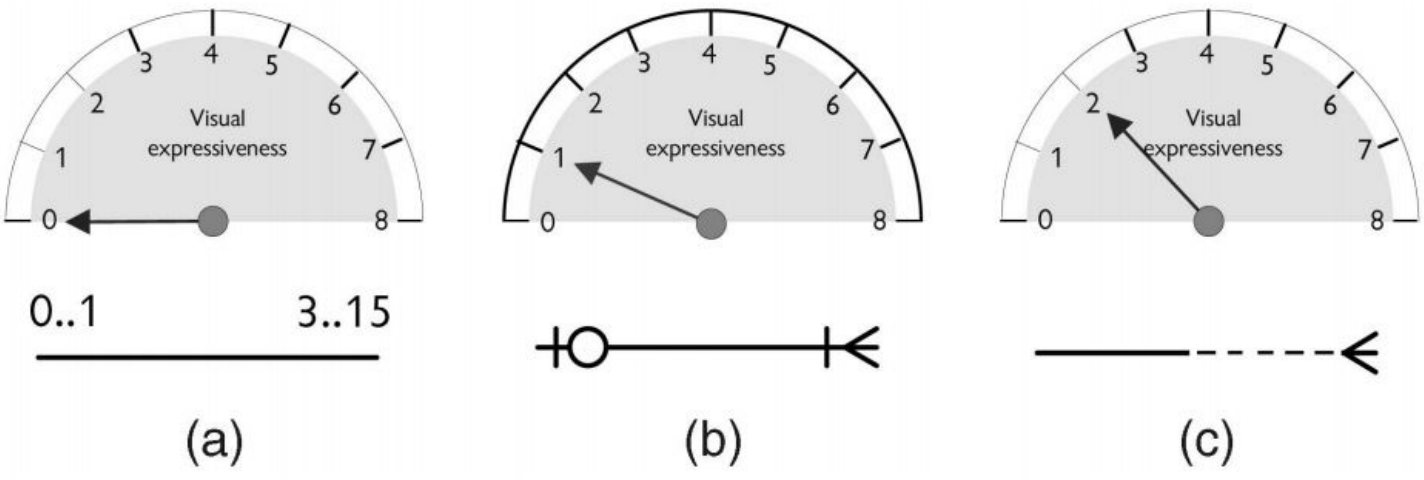


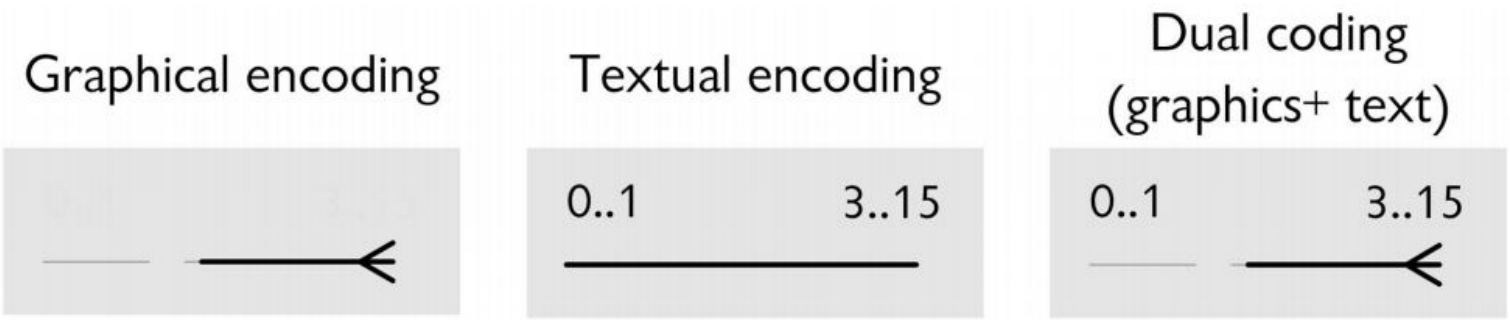
Fig. 31. Levels of visual expressiveness: (a) UML = 0, (b) IE = 1 (shape), and (c) Oracle = 2 (shape, brightness).

Beware of different encoding capabilities

Variable	Power	Capacity
Horizontal position (x)	Interval	10-15
Vertical position (y)	Interval	10-15
Size	Interval	20
Brightness	Ordinal	6-7
Colour	Nominal	7-10
Texture	Nominal	2-5
Shape	Nominal	Unlimited
Orientation	Nominal	4

Principle of Dual Coding: Use Text to Complement Graphics

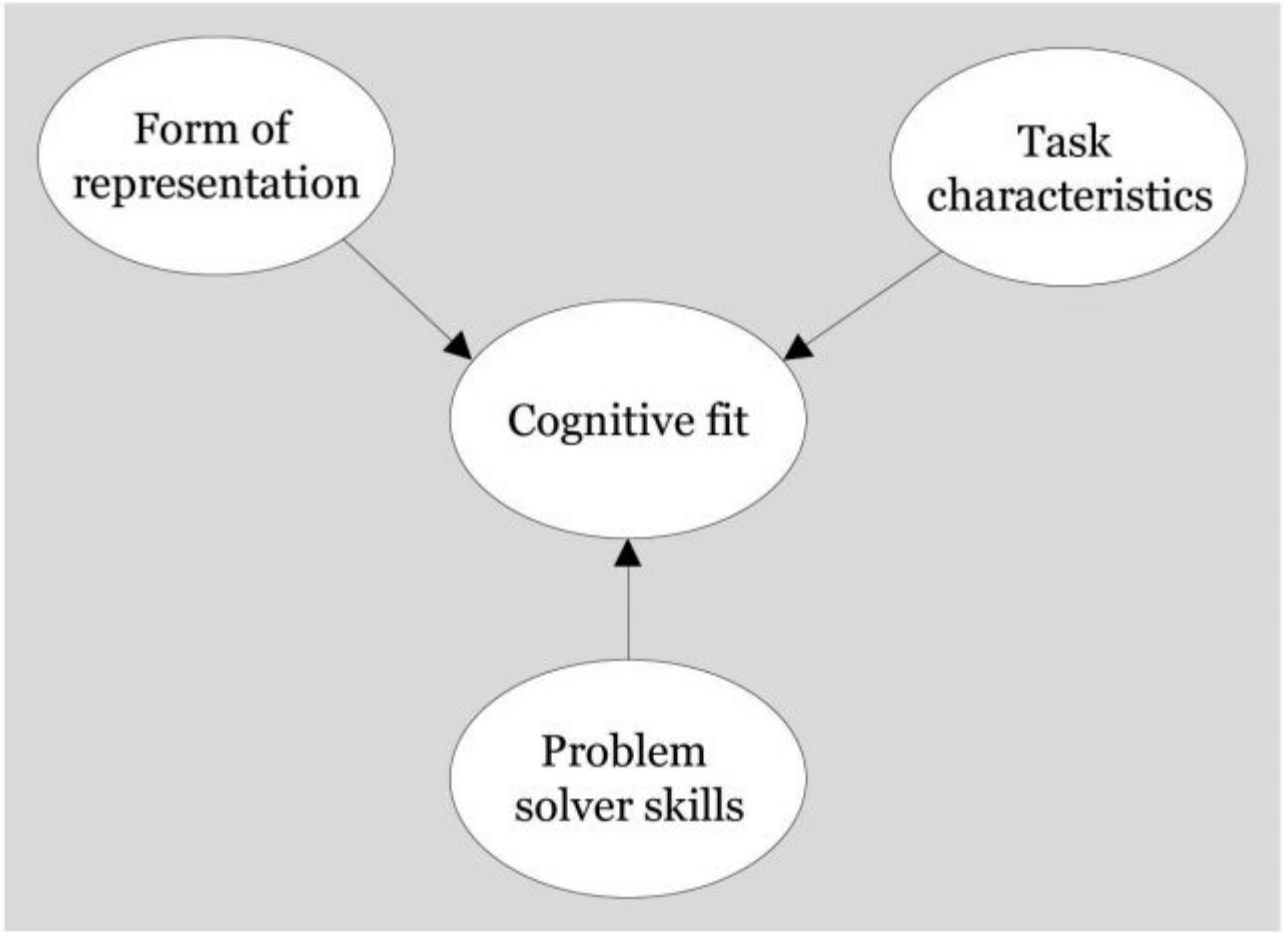
- Do not use text to replace graphics – instead, annotate them for increased expressiveness



Principle of Graphic Economy: The Number of Different Graphical Symbols Should Be Cognitively Manageable

- Reduce semantic complexity.
- Introduce symbol deficit.
- Increase visual expressiveness.

Principle of Cognitive Fit: Use Different Visual Dialects for Different Tasks and Audiences



What strategies are available to us?

- Constructive approaches
  - Our own expertise and common sense
  - Usability heuristics such as the “Physics of notations”
  - Use the wisdom of the crowds
- Evaluation-based (Empirical Studies) approaches
  - “Traditional” DSL usability evaluations
  - User monitoring while using the DSL

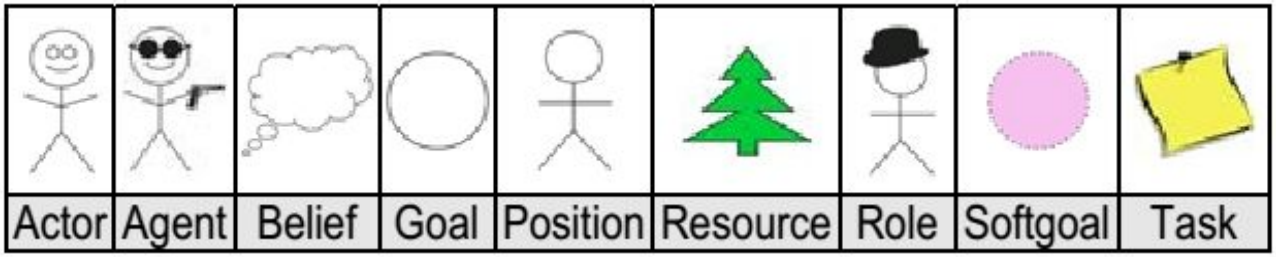
Case study

Caire, Genon, Heymans, & Moody, "Visual notation design 2.0: towards user comprehensible requirements engineering notations". Proceedings of the *21st IEEE International Requirements Engineering Conference (RE), 2013.* (pp. 115–124).

Example: revamping i*

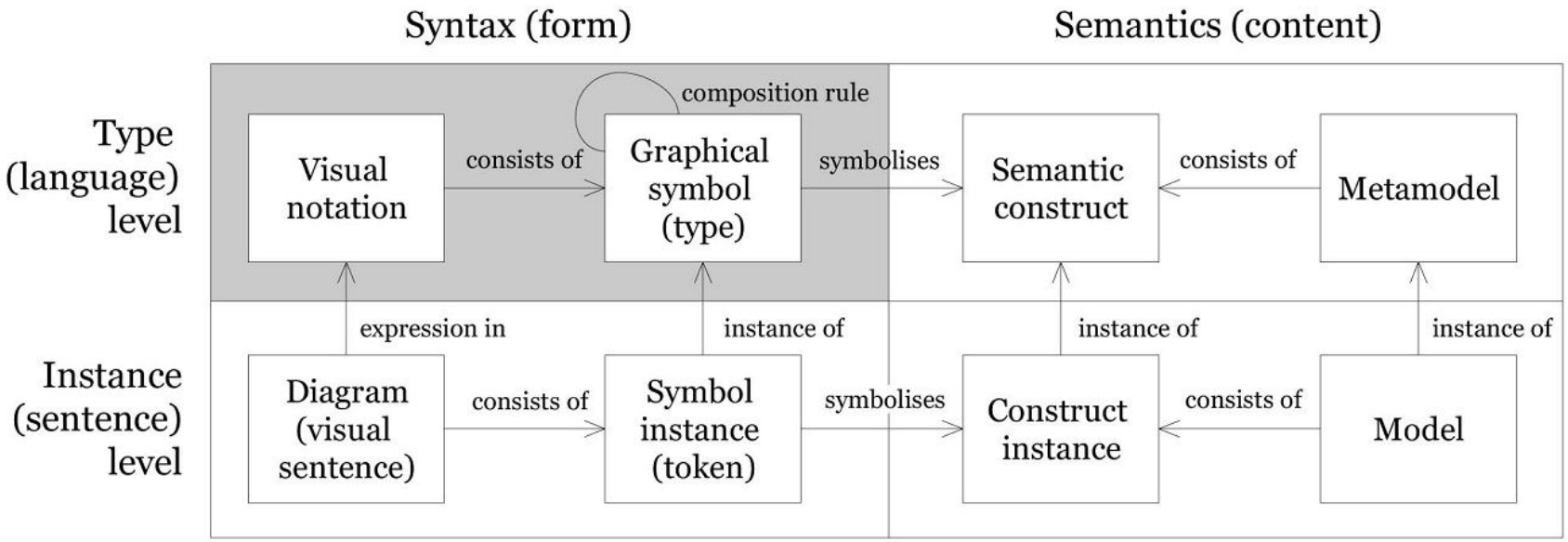
- i* (original concrete syntax)
  - produced by experts using intuition – unself conscious design

- i* (revamped concrete syntax, based on the Physics of notation)

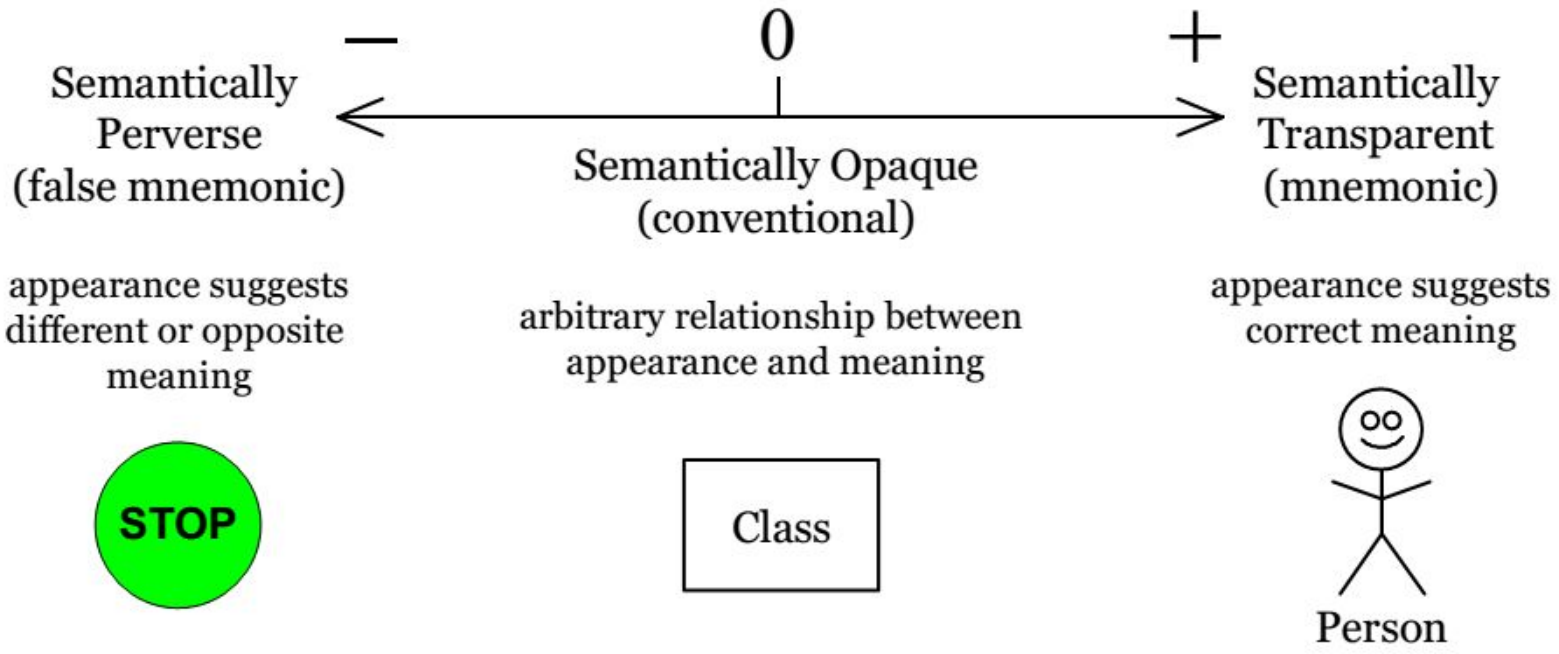


Does it work?

Scoping the study



Study focus on semantic transparency

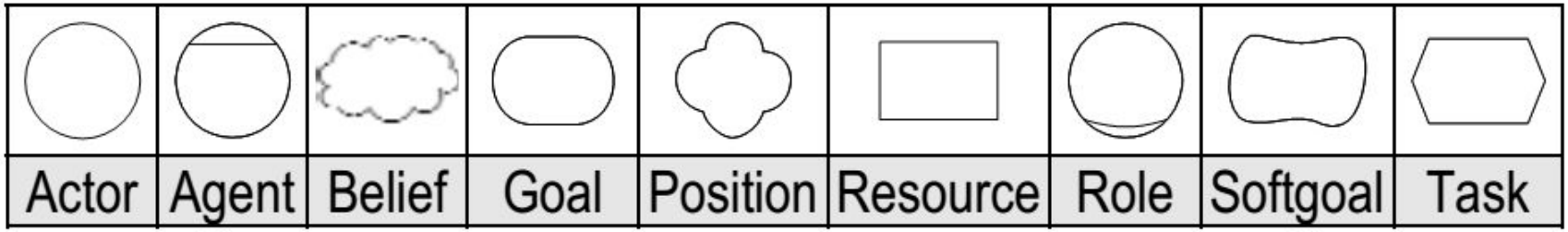


Semantic transparency coefficient

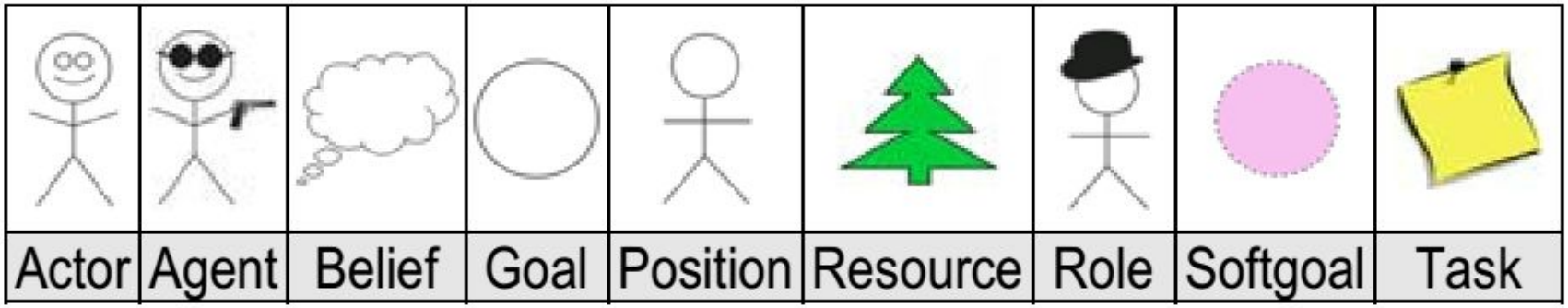
- Intuition – similar to a correlation coefficient
- Computed with: $$\frac{maximum\ frequency - expected\ frequency}{total\ responses - expected\ frequency}$$
  - where:
    - Maximum frequency – hits on the symbol
    - Expected frequency – number of potential hits by chance (participants/choices)
    - Total responses – number of identification attempts
- Defined in the interval [–1, 1]
  - where:
    - –1 means semantically perverse
    - 1 means semantically transparent

Which notation is the best? (i)

- Standard i*
  - produced by experts using intuition – unselfconscious design

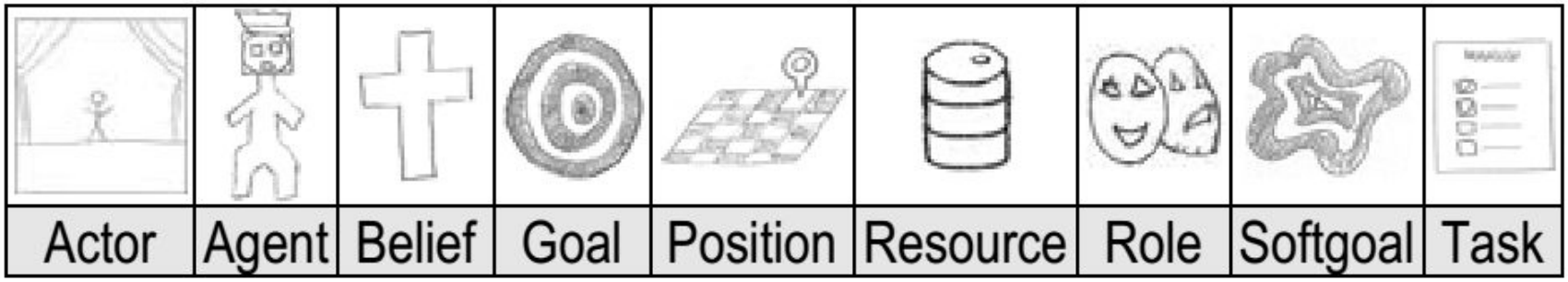


- PoN i*

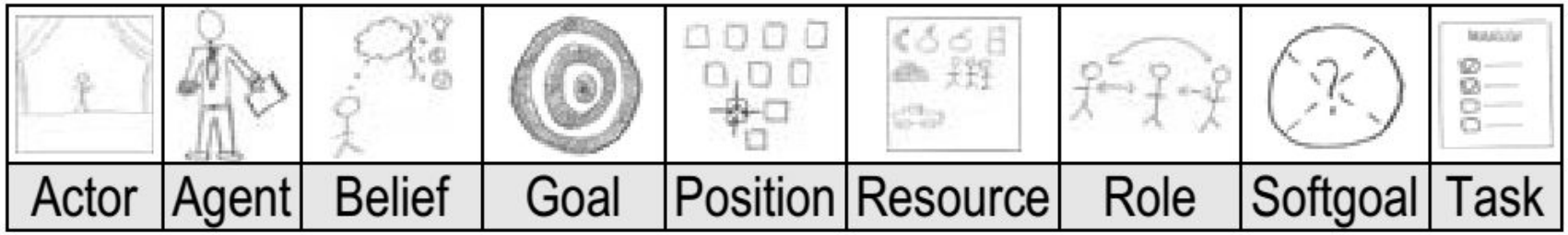


Which notation is the best? (ii)

- Stereotype i*
  - Most common symbols produced by novices

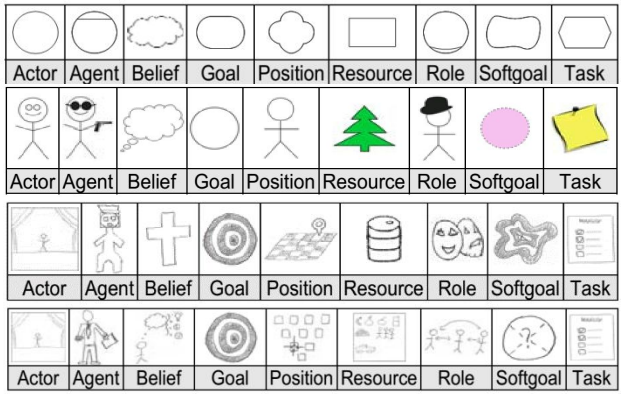


- Prototype symbol set i*
  - Selected by novices



Variables and hypothesis

- Independent variable
  - Symbol set



- Dependent variables
  - Hit rate (% of recognized symbols)
  - Semantic transparency coefficient

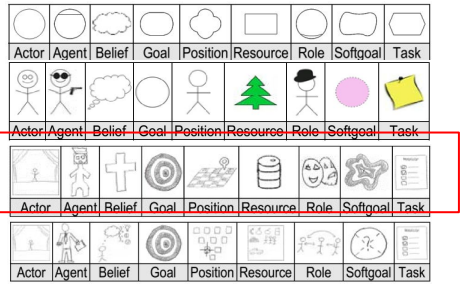
$$\frac{maximum\ frequency - expected\ frequency}{total\ responses - expected\ frequency}$$

- Hypothesis
  - Prototype > Stereotype > PoN > Standard i*

Which notation maximizes hit rate?

TABLE I. HIT RATE ANALYSIS
(GREEN = ABOVE ISO THRESHOLD; UNDERLINE = BEST)

	Standard	PoN	Stereotype	Prototype
Actor	11.1%	37.5%	62.5%	43.8%
Agent	11.1%	37.5%	50.0%	37.5%
Belief	33.3%	43.8%	93.8%	31.3%
Goal	11.8%	31.3%	56.3%	31.3%
Position	5.6%	12.5%	43.8%	50.0%
Resource	11.1%	50.0%	75.0%	37.5%
Role	11.1%	43.8%	75.0%	43.8%
Softgoal	50.0%	12.5%	75.0%	50.0%
Task	11.1%	81.3%	75.0%	50.0%
Mean hit rate	**17.4%**	**38.9%**	**67.4%**	**41.7%**
Std dev	**14.5%**	**20.7%**	**15.6%**	**7.7%**
Group size (n)	**18**	**16**	**16**	**16**



Which notation maximizes semantic transparency?

TABLE II. SEMANTIC TRANSPARENCY COEFFICIENT RESULTS
(GREEN = TRANSPARENT; UNDERLINE = BEST)

	Standard	PoN	Stereotype	Prototype
Actor	-0.31	0.30	0.58	0.37
Agent	-0.19	0.30	0.39	0.30
Belief	0.25	0.37	0.83	0.23
Goal	-0.07	0.23	0.45	0.23
Position	-0.19	-0.30	0.33	0.44
Resource	-0.25	0.44	0.64	0.30
Role	-0.19	0.37	0.64	0.37
Softgoal	0.44	-0.16	0.64	0.44
Task	-0.13	0.79	0.64	0.44
Mean	**-0.07**	**0.26**	**0.57**	**0.34**
Std dev	**0.25**	**0.32**	**0.15**	**0.09**
≠ 0 (one sample t-test)	**Opaque (p = .419)**	**Transparent (p = 042*)**	**Transparent (p = .000***)**	**Transparent (p = .000***)**

Which notation has the best semantic transparency?

Caire et al., "Visual notation design 2.0: towards user comprehensible requirements engineering notations". RE 2013.

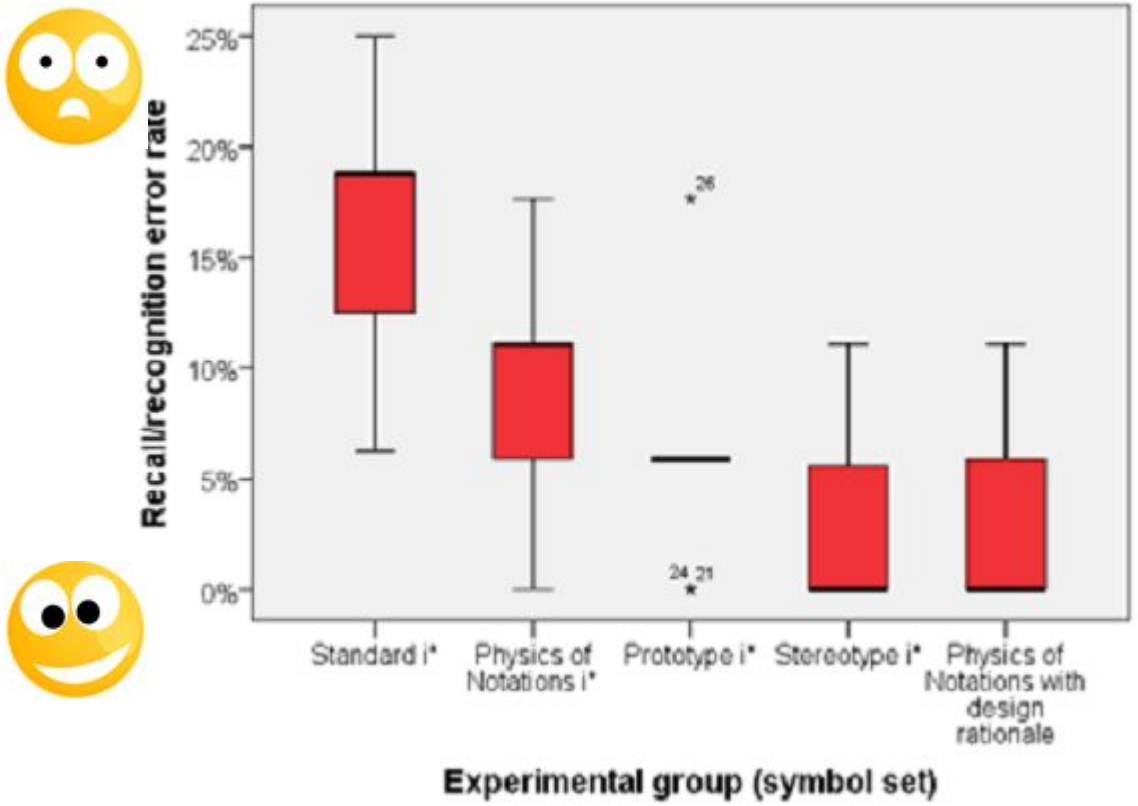
TABLE III. RESULTS OF HYPOTHESIS TESTING FOR SEMANTIC TRANSPARENCY COEFFICIENT (GREEN = CONFIRMED, WHITE = REJECTED, RED = CONVERSE)

Hypothesis	Statistical significance (p)	Practical meaningfulness (d)
H1: PoN > Standard	.005**	1.22+++
H2: Stereotype> Standard	.000***	3.32+++
H3: Prototype > Standard	.002***	2.19+++
H4: Stereotype > PoN	.000***	1.57+++
H5: Prototype > PoN	.703	–
H6: Prototype > Stereotype	.001***	–2.21+++

Statistical significance: * significant with $\alpha = .05$, ** $\alpha = .01$, *** $\alpha = .005$
Practical meaningfulness: + small effect ($|d| \geq .2$), ++ medium effect ($|d| \geq .5$), +++ = large effect ($|d| \geq .8$)

- Ste
(Ou ndard)

Which notation contains the most easy to recognize symbols?



Note: Caire et al. added an extra language combining physics of notations with a design rationale for this particular evaluation.

Which syntax is easier to recognize?

TABLE IV. RESULTS OF HYPOTHESIS TESTING FOR RECOGNITION (GREEN = CONFIRMED, WHITE = REJECTED)

Hypothesis	Statistical significance (p)	Practical meaningfulness (d)
H13: PoN > Standard	.006**	1.16 +++
H14: Stereotype> Standard	.000***	2.32+++
H15: Prototype > Standard	.000***	1.65+++
H16: Stereotype > PoN	.022***	1.02+++
H17: Prototype > PoN	.052	–
H18: Prototype > Stereotype	.708	–
H19: PoN DR > PoN	.041*	1.05+++

What are the main conclusions
of this semantic transparency evaluation?

We can objectively **measure** the semantic transparency of symbols and notations

- Setting up blind interpretation experiments
- Measuring successful recognition of symbols
  - Hit rate
  - Semantic transparency coefficient

We can objectively **improve** the semantic transparency of visual notations

- Conduct symbolization experiments
- Perform stereotyping analysis
- Approach allowed designing new symbols that meet the ISO threshold for public information and safety symbols (67%) – self explanatory symbols

Novices can design more semantically transparent symbols than experts

- 8 out of the 9 best symbols designed by novices
- 8 out of the 9 worst symbols designed by experts

We can actively (and productively) involve end users in the process of designing notations through...

- Symbolisation experiments
- Blind interpretation experiments
- Recognition experiments

We can evaluate user comprehensibility of visual notations prior to their release

- Use domain users (the DSL target audience) in
  - Blind interpretation studies
  - Blind recognition studies
- Approach inspired by clinical trials for drugs or usability testing for software systems
- Routine approach for international safety symbols certification, but not typical with software languages

What effect does semantic transparency have on understanding by novices?

- Semantic transparency significantly increases recognition accuracy and reduces interpretation errors
- 10% increase in semantic transparency leads to a 6.6% reduction in interpretation errors.
  - Consider the cost of requirements errors in software development

What effect does use of explicit design principles have on cognitive effectiveness of visual notations?

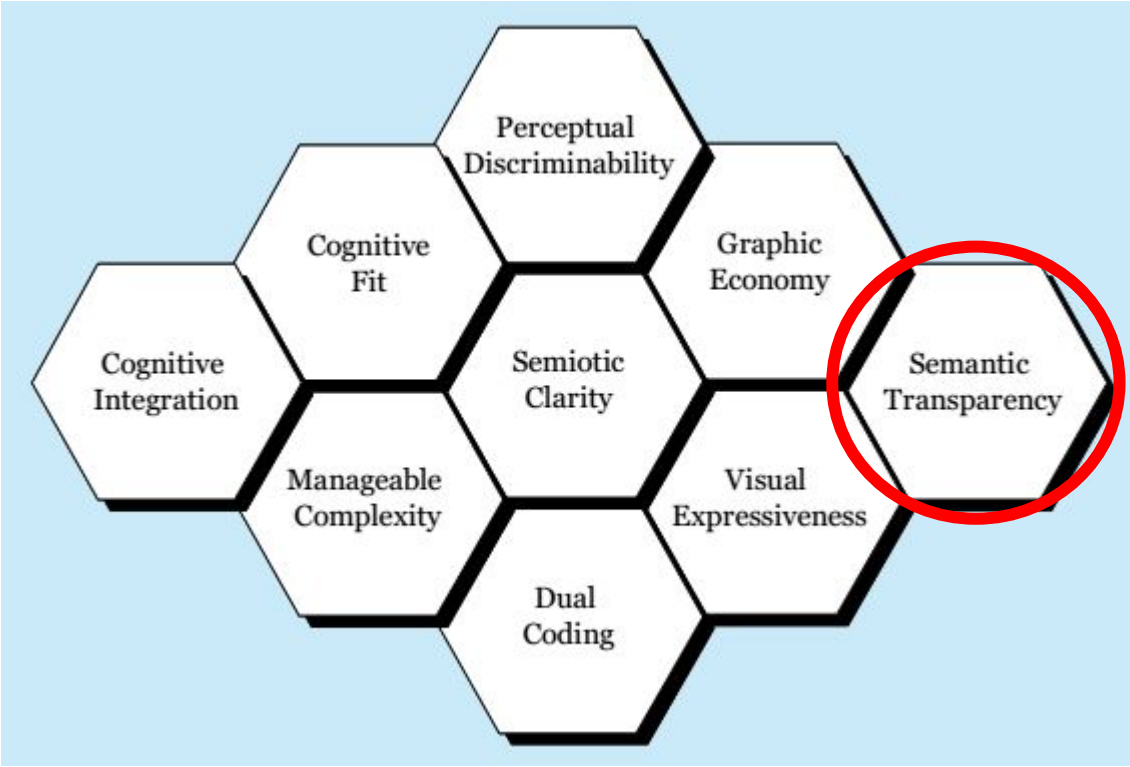
- Use of explicit design principles:
  - Significantly improved semantic transparency
  - Significantly improved cognitive effectiveness
    - To be more precise, recognition accuracy is an early measure of cognitive effectiveness
  - More than doubled the average hit rate for symbols
  - Reduced interpretation errors by almost 50%

Some takeaways

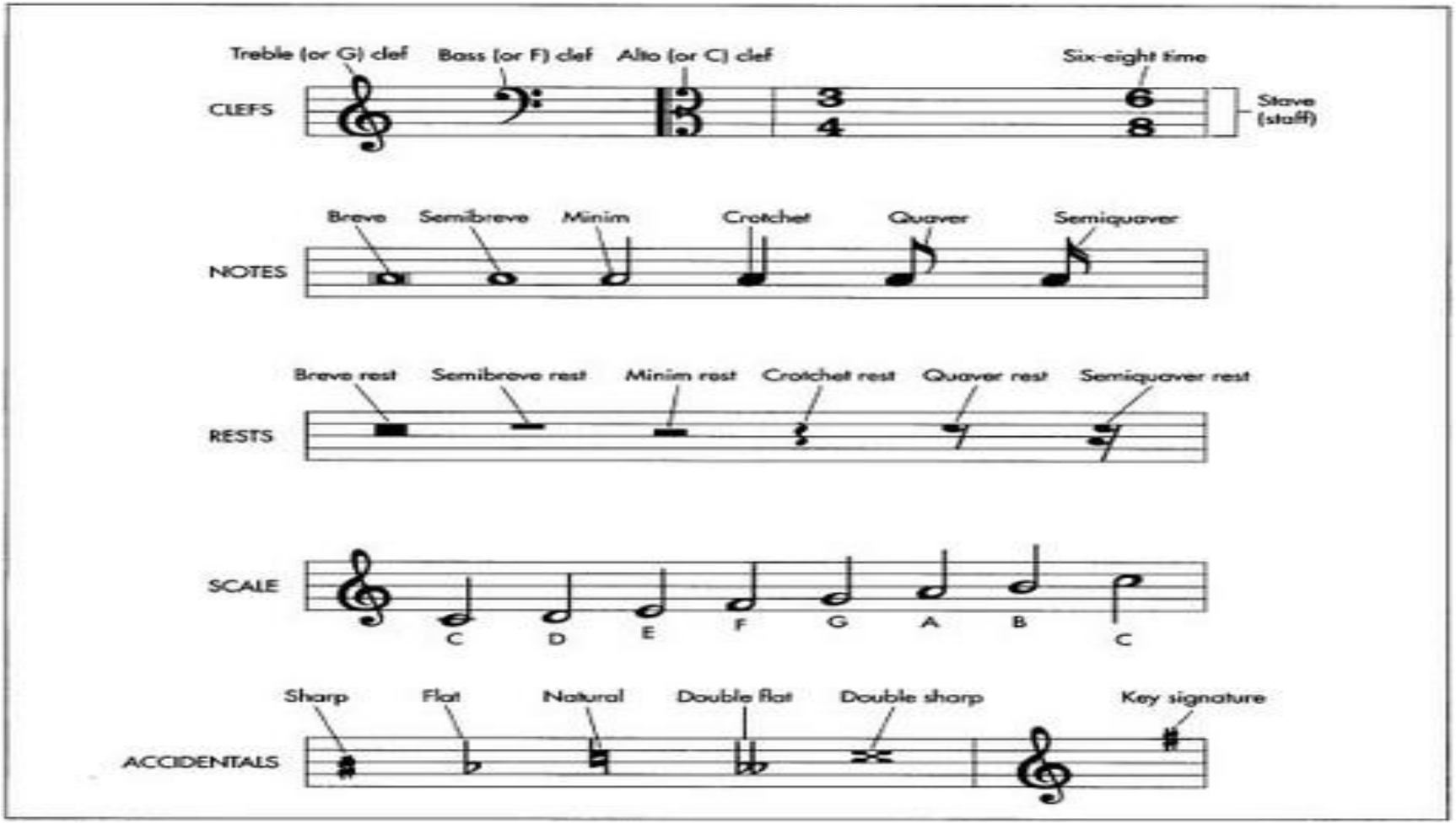
- Over 100 people involved in concrete syntax design
  - Low effort required from each participant (5–20 minutes per task)
  - In contrast, typically, a very low number of experts are involved in concrete syntax design
- Novices, rather than experts, designed (and chose) the most effective symbols
- Design heuristics and rationale significantly improve over traditional language design, but are less effective than “the wisdom of the crowd”, as long as a representative crowd can be assembled

Problem solved?

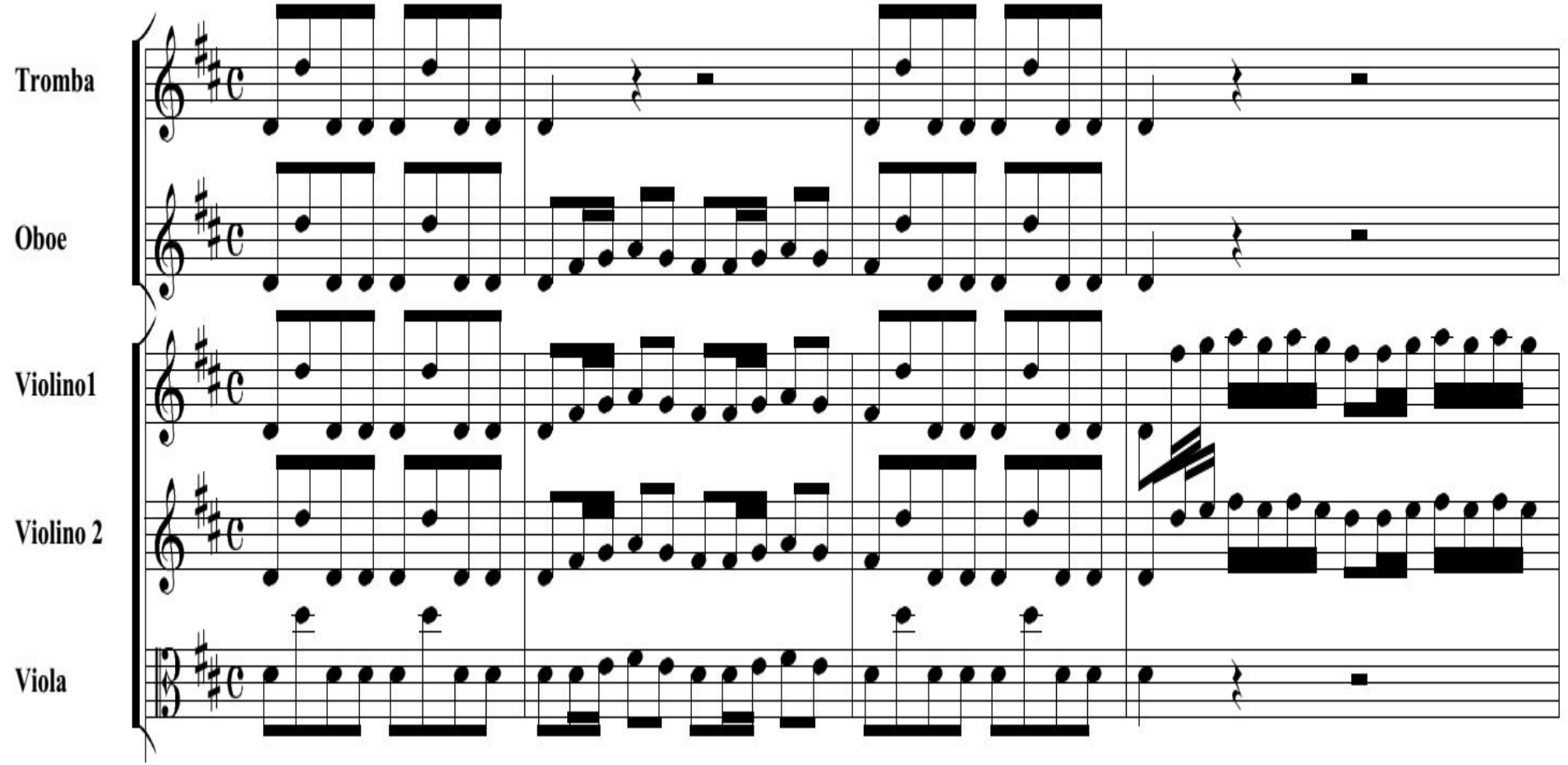
Several physics dimensions to evaluate



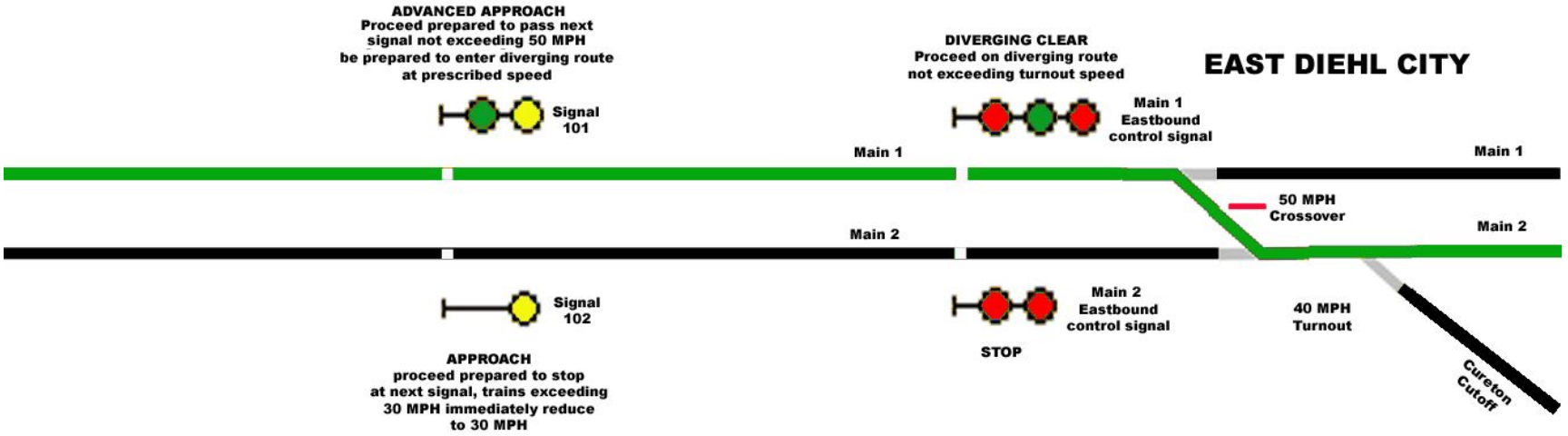
Isolated symbols are but a fraction of a concrete syntax



To read and write with the language, you need to understand how they can be composed



Patterns will typically emerge and should be “easy” to recognize by domain users



Many other challenges to tackle…

- Different “audiences” of the same language are likely to require different dialects
- Reading is much different from writing
- Increased sophistication of language constructs might lead to inconvenient “multi-channel” communication
  - People discussing over a white board
  - People discussing over a coffee table, using common objects as surrogates

We really need to learn,
from real DSL users,
what works for them, and
what does not work for them!

(and it turns out they often lie)

Measuring language users' biosignals

Why bother with human factors?

- Software development is a cognitive intensive activity
- It is likely to be **affected by** (among others)
  - **Cognitive state**
    - Focused, Relaxed, …
  - **Emotional state**
    - Stress…
  - **Personal characteristics**

How is the cognitive effort evaluated in Software Engineering?

- Through proxies
  - Complexity metrics
  - Effort metrics
  - ...
- Ask about it
  - Questionnaires
  - Interviews
  - ...

System usability scale – SUS

- Simple, ten-item Likert scale giving a global view of subjective assessments of usability
- developed by John Brooke at Digital Equipment Corporation in the UK in 1986
- most widely used approach to scaling responses in survey research

System usability scale – SUS

Likert Scale

NASA Task Load Index (NASA-TLX)

developed by the Human Performance Group at NASA's Ames Research Center over a three-year development cycle that included more than 40 laboratory simulations

assessment tool that rates perceived workload in order to assess a task

subjective, multidimensional

six subjective subscales

Mental Demand

How much mental and perceptual activity was required? Was the task easy or demanding, simple or complex?

Physical Demand

How much physical activity was required? Was the task easy or demanding, slack or strenuous?

Temporal Demand

How much time pressure did you feel due to the pace at which the tasks or task elements occurred? Was the pace slow or rapid?

Overall Performance

How successful were you in performing the task? How satisfied were you with your performance?

Effort

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration Level

How irritated, stressed, and annoyed versus content, relaxed, and complacent did you feel during the task?

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

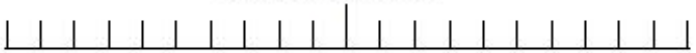
Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

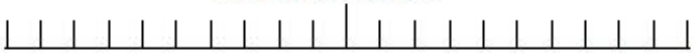


Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?



Very Low — Very High

Problem with asking developers about the perceived complexity of a task

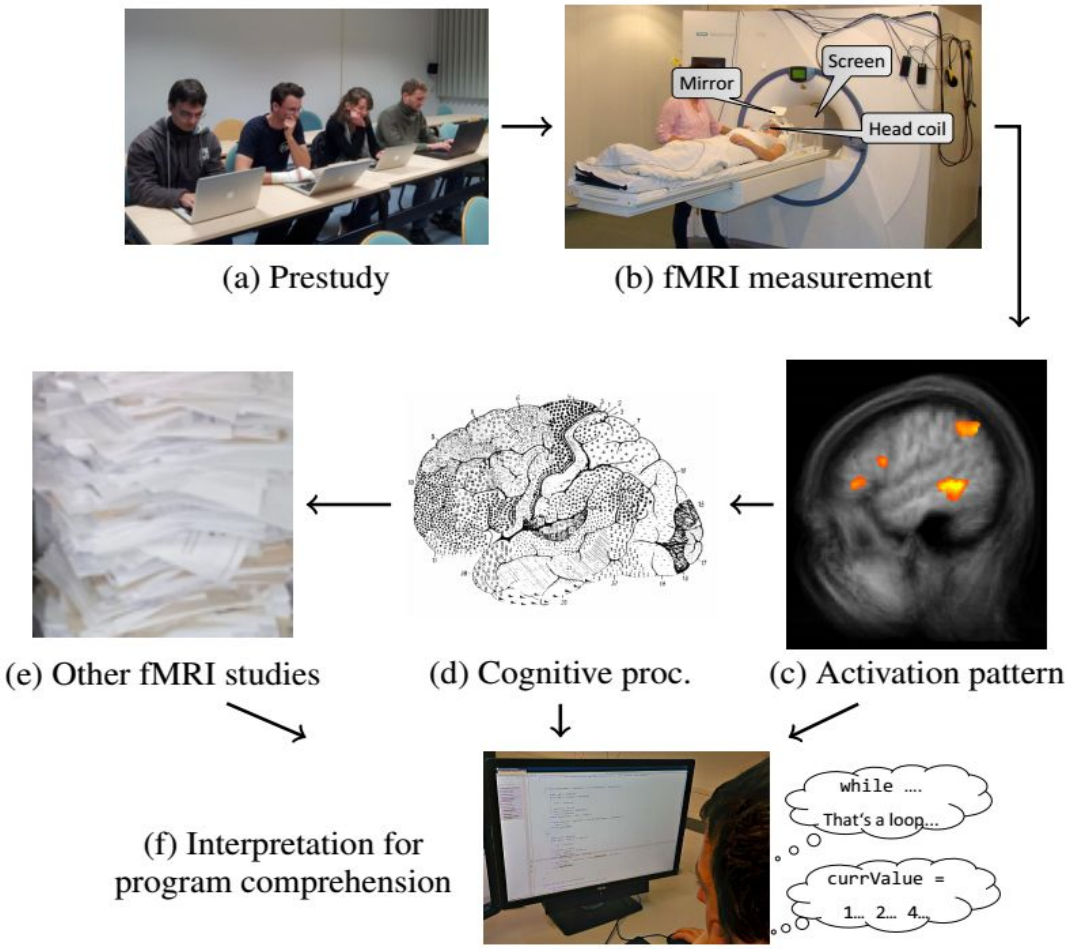
- Inherently subjective answer
  - Depends on the evaluators background
    - Complexity perception likely to be strongly correlated with expertise
  - Depends on non-technical factors such as personality
  - Often biased by the Hawthorne effect
    - Process where human subjects of an experiment change their behavior, simply because they are aware of being studied

Proposed approach to mitigate these problems

- Rather than simply asking the developer about his experience, monitor them while performing their tasks
  - Borrow knowledge from other domains, namely
    - Psychology
    - Human-Computer Interaction
    - ...

How do humans understand software artifacts?

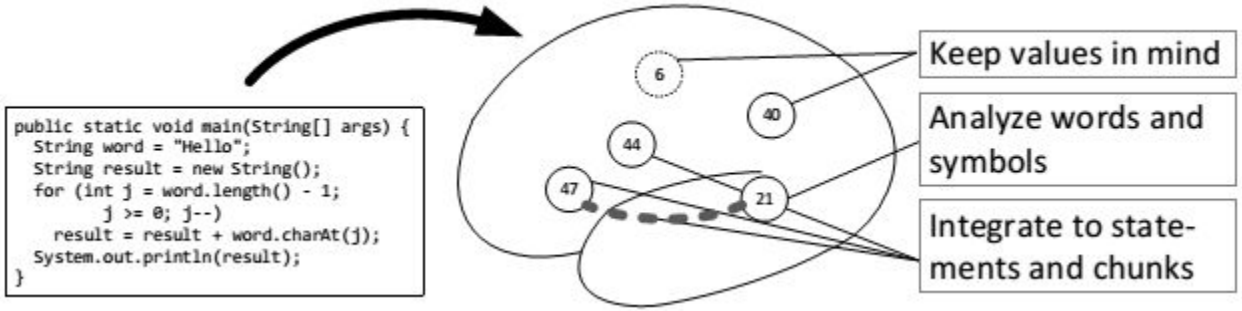
Siegmund *et al.* "Understanding Understanding Source Code with Functional Magnetic Resonance Imaging", ICSE 2014



How are we using our brains?

- Early findings:
  - 5 brain regions activated, related to
    - Working memory
    - Attention
    - Language comprehension

    → Program comprehension
  - Left-hemisphere lateralization (analytical skills)



On the way humans read software artifacts

NOVALINCS
FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Typical tasks in software development

- Language usage tasks:
  - writing
  - reading
  - interpretation
  - comprehension
  - memorization
  - problem solving

What does this code do?

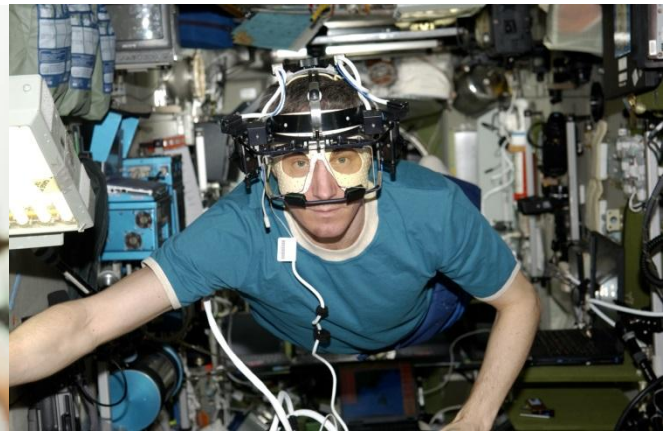
```
1    n = 3
2    while (n > 1):
3        print n
4        n = n - 1
```

How did you read the code?

Why should we care about where the developer is looking at?

- Visual attention triggers mental processes to comprehend and solve a given task
- This makes visual attention a candidate proxy for cognitive effort; one may track, for instance:
  - Location of gaze
  - Duration of gaze
  - ...

How can we track eye movement?

Too intrusive? There are more convenient options…

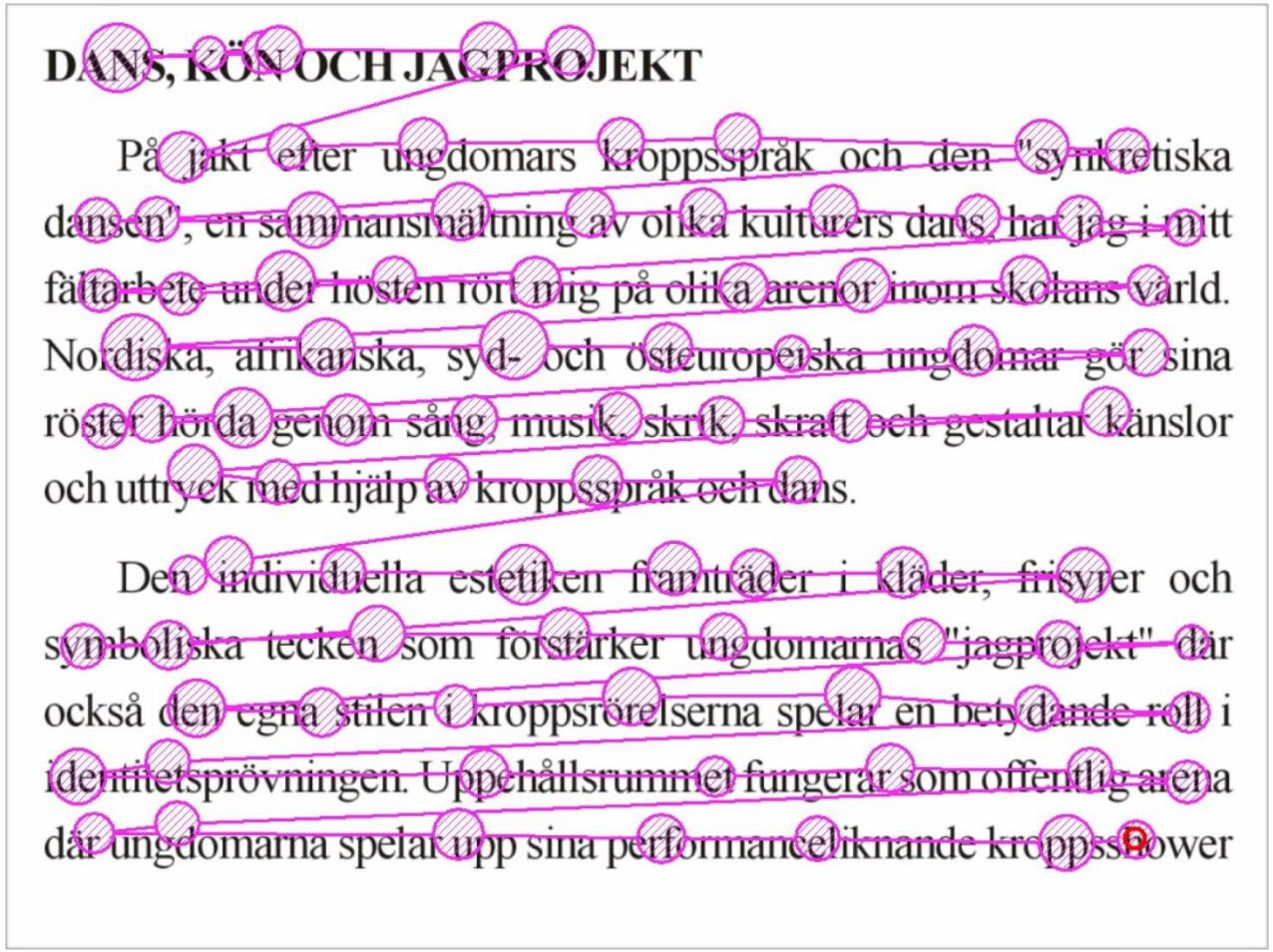
Main types of eye gaze data

- Fixation
  - The gaze fixes in a given position

Main types of eye gaze data

- Saccade
  - Quick movement from one fixation point to the next

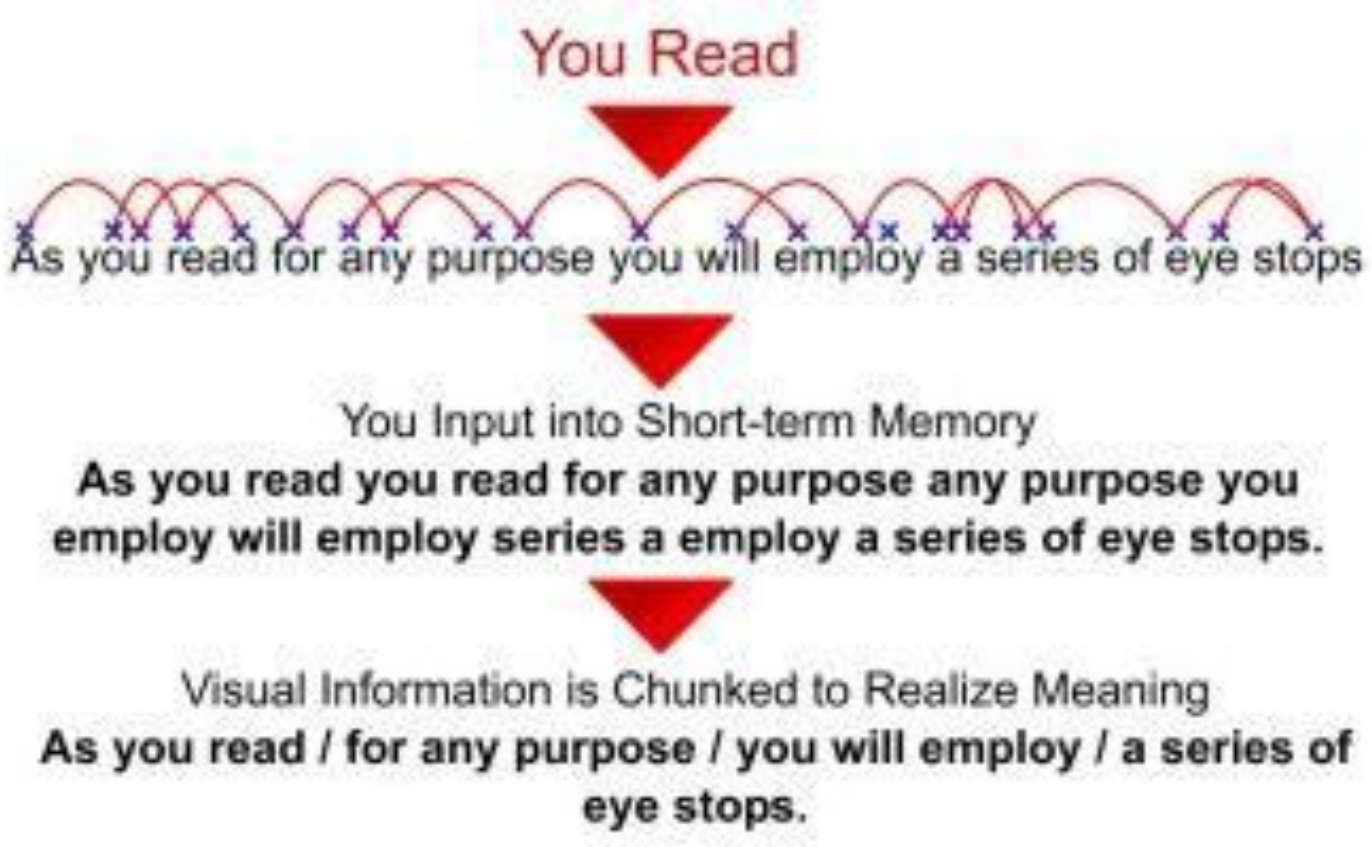
Reading text in natural language



NOVALINCS

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Making sense of the text



How is *natural language reading* different from *textual DSL reading*?

	Natural language	Textual DSL
Syntax	Larger vocabulary Complex grammar rules Freedom to bend rules	Limited vocabulary Simpler grammar rules **Stricter rules**
Semantics	Text comprehension Domain comprehension	Text comprehension Domain comprehension **Execution comprehension**

Reading from a textual DSL

- Humans tend to read natural language text linearly
- Novice developers read the textual DSL less linearly than natural language text
- Expert developers read the textual DSL less linearly than novices

- Non-linear reading skills increase with expertise

Reading “modes”

- Natural language
  - Proficient vs novice readers
    - Proficient readers often skip words
    - Novice readers read/re-read more frequently
- Textual DSL
  - Novice developers tend to read the textual DSL like they would read a natural language text

Assessing reading with eye trackers

- Vertical Next Text
  - % of forward saccades that either stay on the same line or move one line down.
- Vertical Later Text Definition
  - % of forward saccades that either stay on the same line or move down any number of lines.

Assessing reading with eye trackers

- Horizontal Later Text
  - % of forward saccades within a line
- Regression Rate
  - % of backward saccades of any length
- Line regression rate
  - % of backward saccades within a line.

Eye tracking measures

- Saccade length
  - Average Euclidean distance between the participant's two consecutive fixations in each trial
  - Hint: experts tend to exhibit higher saccade length than novices

Eye tracking measures

- Element coverage
  - Percentage of words / source code elements looked at by the subject
  - Hint: experts look at fewer but more relevant words in a textual DSL than novices

Eye tracking measures (for textual source code)

- Story order
  - Program read line by line, from left to right
  - N-W alignment score of fixation order with linear text reading order
- Execution order
  - Program read following its control flow
  - N-W alignment score of fixation order with the program's control flow order

Story vs execution order (for textual source code)

```
1 n = 3
2 while (n > 1):
3     print n
4     n = n - 1
```

Story order: 1, 2, 3, 4

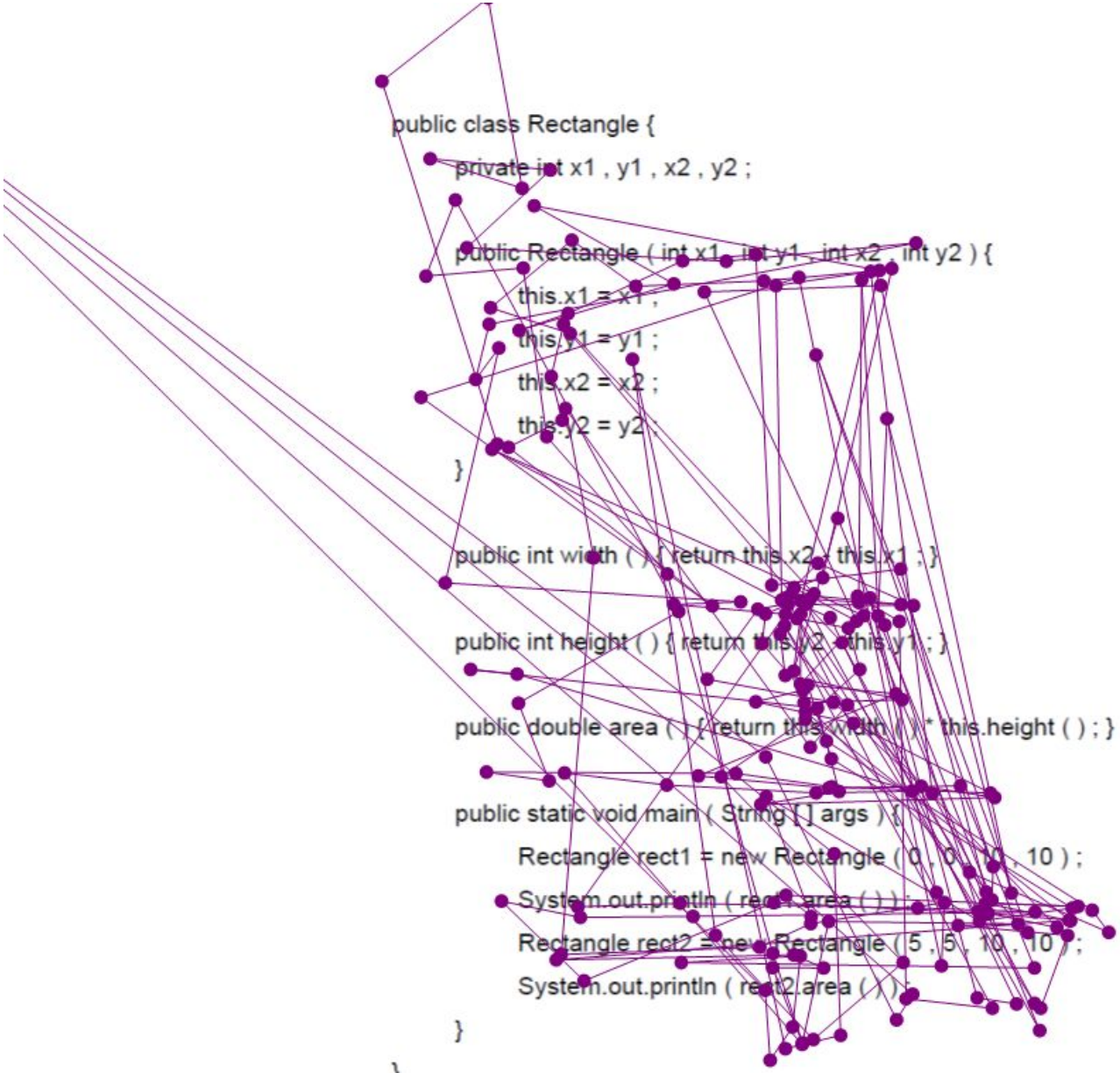
Execution order: 1, 2, 3, 4, 2, 3, 4, 2

Example:

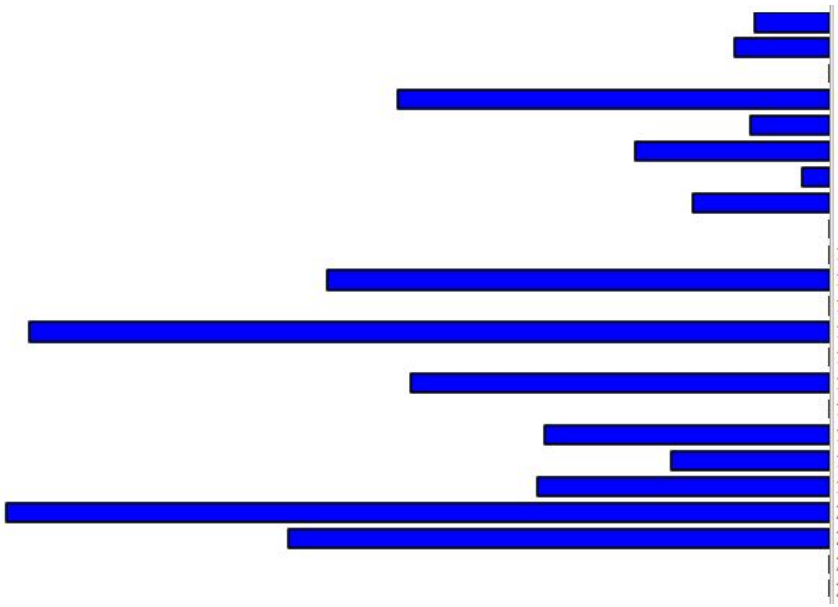
- Please study the code on the next slide
- You will be asked about the return value of

rect2.area()

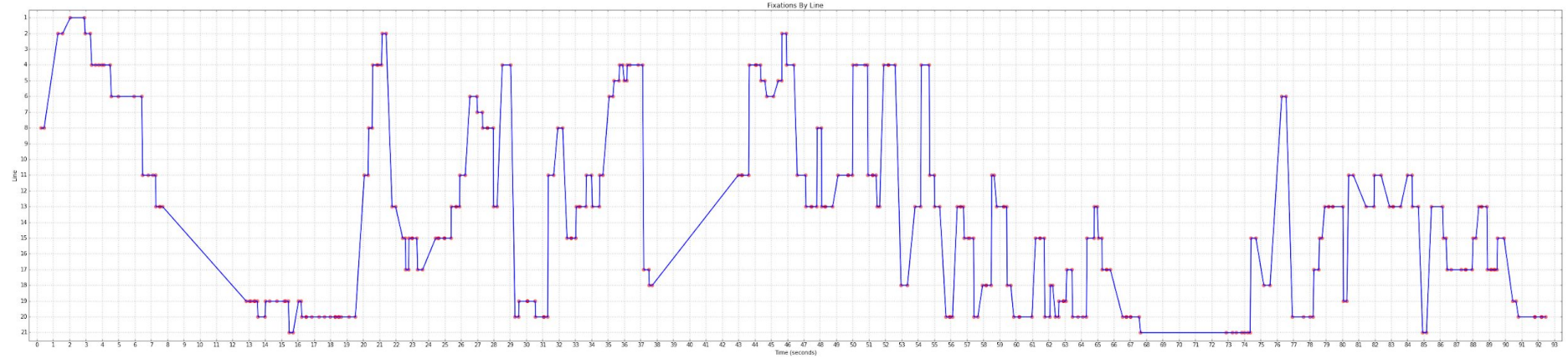
```
public class Rectangle {
        private int x1 , y1 , x2 , y2 ;

        public Rectangle ( int x1 , int y1 , int x2 , int y2 ) {
                this.x1 = x1 ;
                this.y1 = y1 ;
                this.x2 = x2 ;
                this.y2 = y2 ;
        }

        public int width ( ) { return this.x2 - this.x1 ; }

        public int height ( ) { return this.y2 - this.y1 ; }

        public double area ( ) { return this.width ( ) * this.height ( ) ; }

        public static void main ( String [ ] args ) {
                Rectangle rect1 = new Rectangle ( 0 , 0 , 10 , 10 ) ;
                System.out.println ( rect1.area ( ) ) ;
                Rectangle rect2 = new Rectangle ( 5 , 5 , 10 , 10 ) ;
                System.out.println ( rect2.area ( ) ) ;
        }
}
```


public class Rectangle {
private int x1 , y1 , x2 , y2 ;
public Rectangle (int x1 , int y1 , int x2 , int y2) {
this.x1 = x1 ;
this.y1 = y1 ;
this.x2 = x2 ;
this.y2 = y2 ;
}
public int width ( ) { return this.x2 - this.x1 ; }
public int height ( ) { return this.y2 - this.y1 ; }
public double area ( ) { return this.width ( ) * this.height ( ) ; }
public static void main (String [ ] args) {
Rectangle rect1 = new Rectangle (0 , 0 , 10 , 10) ;
System.out.println (rect1.area ( )) ;
Rectangle rect2 = new Rectangle (5 , 5 , 10 , 10) ;
System.out.println (rect2.area ( )) ;
}
}

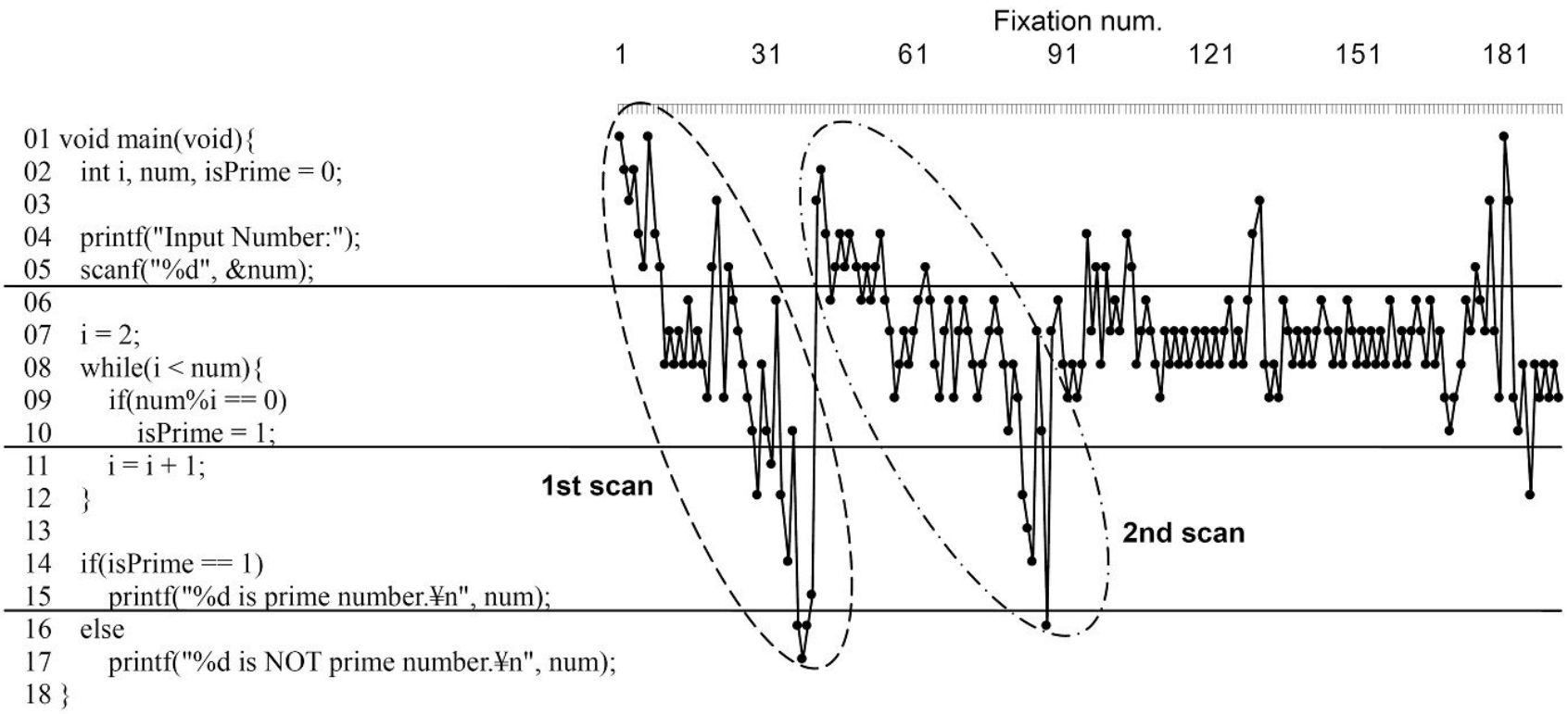
Fixations per line



```
1  public class Rectangle {
2          private int x1 , y1 , x2 , y2 ;
3
4          public Rectangle ( int x1 , int y1 , int x2 , int y2 ) {
5                  this.x1 = x1 ;
6                  this.y1 = y1 ;
7                  this.x2 = x2 ;
8                  this.y2 = y2 ;
9          }
10
11         public int width () { return this.x2 - this.x1 ; }
12
13         public int height () { return this.y2 - this.y1 ; }
14
15         public int double area () { return this.width () * this.height () ; }
16
17         public static void main ( String [] args ) {
18                 Rectangle rect1 = new Rectangle ( 0 , 0 , 10 , 10 ) ;
19                 System.out.println ( rect1.area () ) ;
20                 Rectangle rect2 = new Rectangle ( 5 , 5 , 10 , 10 ) ;
21                 System.out.println ( rect2.area () ) ;
22         }
23 }
```

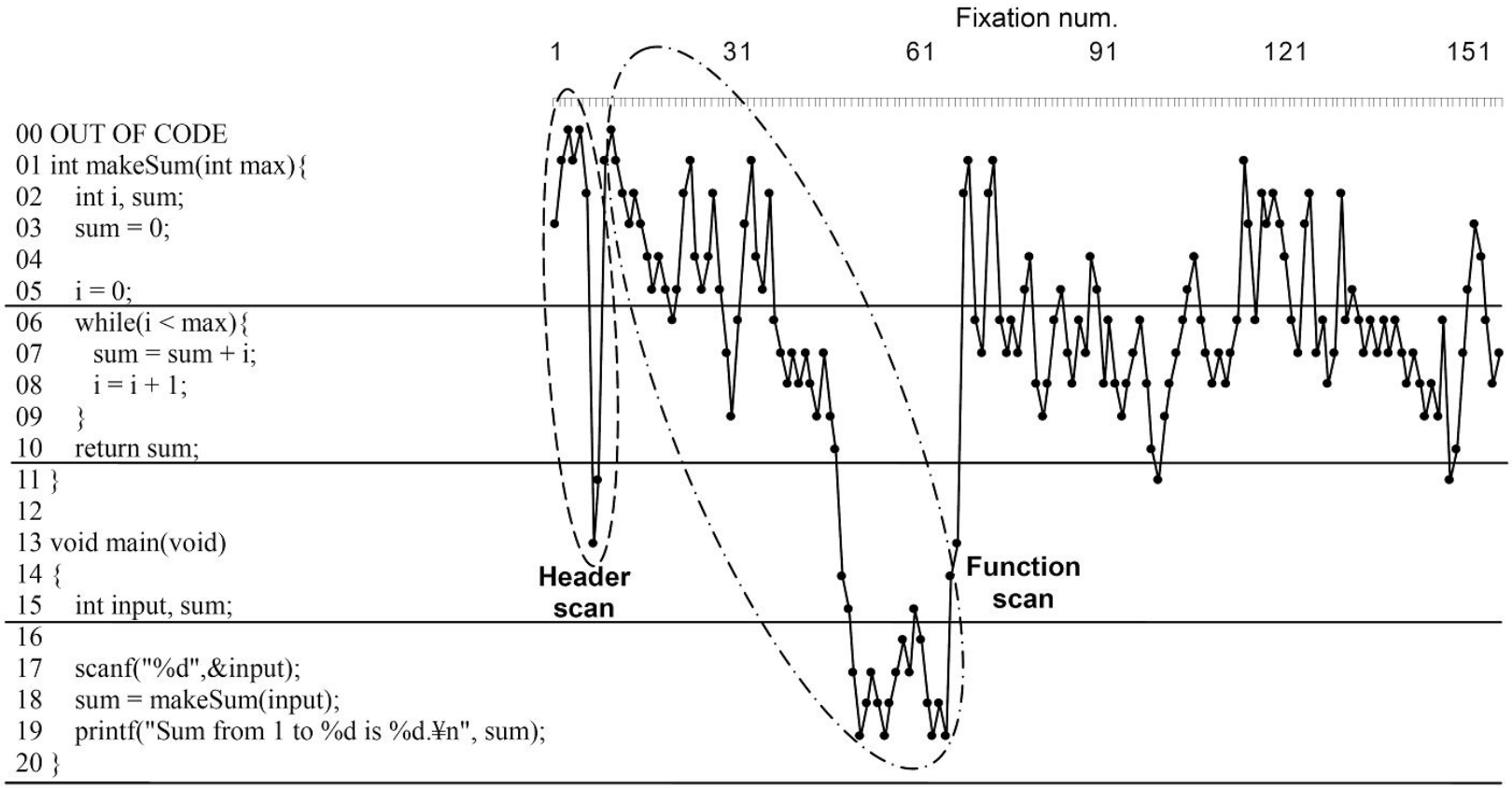


How do we read code?



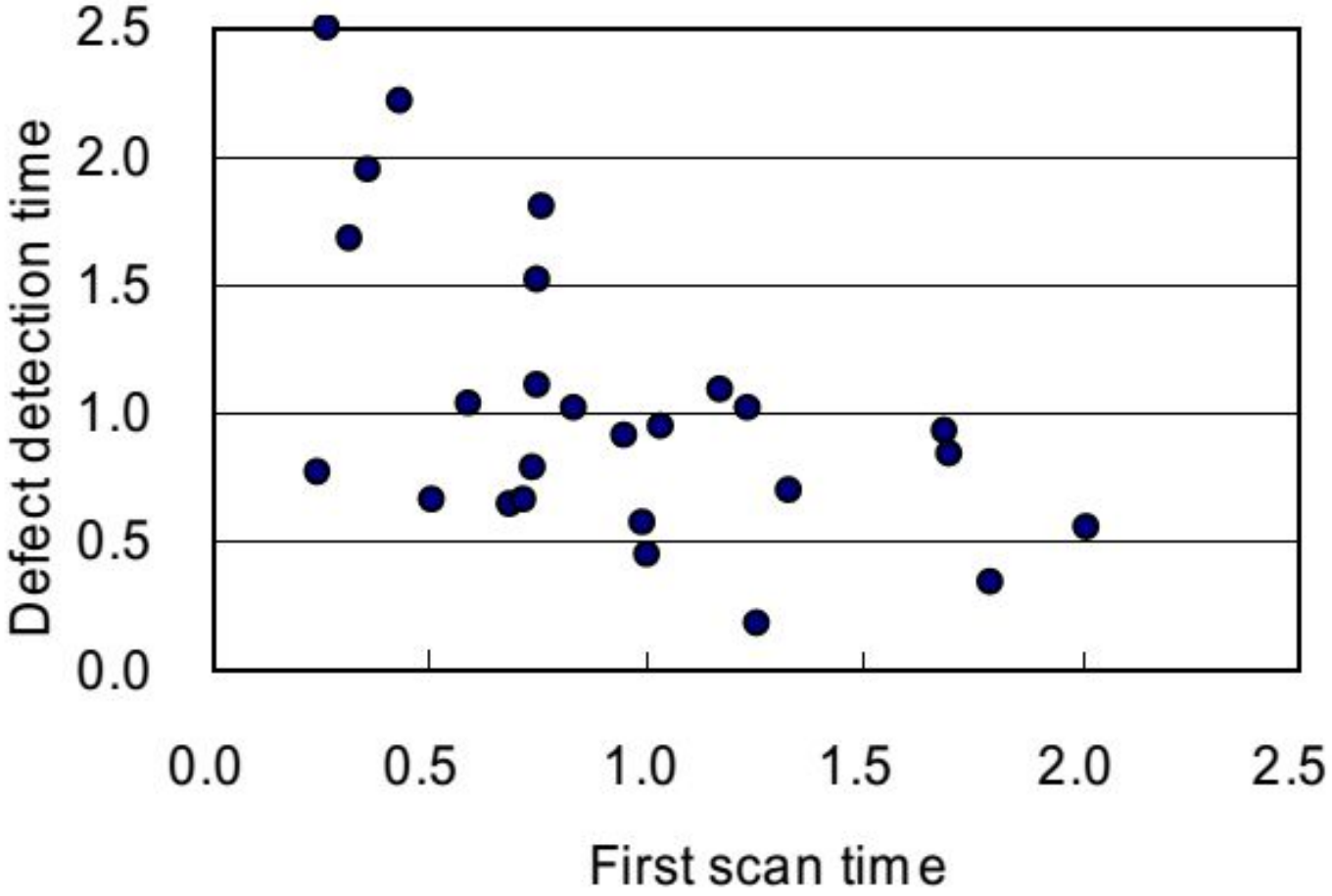
a) Subject E reviewing `Prime`

How do we read code?

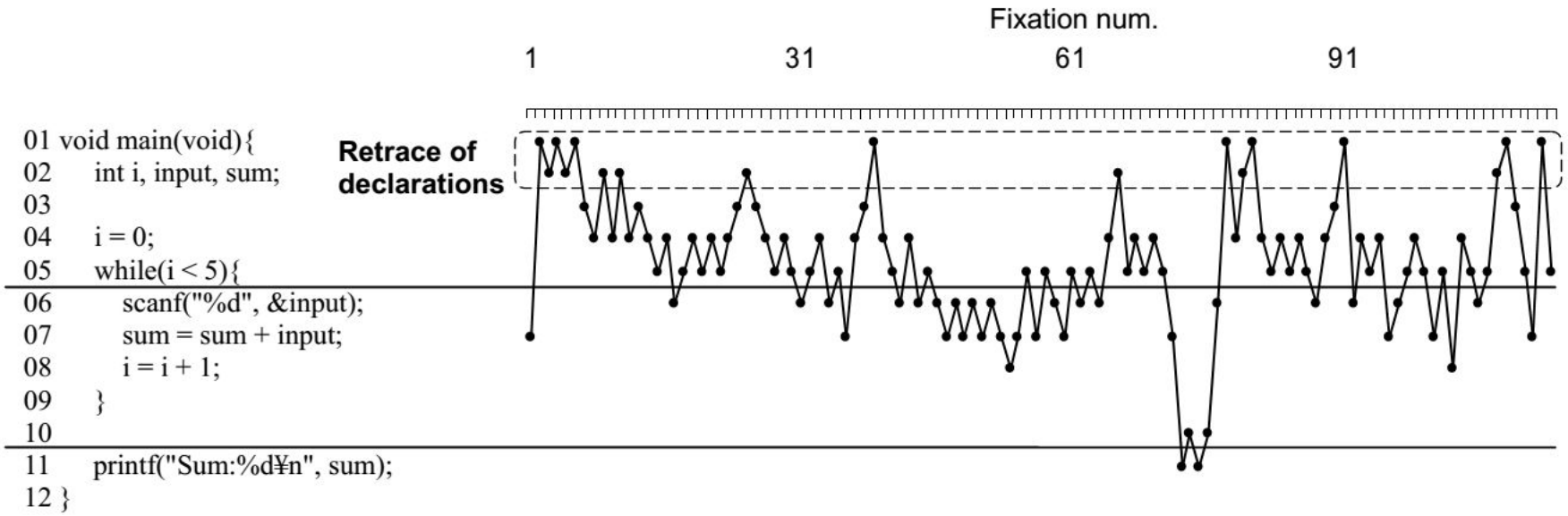


b) Subject C reviewing `Accumulate`

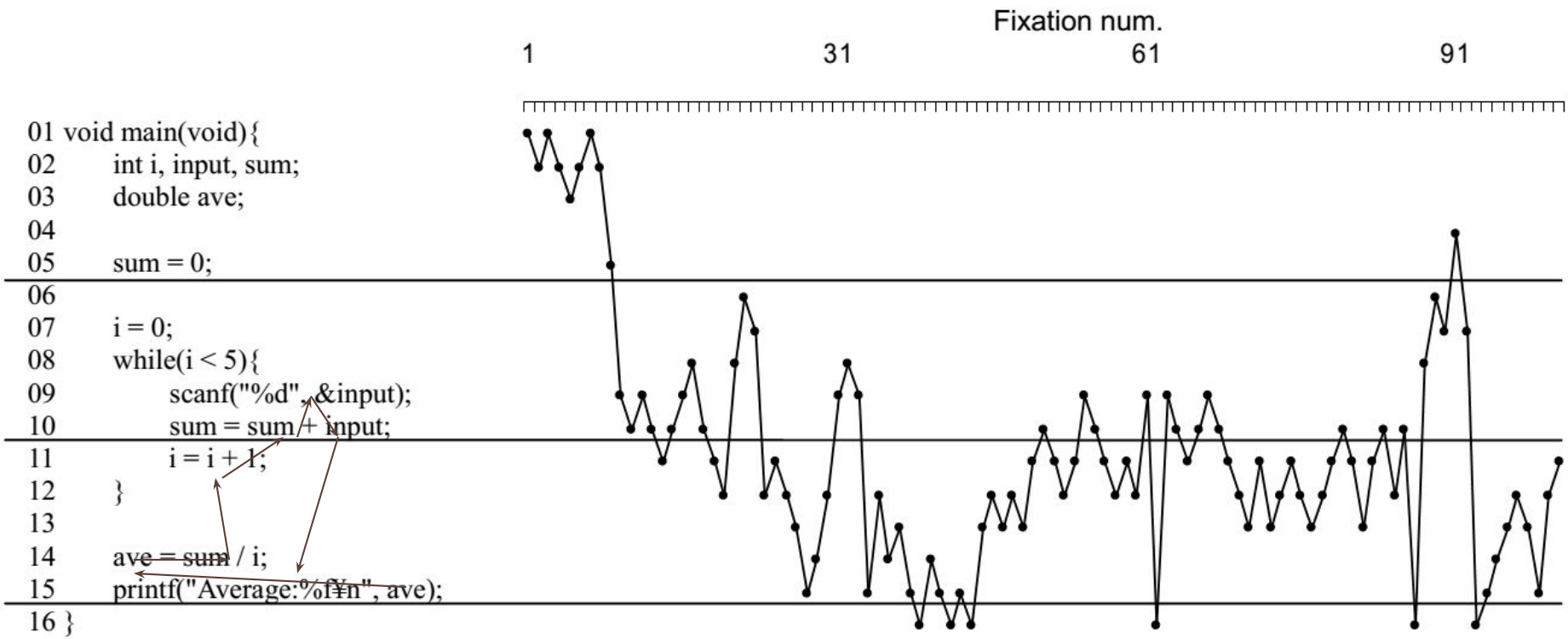
First scan correlates with time to detect defects



Retrace declaration pattern



Retrace reference pattern



Some takeaways

- When confronting practitioners with the trace of their eye movement, they were able to reconstruct their thought process with much more detail

What is the Impact of Bad Layout in the Understandability of Social Goal Models?

Mafalda Santos, Catarina Gralha, Miguel Goulão, João Araújo, Ana Moreira and João Cambeiro

NOVALINCS
FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

The *i** language community has a set of recommended layout guidelines

- The guidelines are supposed to help building better *i** specifications

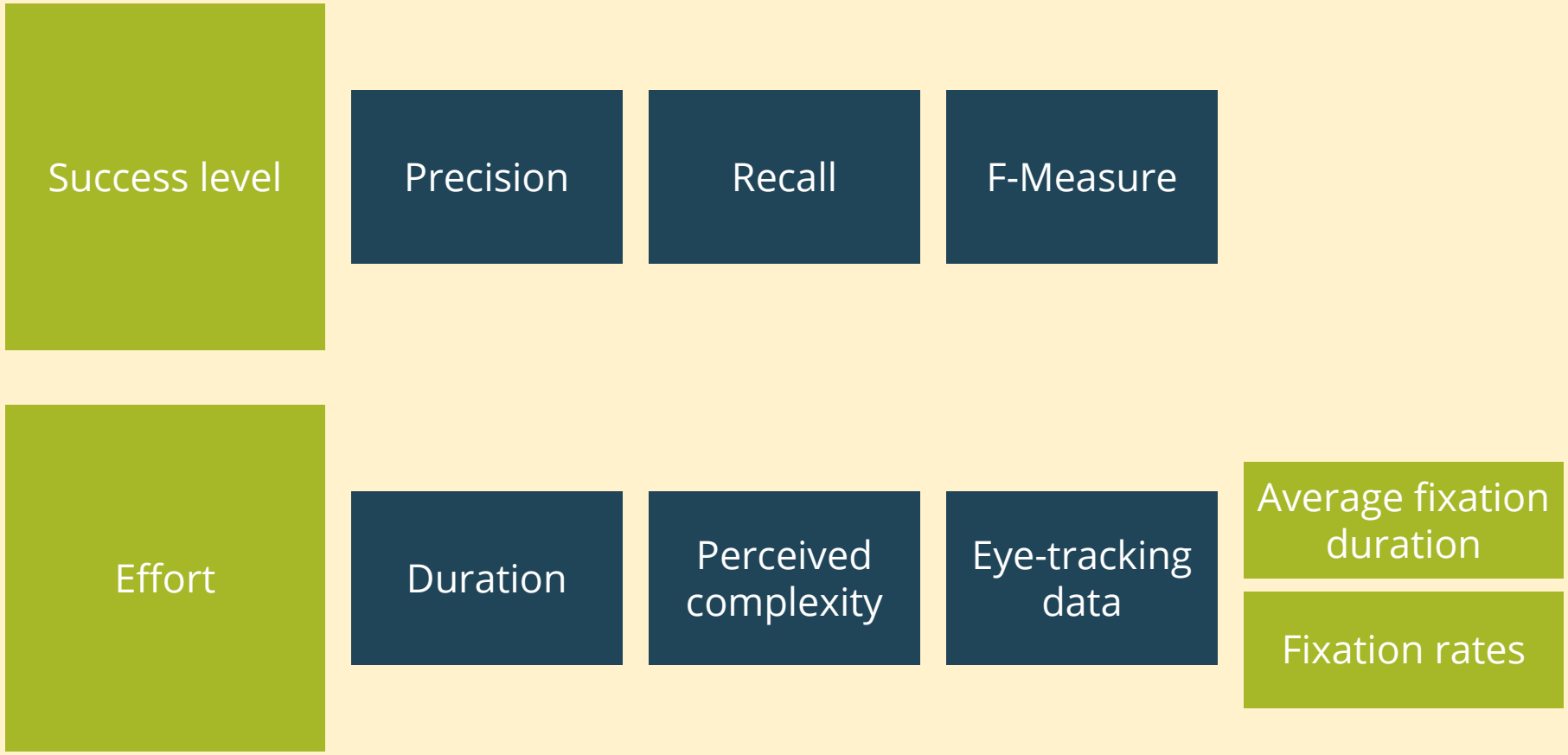
#	Guideline	Enforced	Layout
1	Avoid or minimise drawing intersecting Links and overlapping Links with other Links and elements' text	No	Yes
2	Make both sides of a Dependency Link look like a single, continuous curve as it passes through the Dependum	Yes	Yes
3	Spread the connection points of Dependency Links out on an Actor	Yes	Yes
4	Keep elements horizontal. Do not tilt or twist them	Yes	Yes
5	Avoid or minimise overlapping boundaries of Actors where possible	No	Yes
6	Keep Dependency Links outside the boundaries of Actors to improve the readability of the models	No	Yes
7	Use the conventional Actors' boundaries (circles) unless other shapes such as rectangles can improve models' layout	Yes	Yes
8	Avoid overlapping elements inside or outside Actors	No	Yes
9	Connect each Dependency Link in an SR model to the correct element within the actor	No	No
10	Adopt or follow a consistent direction for the goal refinement/ decomposition hierarchy as much as possible	No	Yes
11	Do not draw SR model elements outside the boundaries of the corresponding actors	No	No
12	Unconnected elements within an Actor is indicative of an incomplete model	No	No
13	Don't extend the text of the name of the element beyond the element's border	Yes	Yes
14	Split a large and complex model into consistent pieces to facilitate easier presentation and rendering	No	Yes
15	Don't extract or zoom into a section of an Actor in a model without showing the incoming and outgoing dependencies with other actors or parts of the model	No	Yes
16	Use the specialised actors notation to the degree that you can gain advantage in instantiating the actual stakeholders	Yes	No
17	Use the leaf-level tasks as the system requirements, not the high level Goals and Softgoals	No	No

Research questions

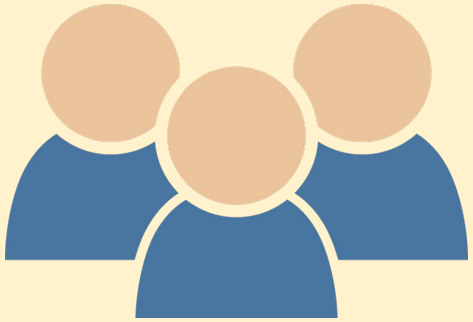
Does adherence to **layout guidelines** influence the **understandability** of i* models?

Does adherence to **layout guidelines** influence the **ability to review** i* models?

Study method: quasi-experiment with a combination of measures

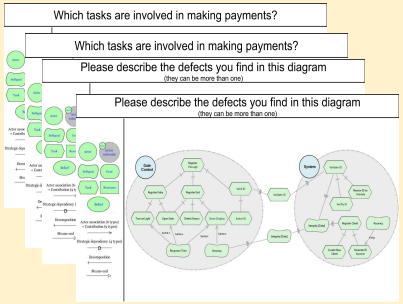


Experimental materials and participants



18 participants

1 eye-tracker



4 domains

Protocol of the experiment

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

Male

Female

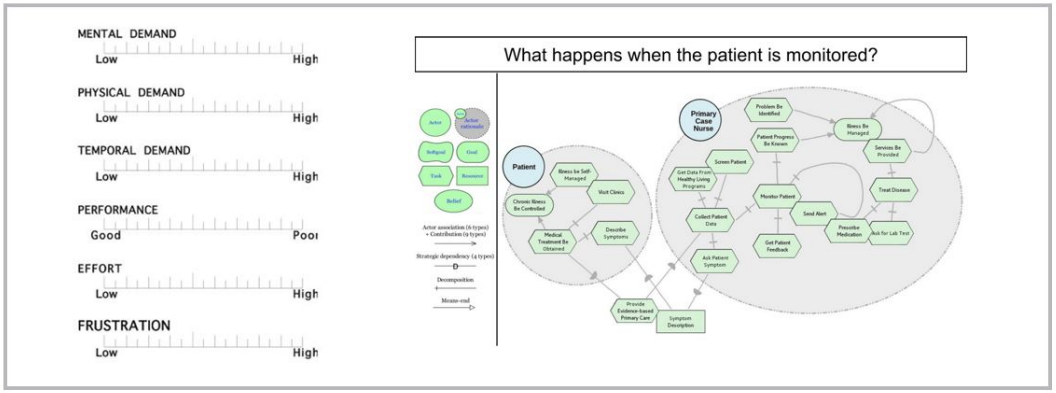
Age *

A sua resposta

Nationality *

A sua resposta

Field of Studies *



Read the consent information letter

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

- Male
- Female

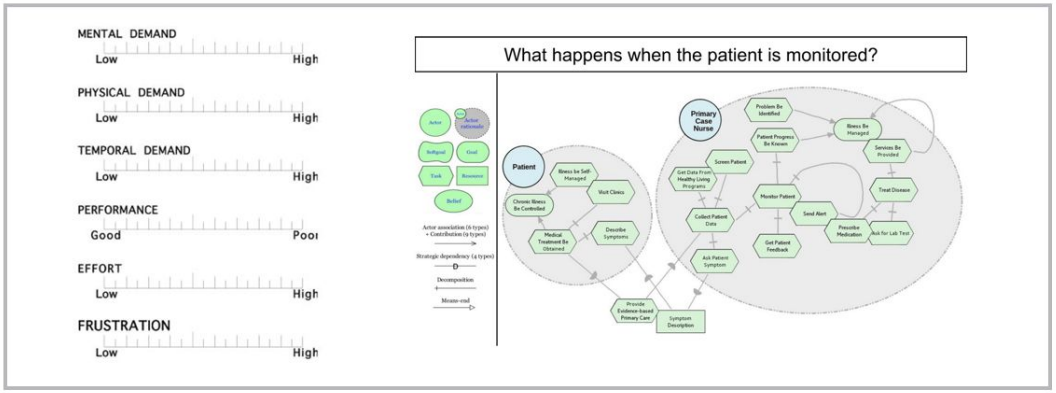
Age *

A sua resposta

Nationality *

A sua resposta

Field of Studies *



Watch a video tutorial about *i**

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

Male

Female

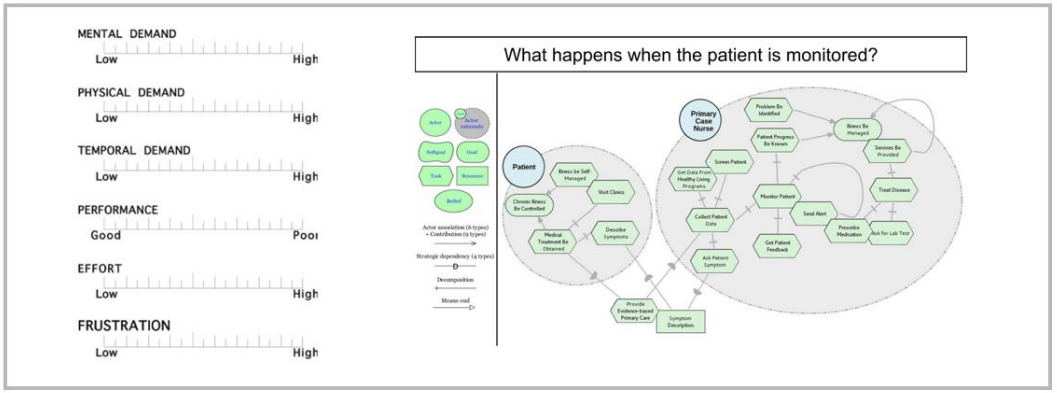
Age *

A sua resposta

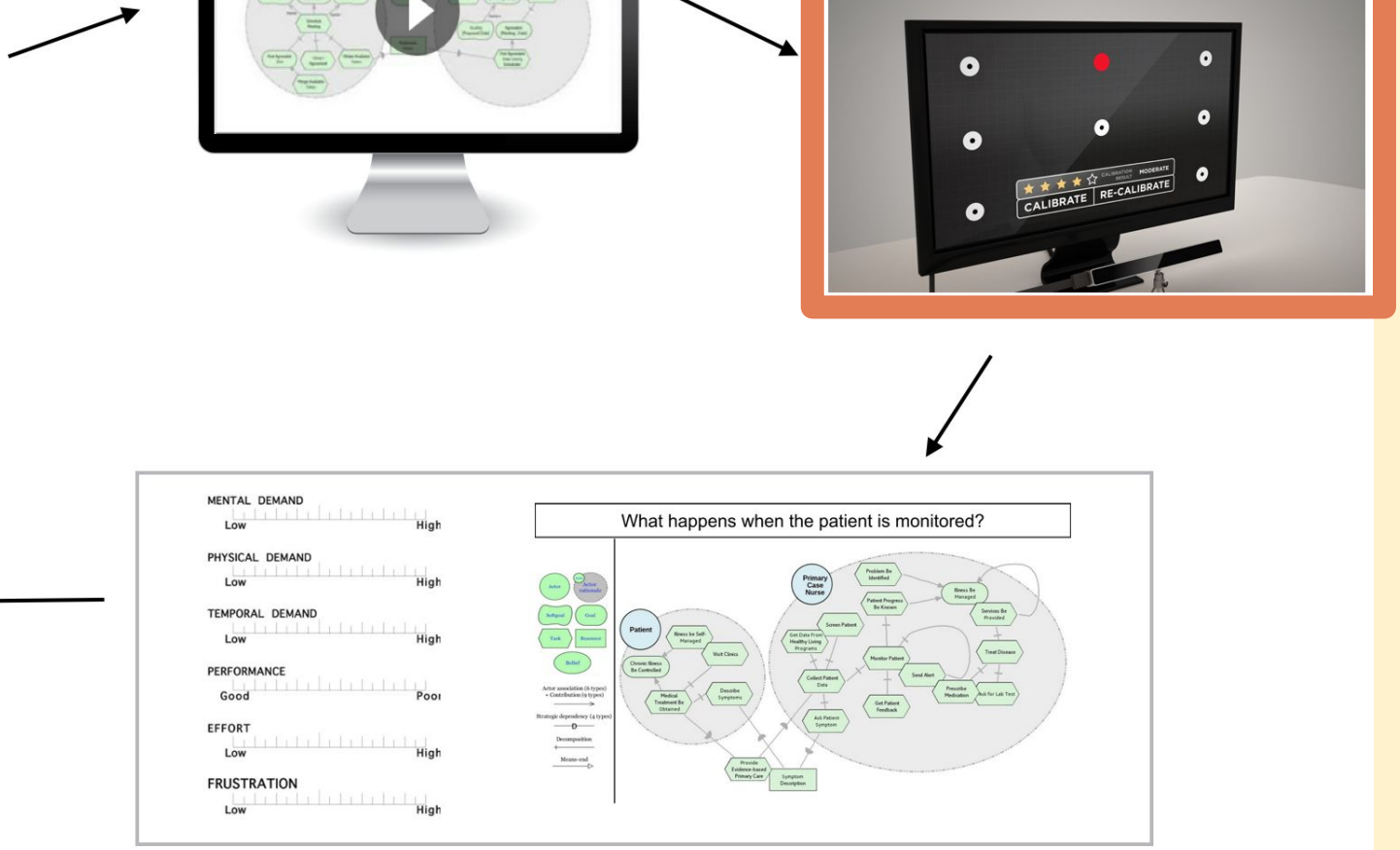
Nationality *

A sua resposta

Field of Studies *



Calibrate the eye-tracker



Answer a question about a model

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

- Male
- Female

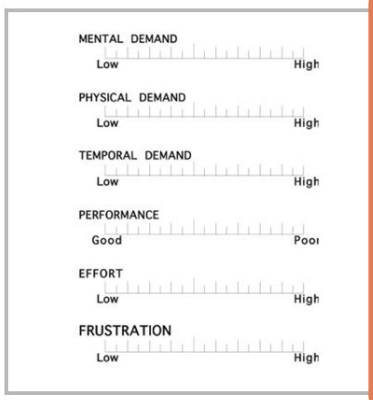
Age *

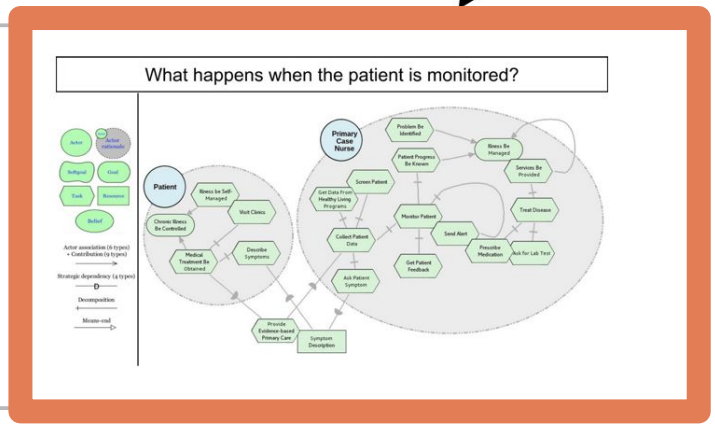
A sua resposta

Nationality *

A sua resposta

Field of Studies *





Answer a NASA TLX questionnaire

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

Male

Female

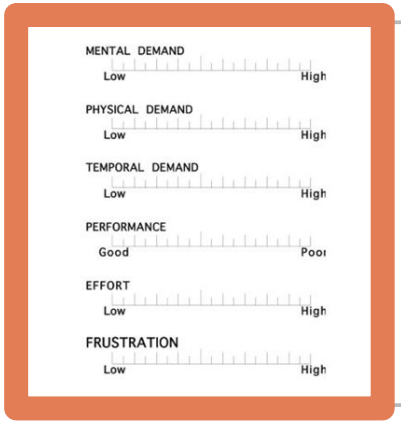
Age *

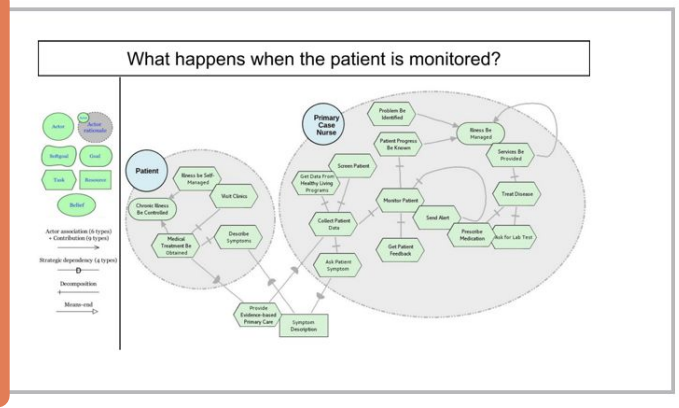
A sua resposta

Nationality *

A sua resposta

Field of Studies *





Answer to demographic questions

Consent information letter

Information to participants

This experimental work is conducted withi
Informatics (NOVA LINCS). NOVA LINCS is
network in the area of Computer Science
hosted at the Departamento de Informá
Universidade NOVA de Lisboa (DI-NOVA),

All information stated as part of this exper

Prof. Miguel Goulão is responsible for
mgoul@fct.unl.pt; +351 21 294 85 36 (ext.

We would like to emphasize that:

- your participation is entirely volunt
- you are free to refuse to answer any

Demographic Data

*Obrigatório

Gender *

Male

Female

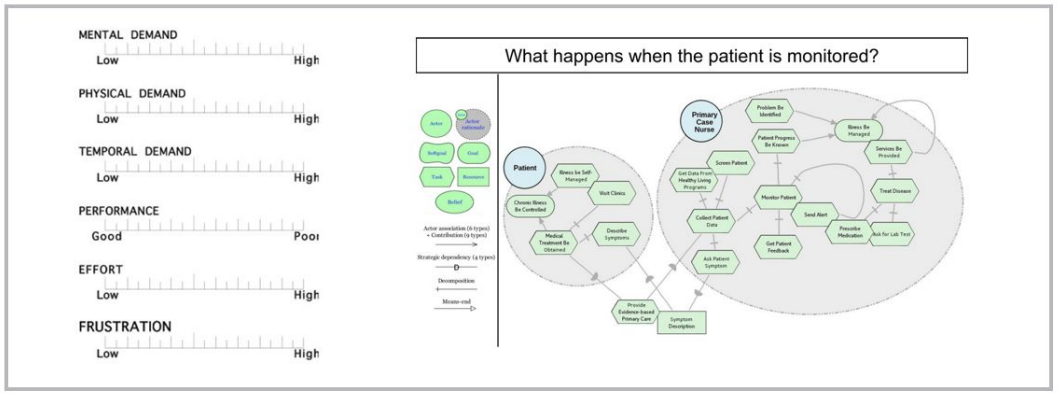
Age *

A sua resposta

Nationality *

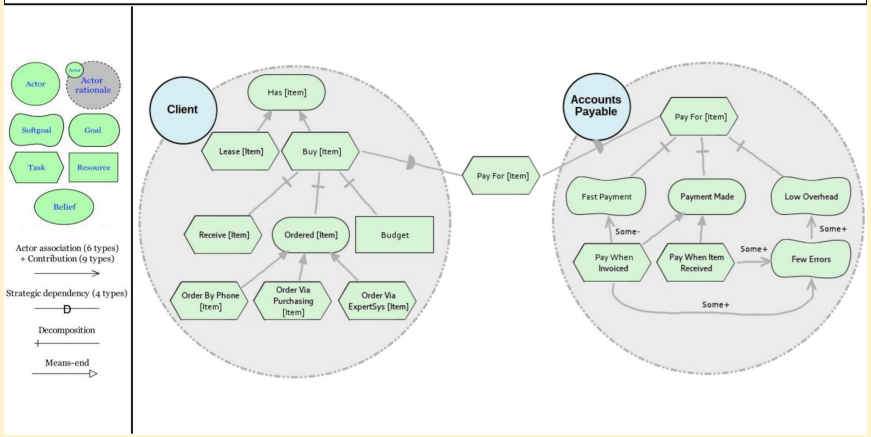
A sua resposta

Field of Studies *

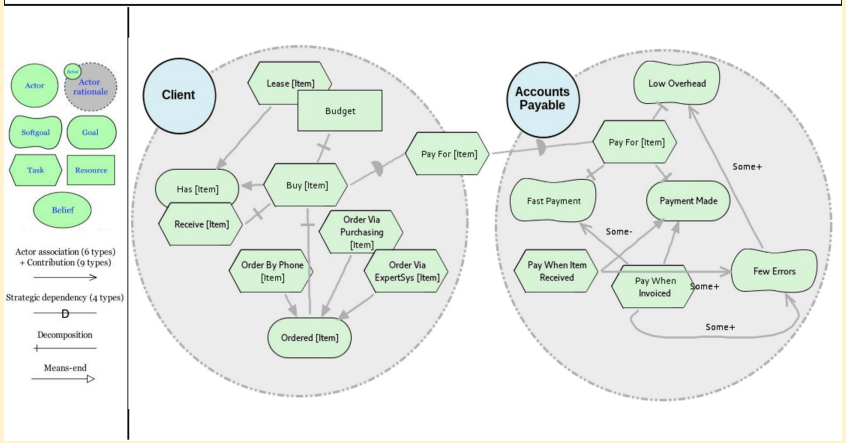


Two understanding and two review tasks, both with good and bad layout

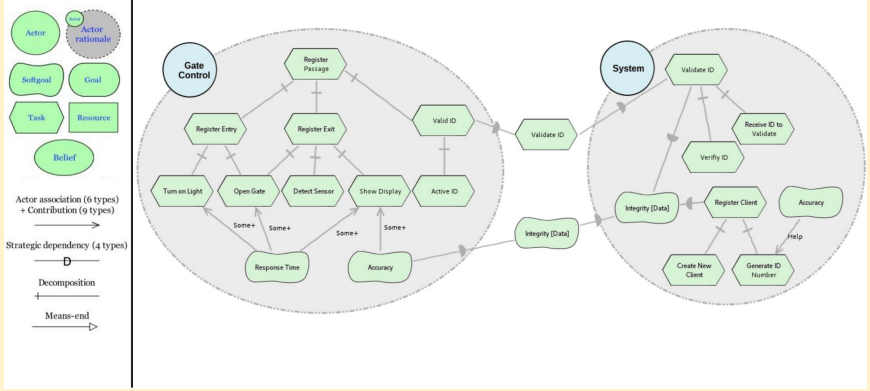
Which tasks are involved in making payments?



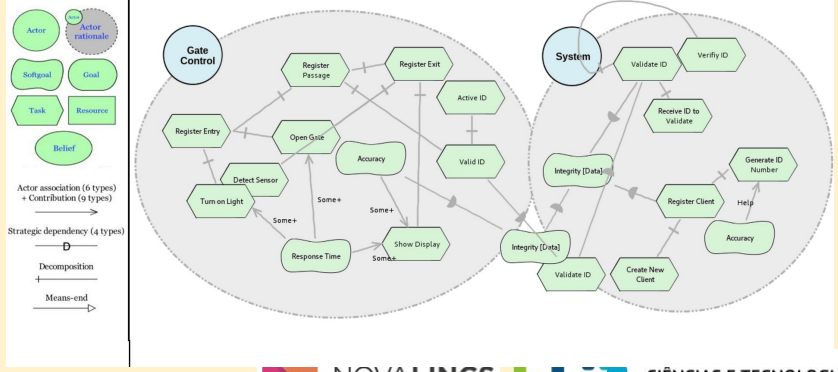
Which tasks are involved in making payments?



Please describe the defects you find in this diagram
(they can be more than one)

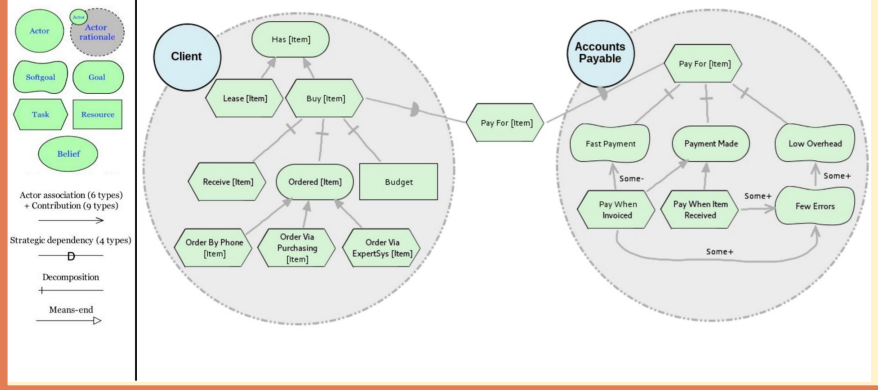


Please describe the defects you find in this diagram
(they can be more than one)

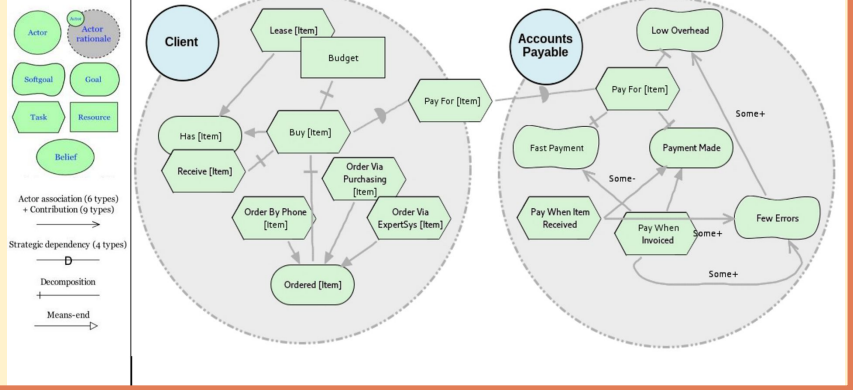


Two **understanding** and two review tasks, both with good and bad layout

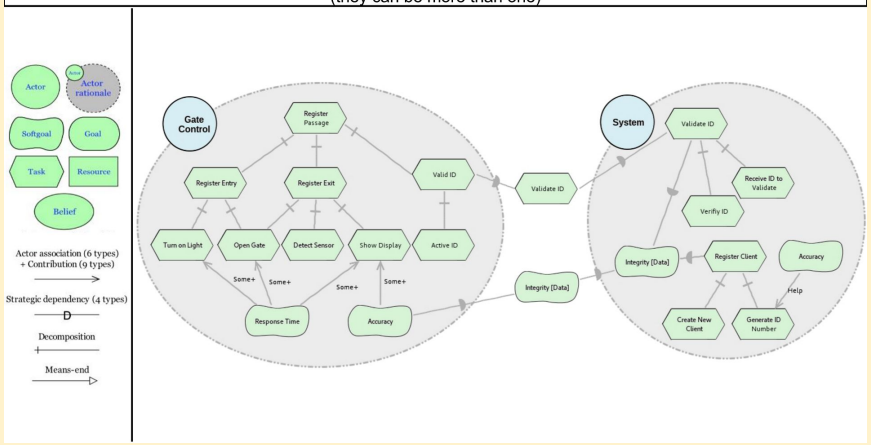
Which tasks are involved in making payments?



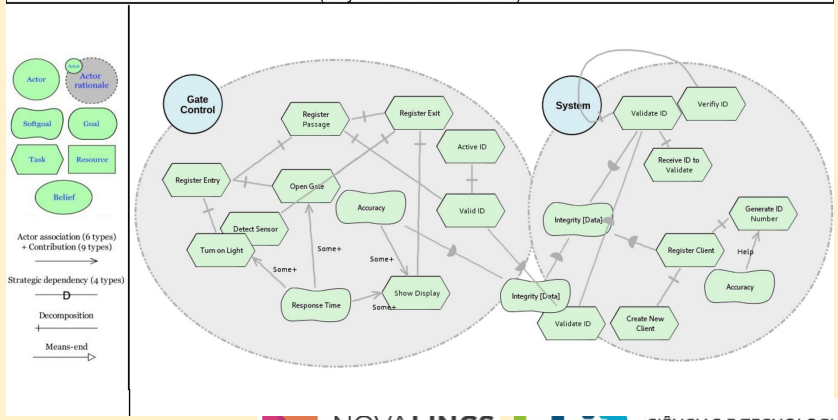
Which tasks are involved in making payments?



Please describe the defects you find in this diagram
(they can be more than one)

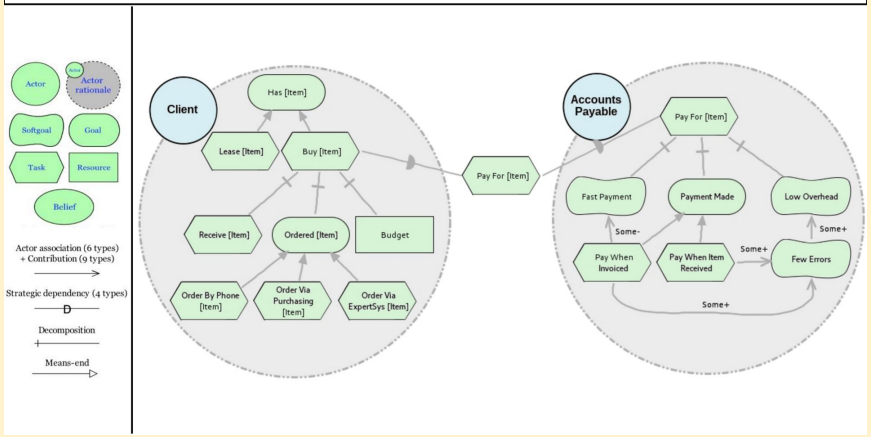


Please describe the defects you find in this diagram
(they can be more than one)

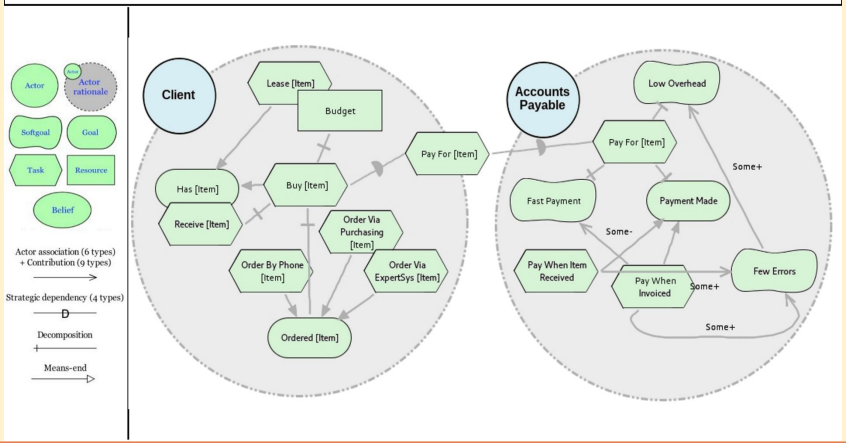


Two understanding and two **review** tasks, both with good and bad layout

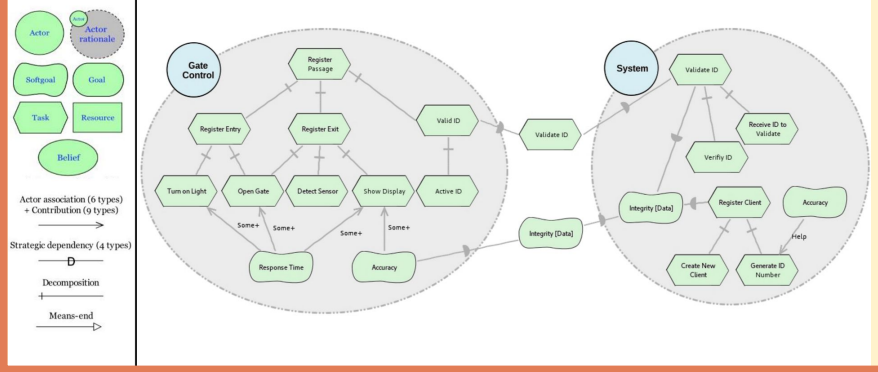
Which tasks are involved in making payments?



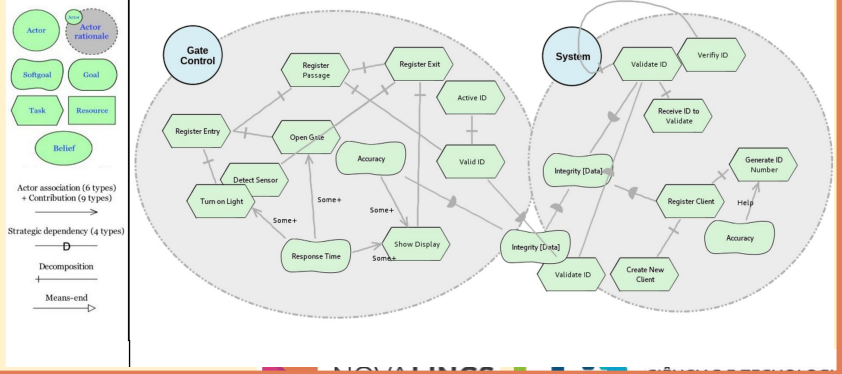
Which tasks are involved in making payments?



Please describe the defects you find in this diagram
(they can be more than one)

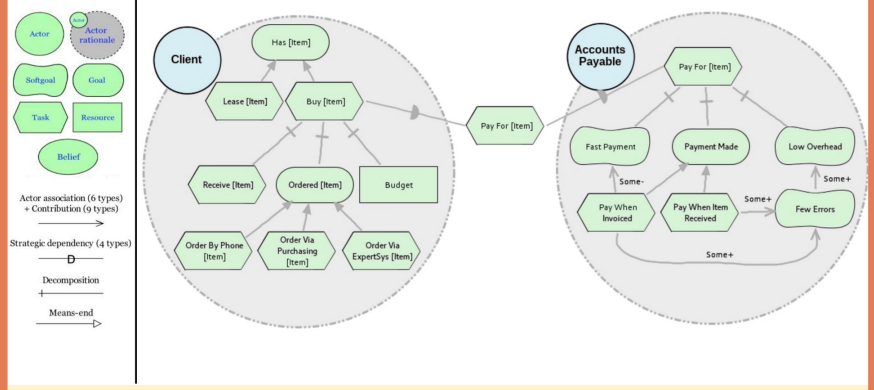


Please describe the defects you find in this diagram
(they can be more than one)

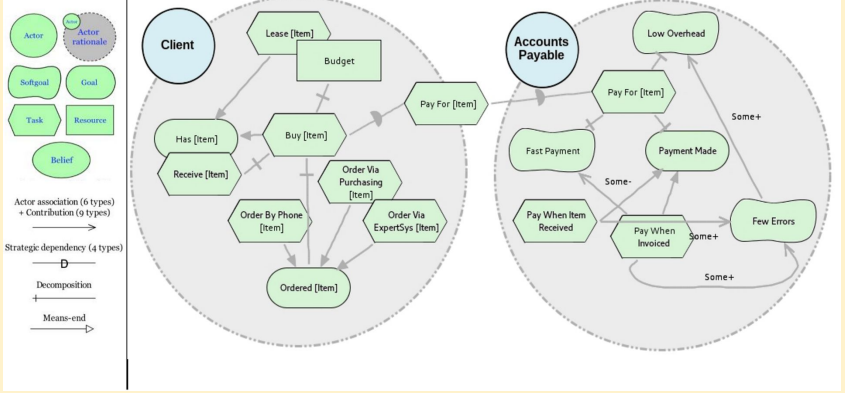


Two understanding and two review tasks, both with **good** and bad layout

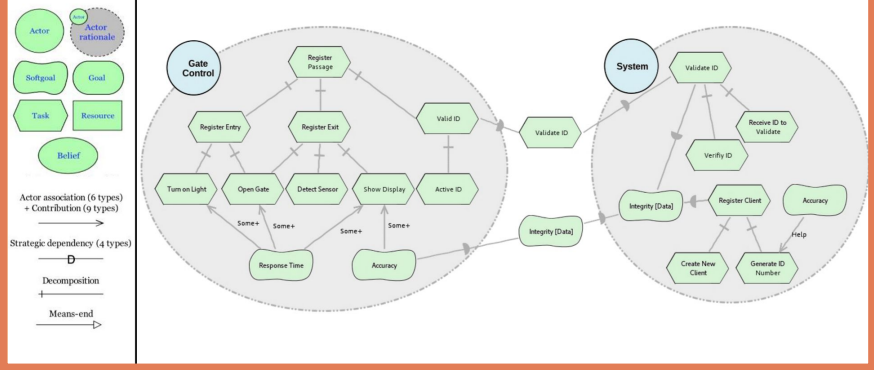
Which tasks are involved in making payments?



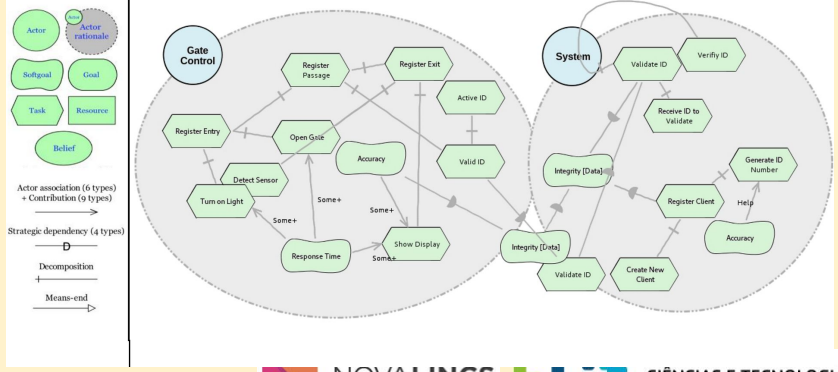
Which tasks are involved in making payments?



Please describe the defects you find in this diagram
(they can be more than one)

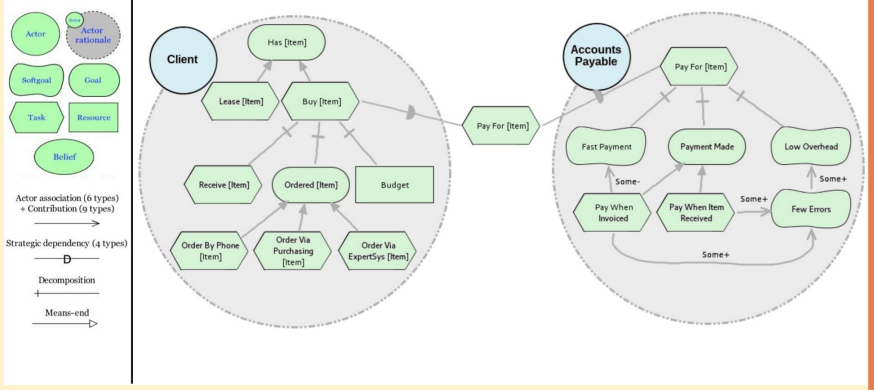


Please describe the defects you find in this diagram
(they can be more than one)

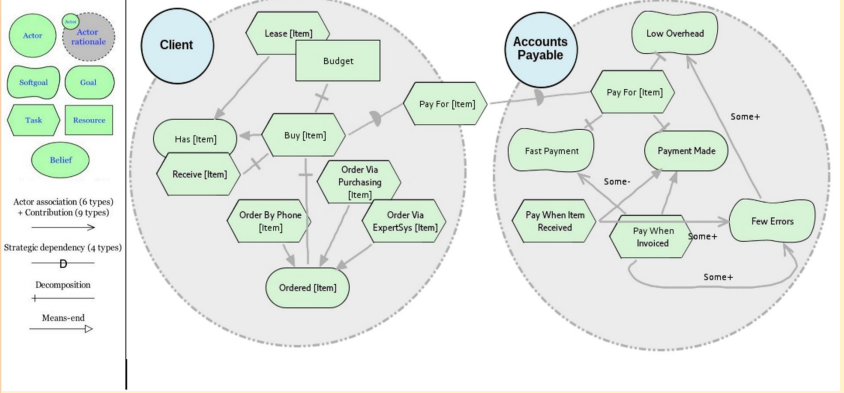


Two understanding and two review tasks, both with good and **bad** layout

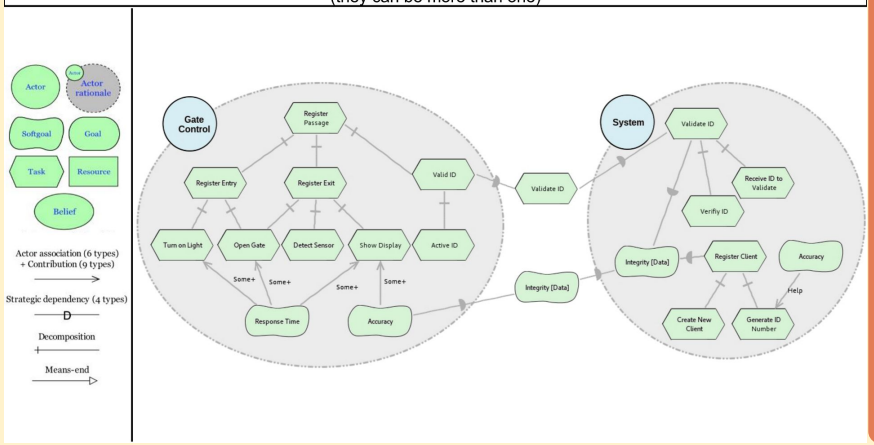
Which tasks are involved in making payments?



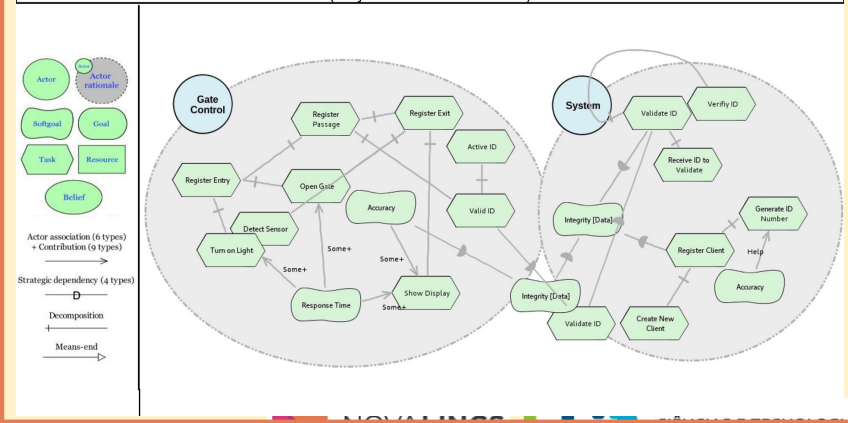
Which tasks are involved in making payments?



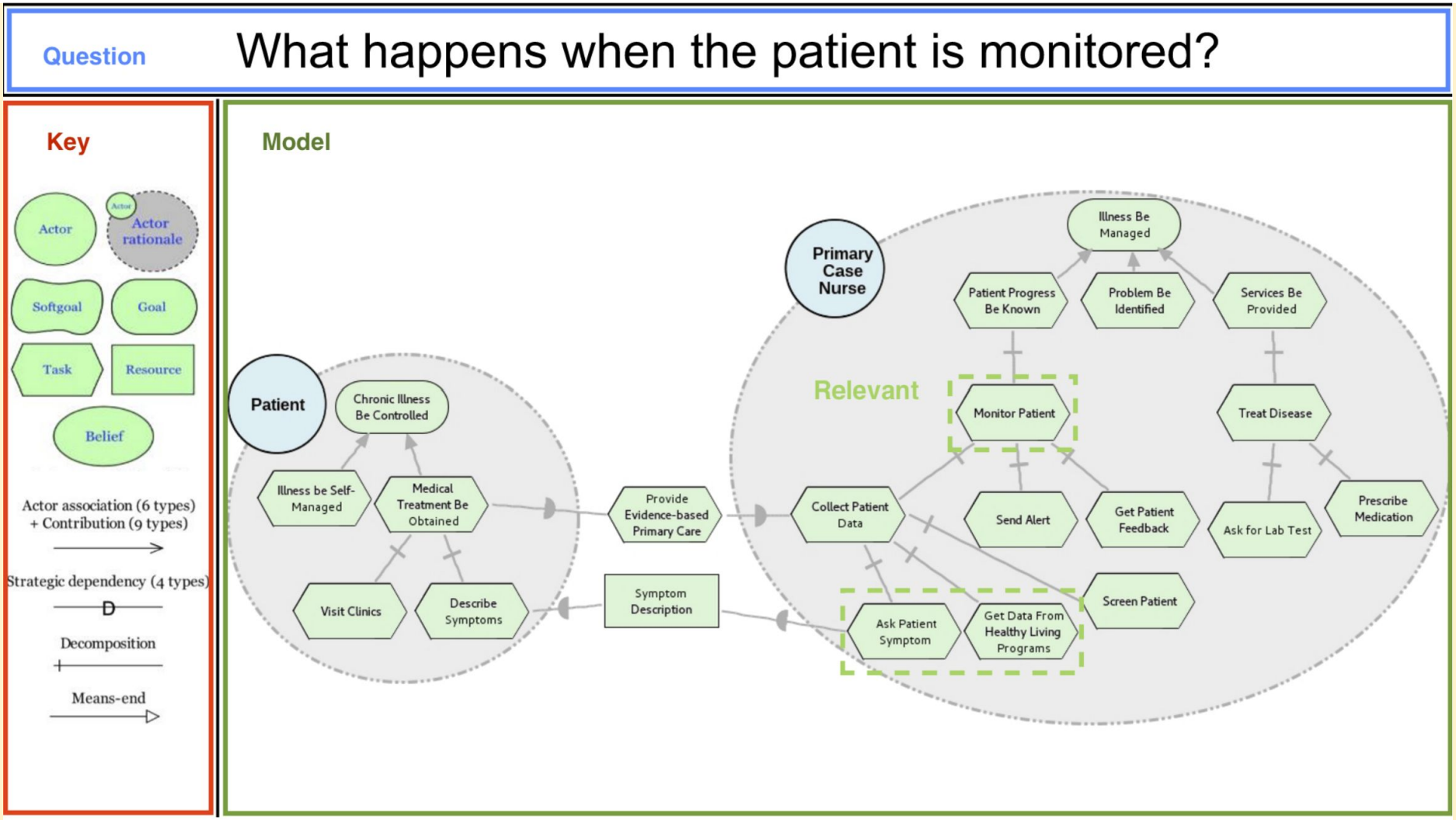
Please describe the defects you find in this diagram
(they can be more than one)



Please describe the defects you find in this diagram
(they can be more than one)



Areas of interest: question, key and model



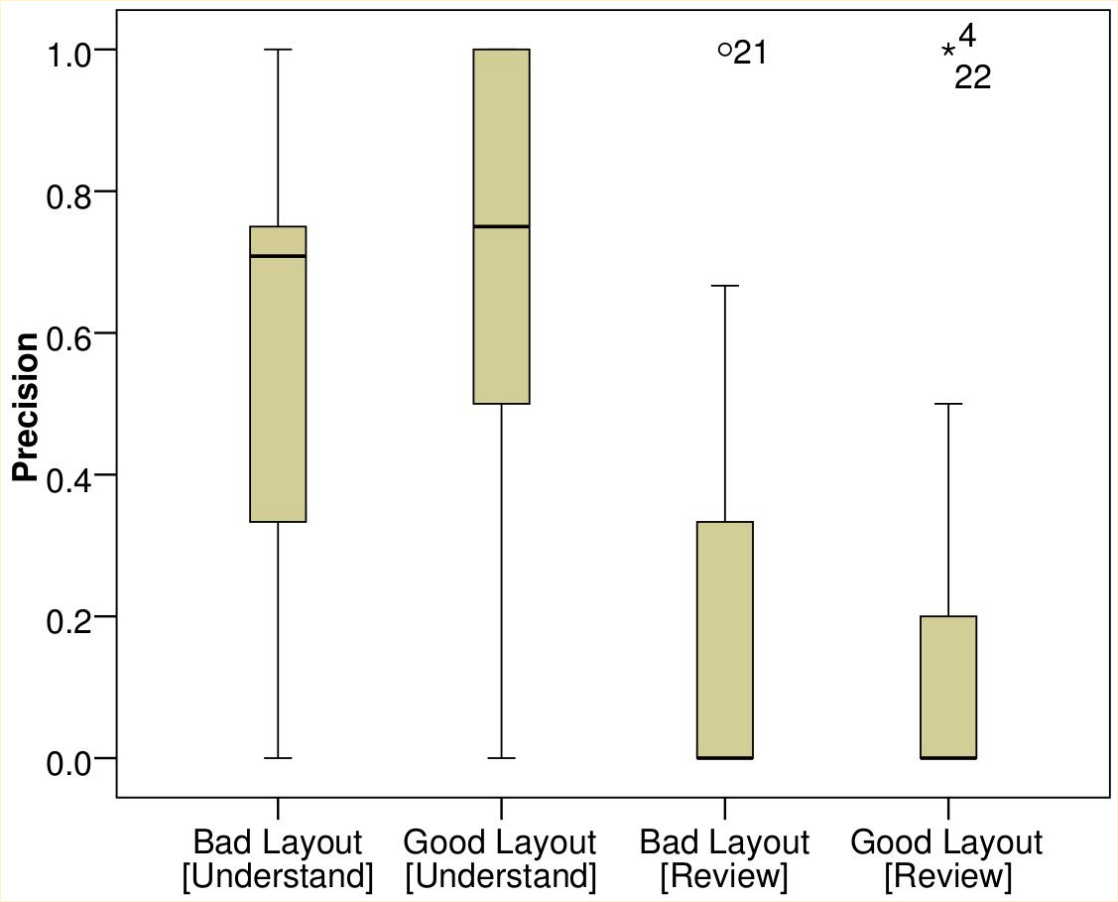
Experiment design

- Participants performed the tasks in different order, to mitigate the risk of learning effects

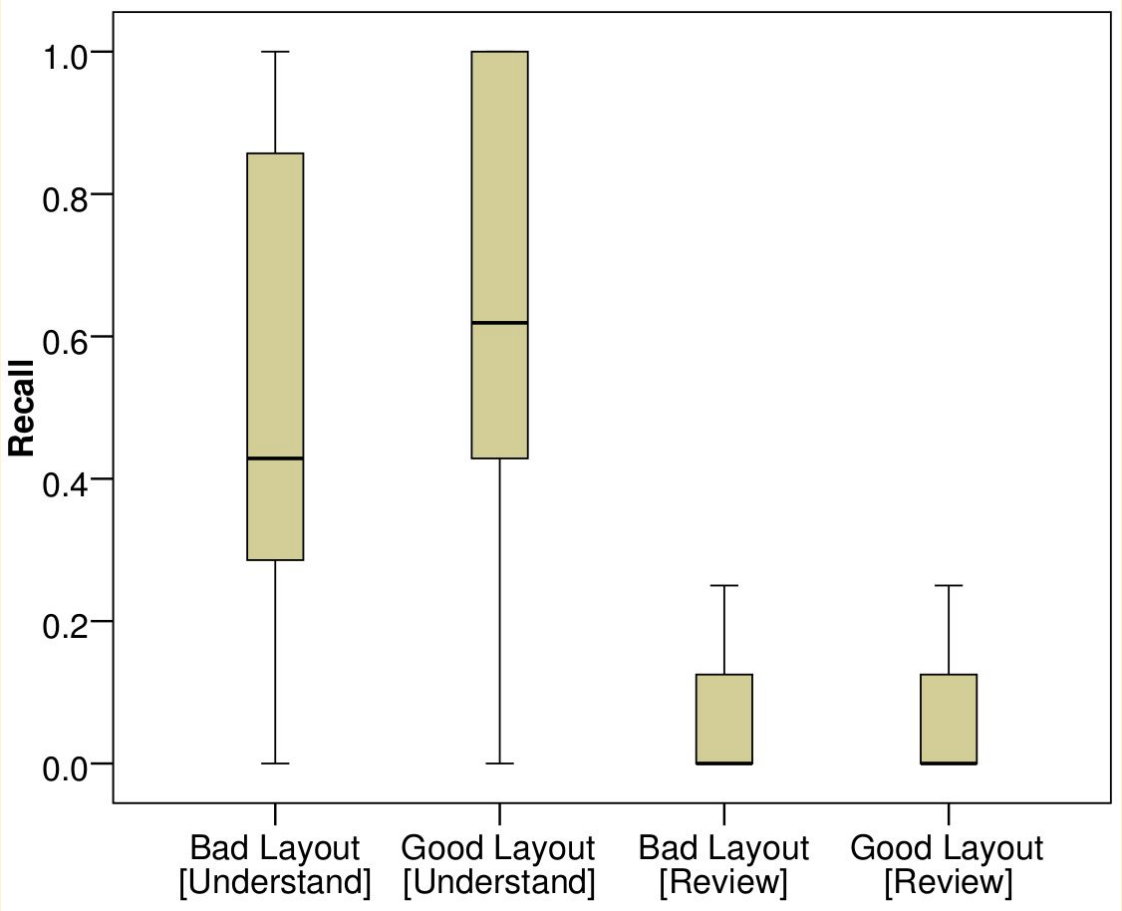
Participant	Letter	Tutorial	T1	T2	T3	T4	Back.
1	✓	✓	U1G	U2B	D3G	D4B	✓
2	✓	✓	D4G	U1B	U2G	D3B	✓
3	✓	✓	U2B	D3G	D4B	U1G	✓
4	✓	✓	D3B	D4B	U1B	U2G	✓
...	✓	✓	...	...	...	...	✓

So what was the impact of good and bad layouts in understanding and reviewing i* models?

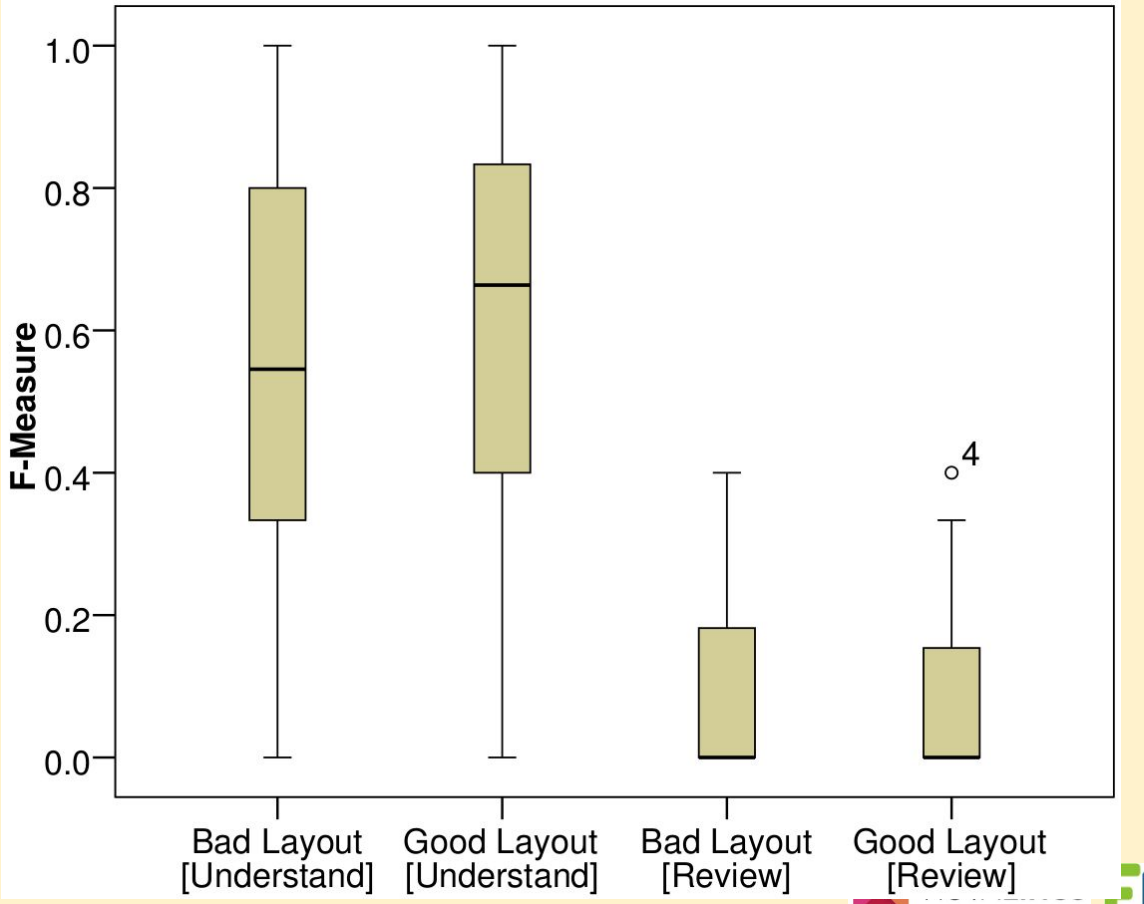
Precision is higher for understanding tasks, but there is no statistically significant difference between layouts



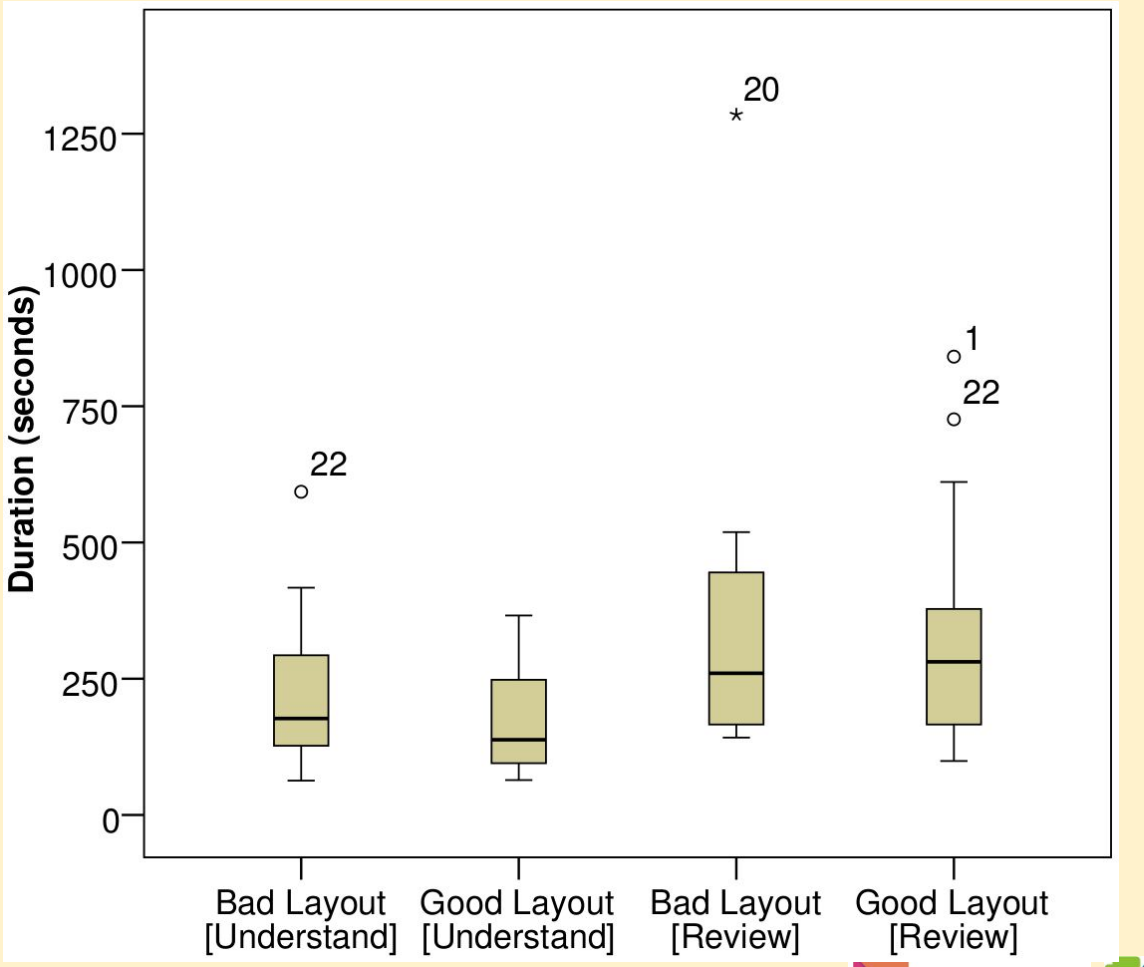
Recall is better for understanding tasks with good layout, but the difference is not statistically significant



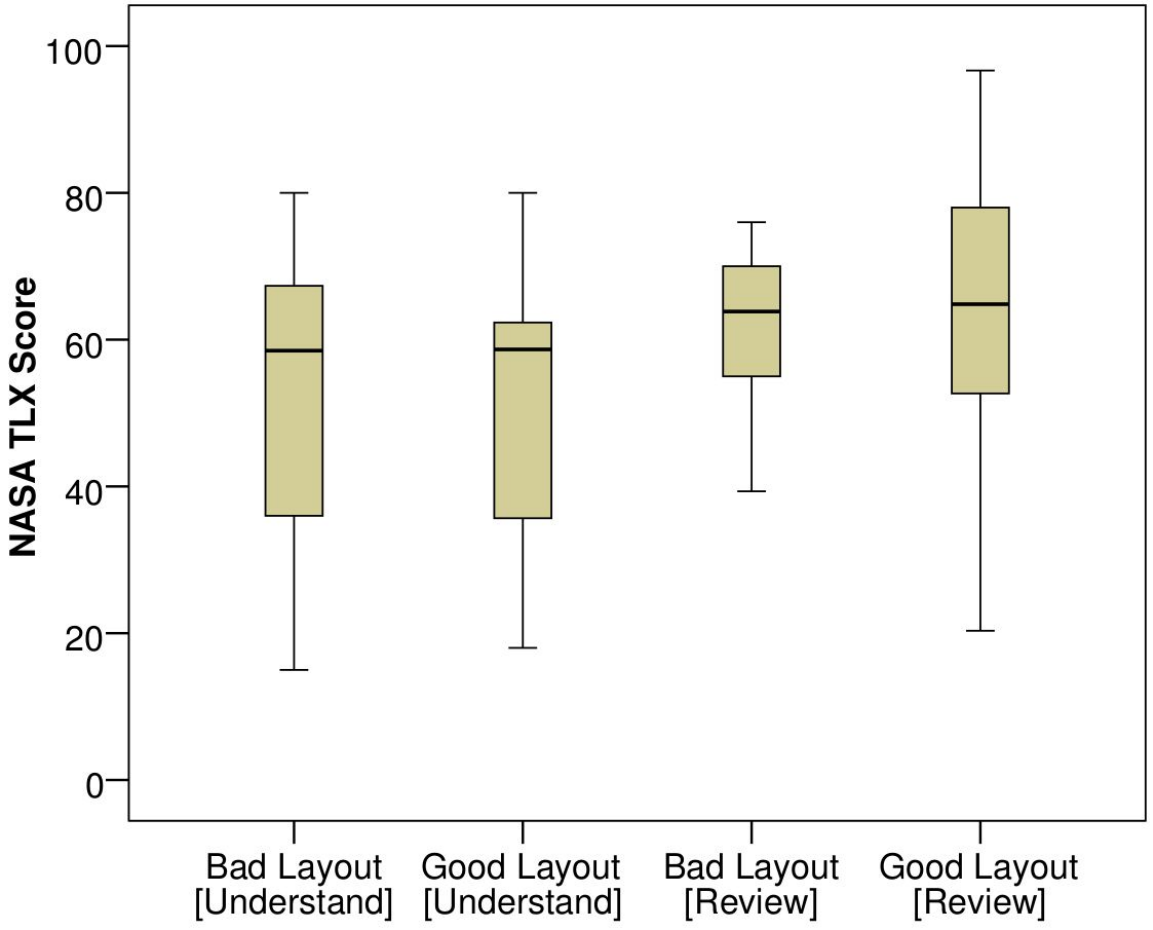
F-Measure is better for understanding tasks with good layout, but the difference is not statistically significant



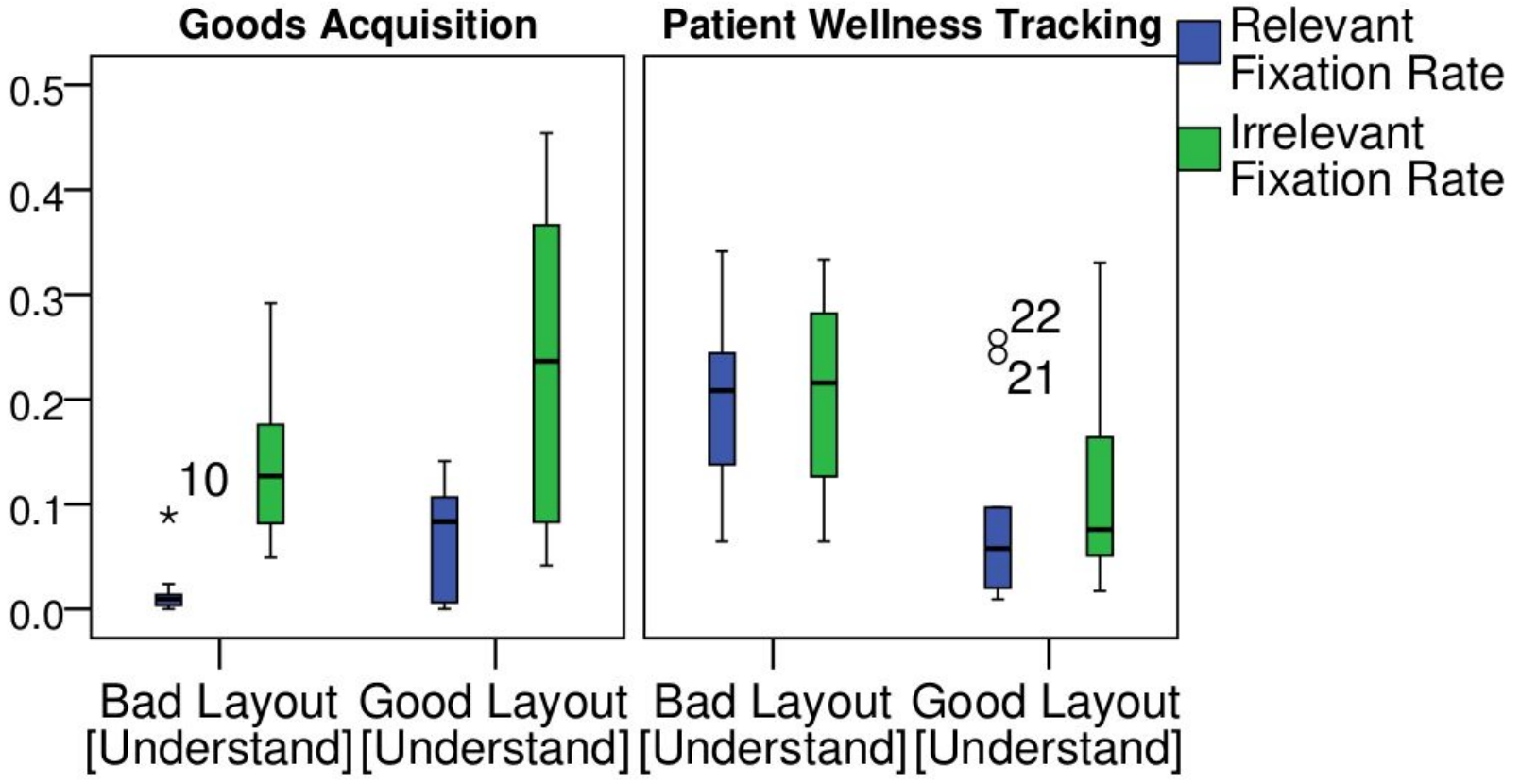
There is no difference in terms of **duration**, between good and bad layouts for both tasks



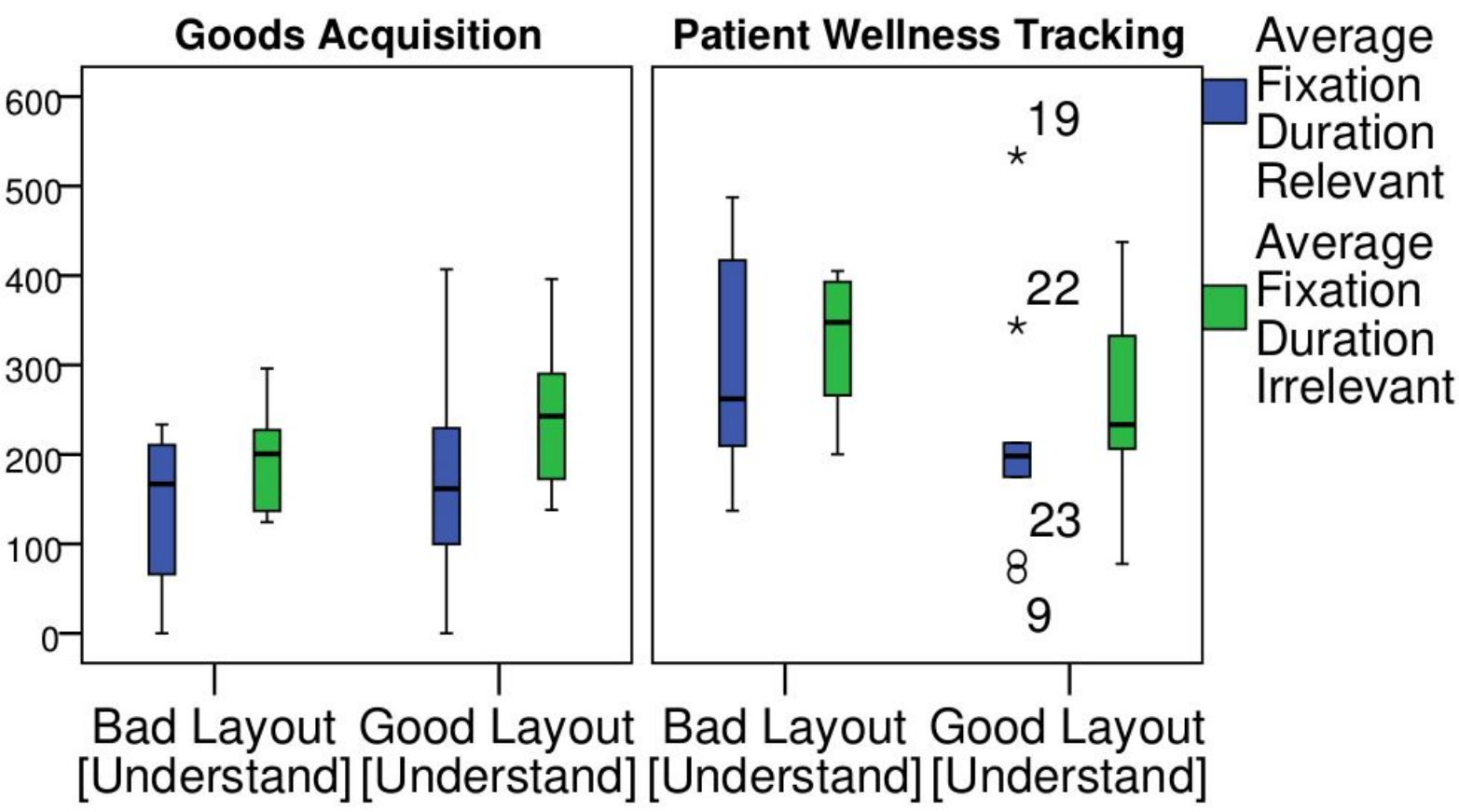
There is no difference in the **perception of complexity** of the tasks, for both layouts



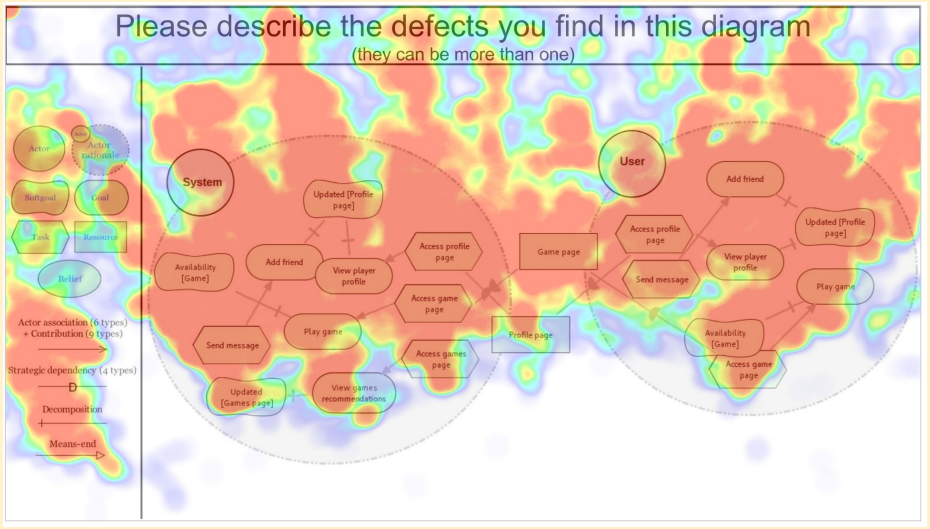
Relevant **fixation rates** superior in understanding tasks, but the difference for layouts is not statistically significant



Average fixation duration varies depending on the model, but the difference is not statistically significant



Heat map for the good layout is less scattered than the one for bad layout



Conclusions

Understanding tasks were more accessible than reviewing tasks

The layout was not a significant factor for any of the measures, for both understanding and review tasks

In heap maps for the review tasks, participants' gaze seemed more dispersed by the noise cause by bad layouts

We expect layout quality to have a stronger impact as diagrams increase in size and complexity

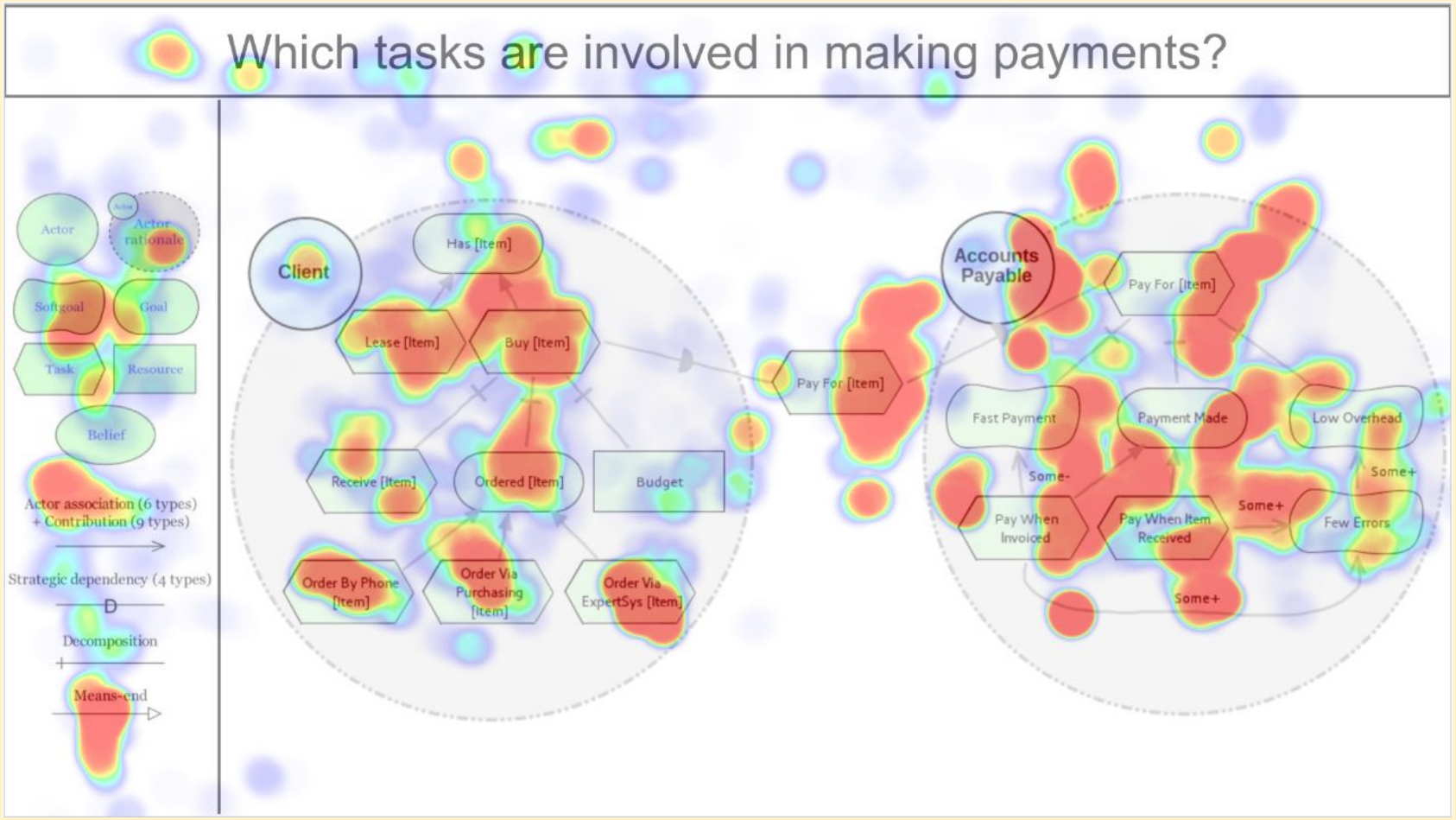
What if we use a better syntax?

Mafalda Santos, Catarina Gralha, Miguel Goulão, João Araújo, Ana Moreira

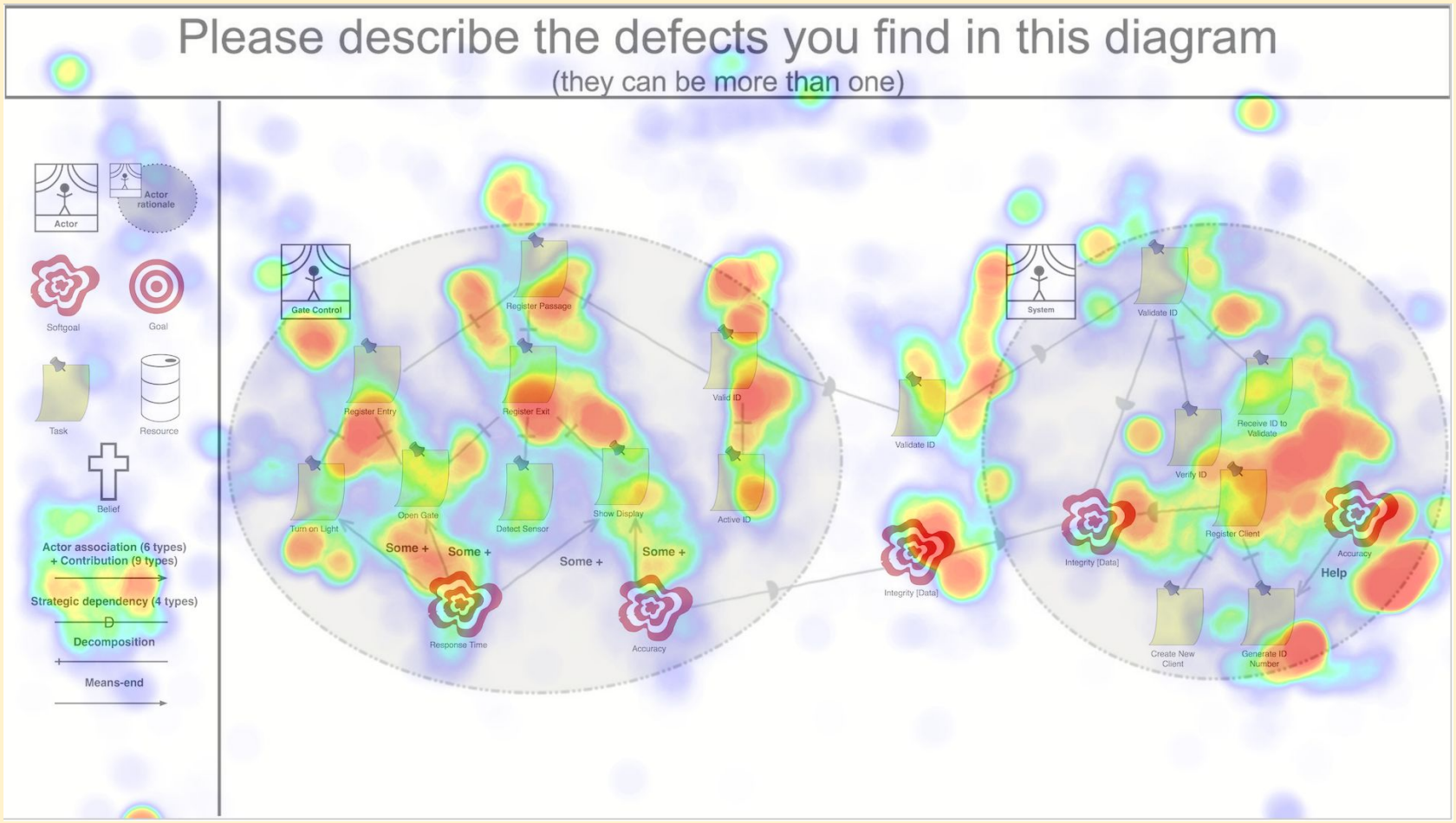
NOVALINCS
FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Understanding task, with improved notation

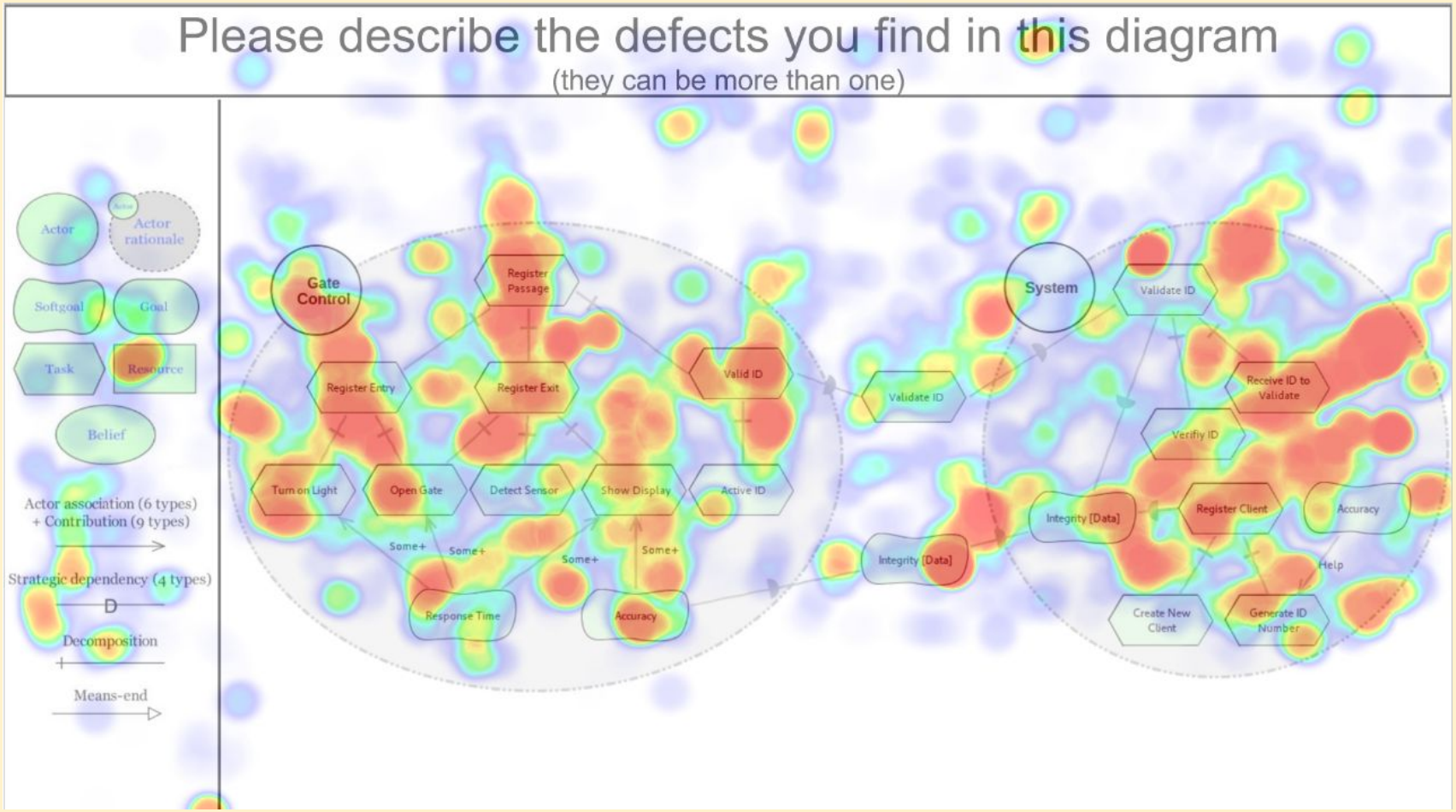
Understanding, with standard notation



Reviewing task, with improved notation



Reviewing task, with standard notation

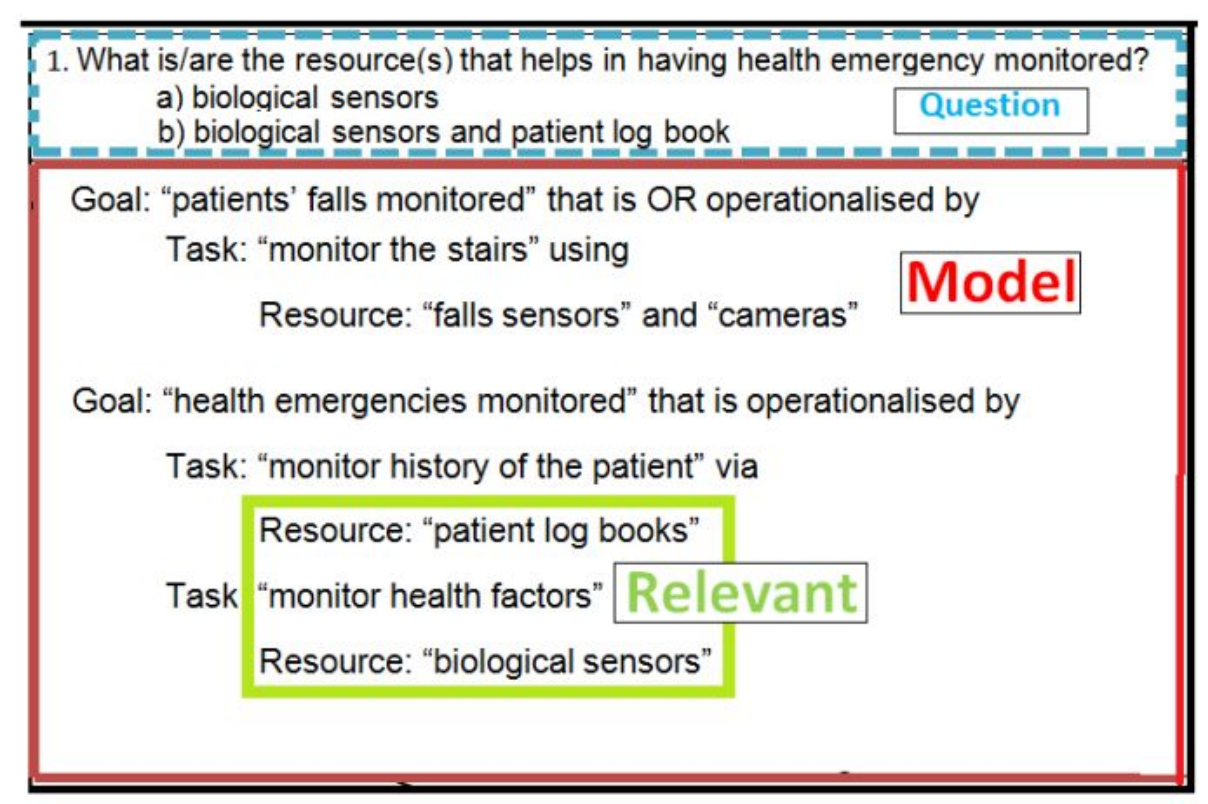


Additional examples

Comparing textual vs graphical requirements notations

Sharafi et al. "An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension", ICPC 2013

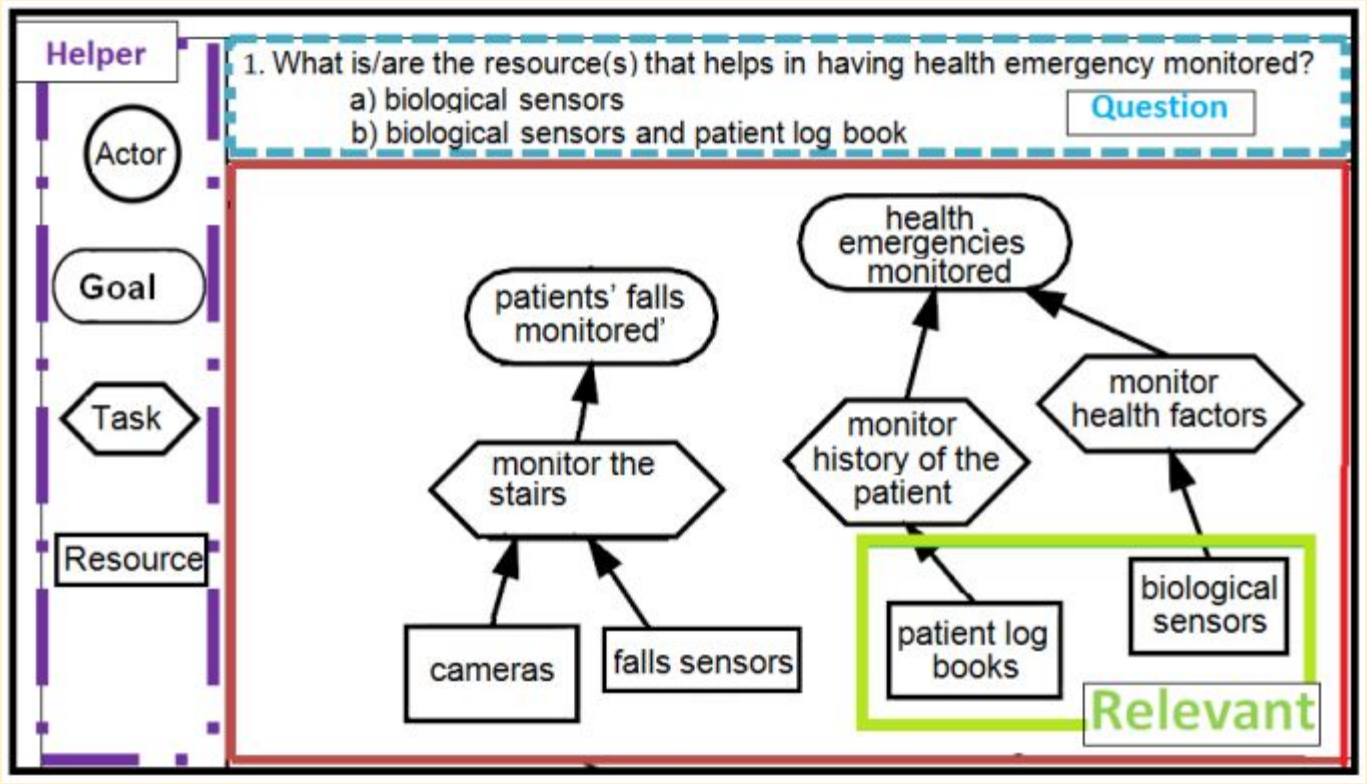
- Textual notation



Comparing textual vs graphical requirements notations

Sharafi et al. "An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension", ICPC 2013

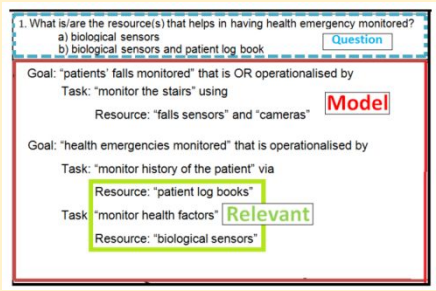
- Graphical notation



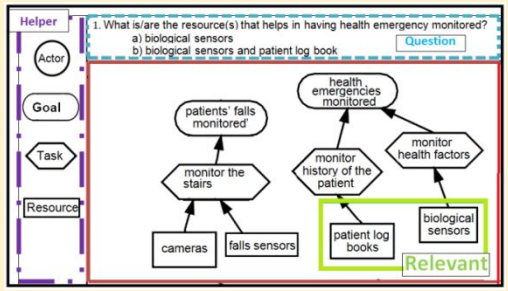
Comparing textual vs graphical requirements notations

Sharafi et al. “An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension”, ICPC 2013

Independent variable: **requirements notation**



vs.



Dependent variables:

- **Accuracy** (percentage of correct answers about these requirements)
- **Time** (taken on each model stimulus)
- **Effort** (amount of visual attention; assumption of positive correlation between amount of visual attention and effort)

Descriptive statistics on collected data

Sharafi et al. “An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension”, ICPC 2013

Collected data	Results
Number of subjects (#)	28
Total number of questions	504
Number of Text-related questions	168
Number of Graphical-related questions	168
Number of Mixed questions	168
Total time of eye-tracking (hours)	2.85
Total number of fixations (#)	50,652

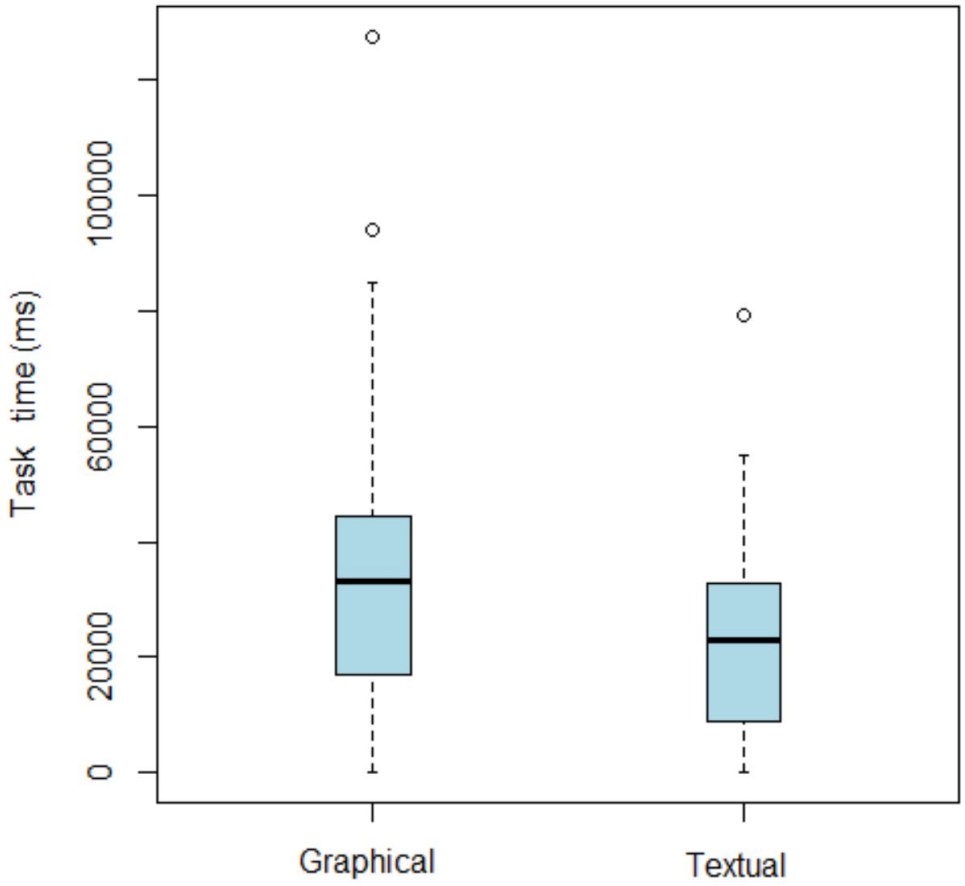
Accuracy was similar in both notations

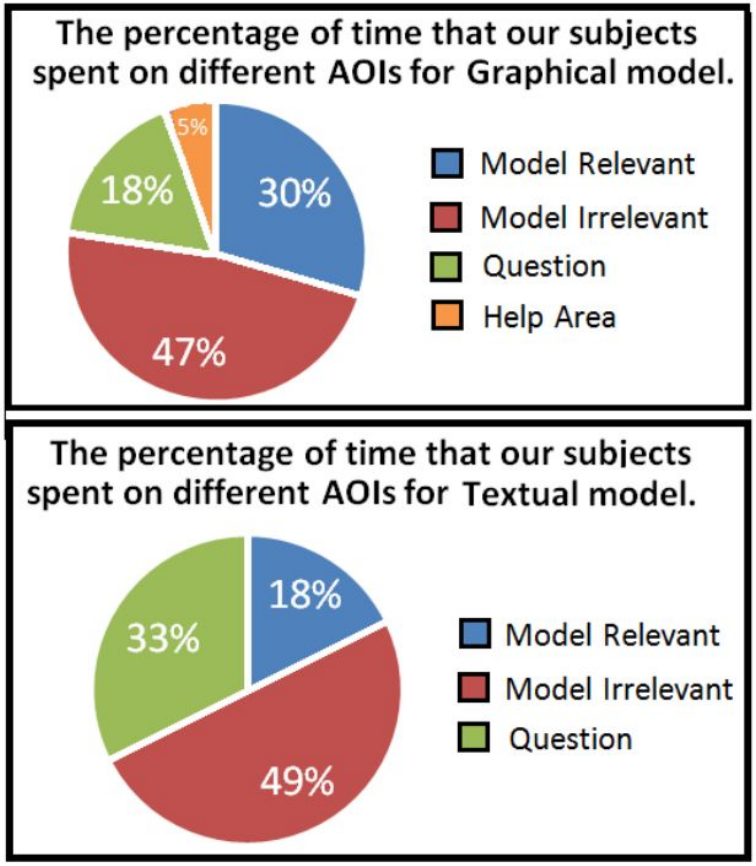
Sharafi et al. “An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension”, ICPC 2013

Answers					
Textual		Graphical		Mixed	
Correct	Wrong	Correct	Wrong	Correct	Wrong
164	4	165	3	162	6
97%	3%	98%	2%	96%	4%

Participants were faster with the textual notation

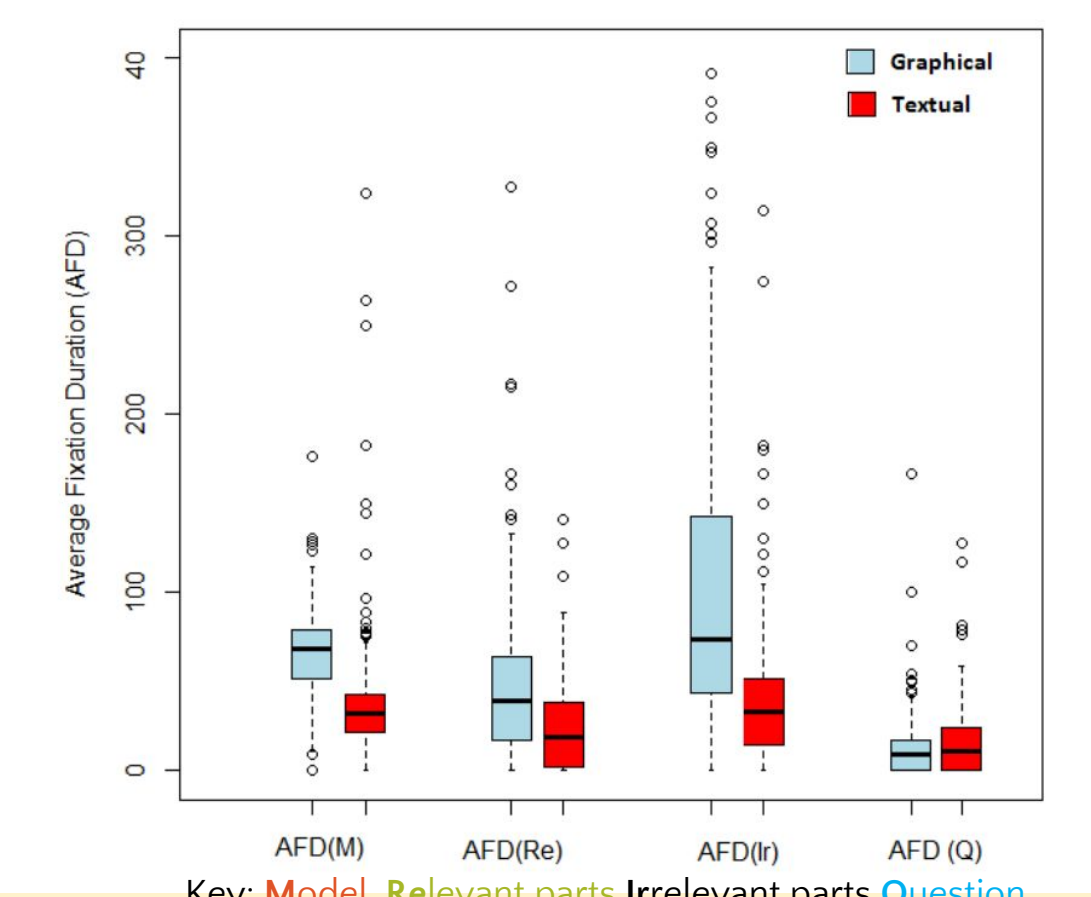
Sharafi et al. “An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension”, ICPC 2013





The graphical notation required more visual effort than the textual notation

Sharafi et al. “An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension”, ICPC 2013



Key: Model Relevant parts Irrelevant parts Question

Other studies not covered here

- Influence of colors, naming conventions, etc in program comprehension
- Influence of layout in diagrams comprehension
- …

Potential implications

- Language improvement
- Tool improvement
- Recruitment
- Training

- Triangulation of techniques to increase soundness of software engineering claims

In the end of the day…

Some usability indicators for a DSL

What to measure?

Development time	Quality of generated model	Help support / Tutorials	Performance
Reuse level	Learnability	Error prevention	Subjective metrics
User bio signals	Adherence to best practices in language design	Maintainability / Evolvability	...

Beware of hidden costs

Evaluate everything

(as long as you make sure there is a real “client” to your usability evaluation BEFORE you actually commit to it)

Examples

DSLs vs GPLs A family of experiments with a detailed report

Tomaž Kosar, Marjan Mernik, Jeffrey C. Carver, "Program comprehension of domain-specific and general-purpose languages: comparison using a family of experiments" Journal of Empirical Software Engineering, 2011

Experiment objectives

- The goal of the experiments is to compare programmers' comprehension of programs written in a DSL vs. their comprehension of programs written in a GPL. A family of experiments is created:

Experiment 1 - Feature diagrams (FD) domain

- provides the ability to capture the variable parts of application domains and provides ways to express which features make up a given system.

Experiment 2 - Graph descriptions (GD) domain

- describes directed and undirected graphs, which consist of nodes and edges. Graphs, subgraphs, nodes and edges have various properties (attributes), e.g., color, shape, and style.

Experiment 3 - Graphical user interface (GUI) domain

- focuses on construction of programs that allow users to visually interact. GUIs are made up of several components, e.g., windows, menus, fields, labels, and buttons.

Context

Languages:

Domain	DSL	GPL
Feature diagrams (FD)	FDL	FD library in Java
Graph descriptions (GD)	DOT	GD library in C
Graphical user interface (GUI)	XAML	Windows forms library in C#

Participants:
undergraduate students from the University of Maribor

Materials

- Materials

	Feature diagrams (FD)		Graph descriptions (GD)		Graphical user interface (GUI)	
	DSL	GPL	DSL	GPL	DSL	GPL
Application1	Menu	Radio	Compiler parts	UML	Registration form	Picture viewer
Application2	Software	Compiler construction	Branching	Flowcharts	Product info	Painter

- Background Questionnaire
  - Participants prior experience with FD, GD, and GUI
  - 10 questions (Likert Scale)

Tests

Learn

- Q1 Select syntactically correct statements.
- Q2 Select program statements with no sense (unreasonable).
- Q3 Select a valid program with the given result.

Understand

- Q4 Select the correct result for the given program.
- Q5 Identify language constructs.
- Q6 Select a program with the same result.
- Q7 Select the correct meaning for the new language construct.
- Q8 Identify correct meaning in the program with comments.

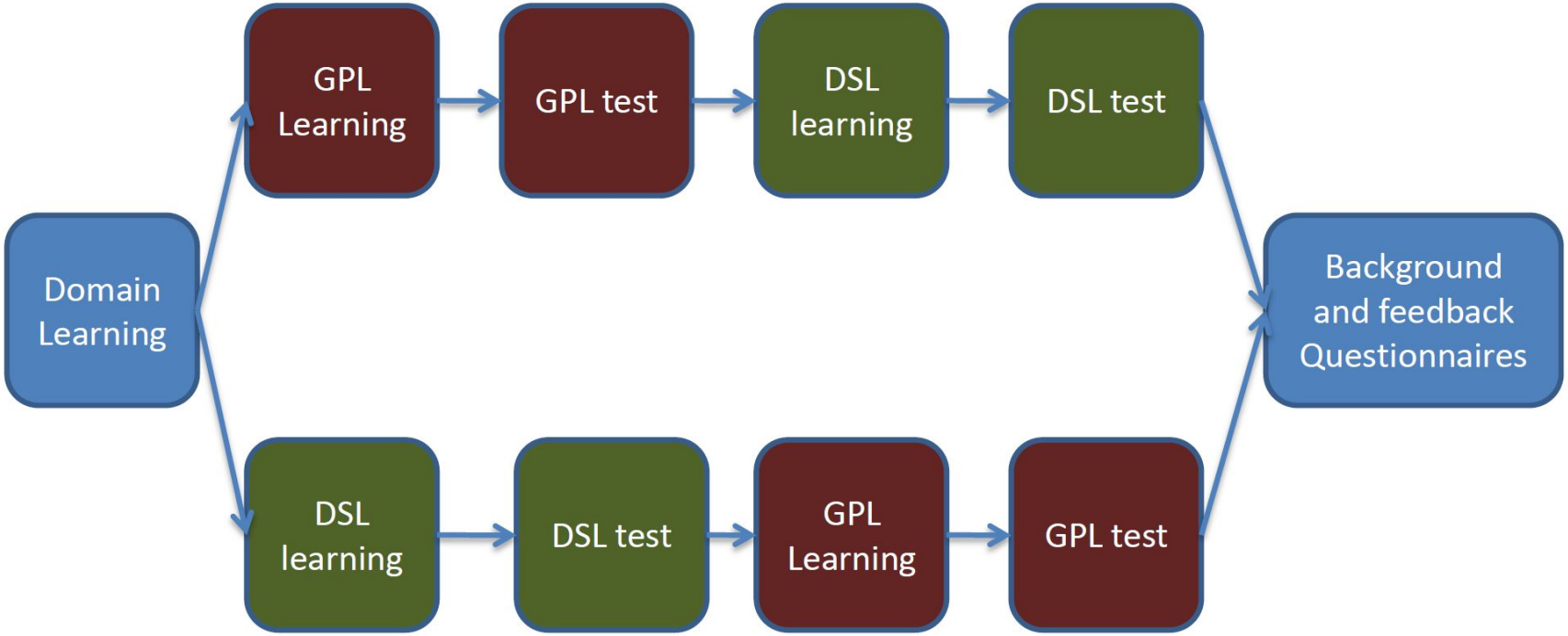
Evolve

- Q9 Expand the program with new functionality.
- Q10 Remove functionality from the program.
- Q11 Change functionality of the program.

Feedback questionnaire

- 8 questions
- 5 points Likert scale
- 2 focuses
  - How well participants understood the DSL and GPL programs
  - Experimental tasks rated:
    - Clarity
    - Consistency
    - DSL vs. GPL

Experimental Procedure



Hypotheses

Hypothesis	
H1*null*	There is no significant difference in the correctness of the participants' program comprehension when using a DSL vs. when using a GPL.
H1*alt*	The use of a DSL significantly increases the correctness of the participants' program comprehension over the use of a GPL.
H2*null*	There is no significant difference in the efficiency of the participants' program comprehension when using a DSL vs. when using a GPL.
H2*alt*	The use of a DSL significantly improves the efficiency of the participants' program comprehension over the use of a GPL.

Independent variables

Language Type:

- DSL: Domain-Specific Language
- GPL API: General-Purpose Language with an API

Domain:

- FD: Feature Diagram
- GD: Graph Descriptions
- GUI: Graphical User Interface

Question Number:

- 1–11

Question Type:

- Learn: questions related to learning the notation and meaning of the program
- Understand: questions related to program understanding
- Evolve: questions that require addition of code

Experience:

- measured with participants' self-evaluation on a 5-point Likert scale (1 = no experience . . . 5 = expert)

Dependent variables

Program Comprehension:

- the percentage of correct answers on the test

Time:

- number of minutes to complete the exam

Efficiency:

- percentage of correct answers divided by number of minutes

Simplicity of Use:

- measured on a 5-point Likert scale (1 = very hard . . . 5 = very easy)

Test Complexity:

- measured on a 5-point Likert scale (1 = very hard . . . 5 = very easy)

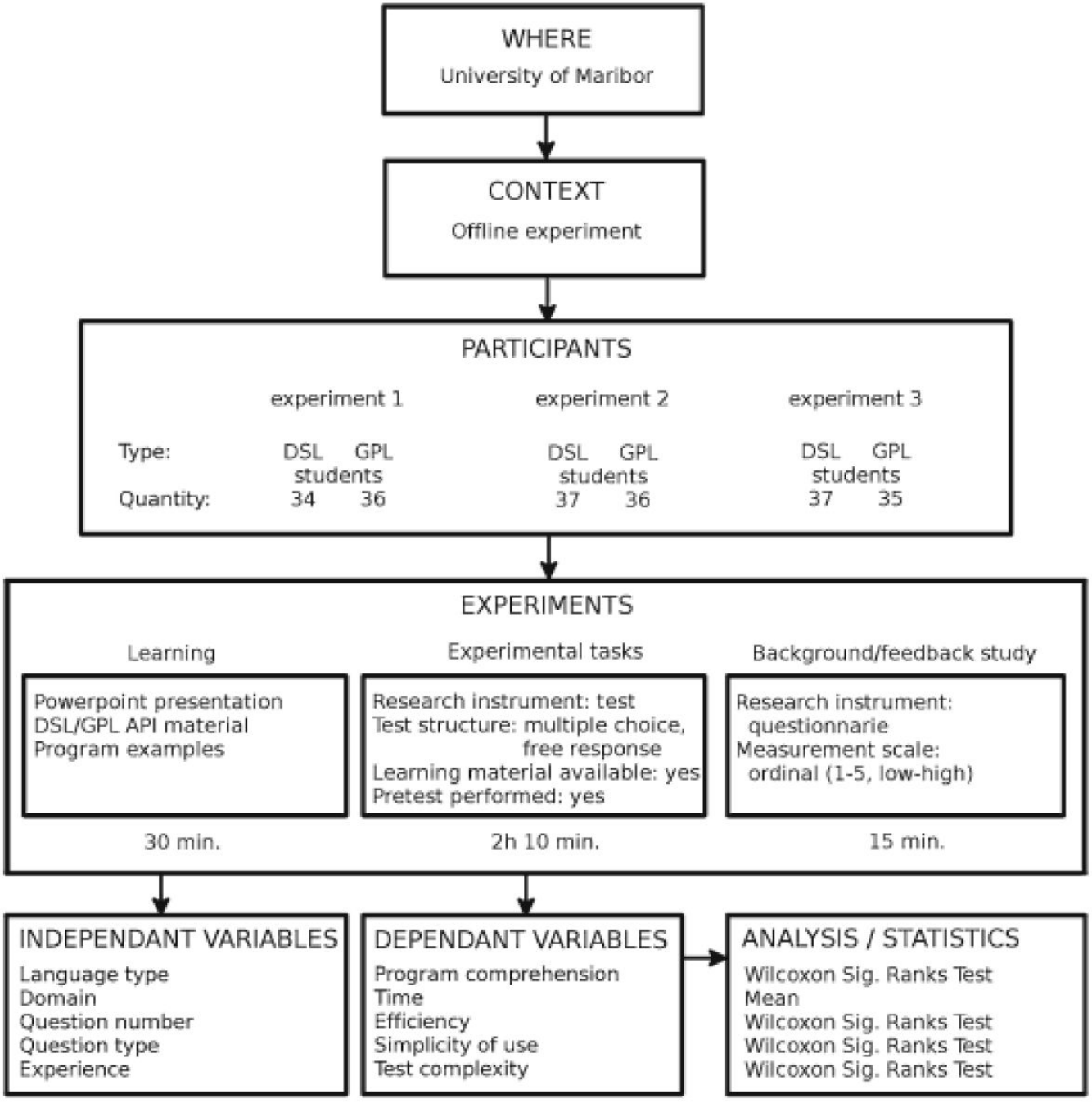
Conclusions

Comprehension correctness

- Significantly better with DSLs than with GPLs
- Similar effect when results are broken by learning, understanding and evolution
- Participants considered the DSL simpler than GPL
- Result significant for FD and GD domains

Comprehension efficiency

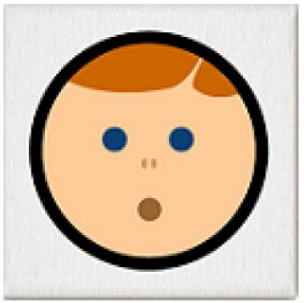
- Participants were faster to complete DSL tests
- Significantly better efficiency (percentage of correct answers by the time required to complete the test)


WHERE
University of Maribor
CONTEXT
Offline experiment
PARTICIPANTS
experiment 1
experiment 2
experiment 3
Type:
DSL GPL
students
DSL GPL
students
DSL GPL
students
Quantity:
34 36
37 36
37 35
EXPERIMENTS
Learning
Experimental tasks
Background/feedback study
Powerpoint presentation
DSL/GPL API material
Program examples
Research instrument: test
Test structure: multiple choice, free response
Learning material available: yes
Pretest performed: yes
Research instrument: questionnarie
Measurement scale: ordinal (1-5, low-high)
30 min.
2h 10 min.
15 min.
INDEPENDANT VARIABLES
Language type
Domain
Question number
Question type
Experience
DEPENDANT VARIABLES
Program comprehension
Time
Efficiency
Simplicity of use
Test complexity
ANALYSIS / STATISTICS
Wilcoxon Sig. Ranks Test
Mean
Wilcoxon Sig. Ranks Test
Wilcoxon Sig. Ranks Test
Wilcoxon Sig. Ranks Test

Pheasant vs. C++/BEE Library

- Two different treatments (the languages)

- Three different user profiles



Informed Programmers

Uninformed Programmers

Non-Programmers

Chosen tasks

- As it is expensive to do all we concentrated on:
  - **writing**
  - reading
  - interpretation
  - comprehension
  - memorization
  - **problem solving**

Variables

- Independent Variables:
  - Language Type
  - Prior education and experience
  - Level of difficulty of tasks
  - Language features
- Dependent Variables:
  - Time
  - Errors
  - Satisfaction

Hypothesis

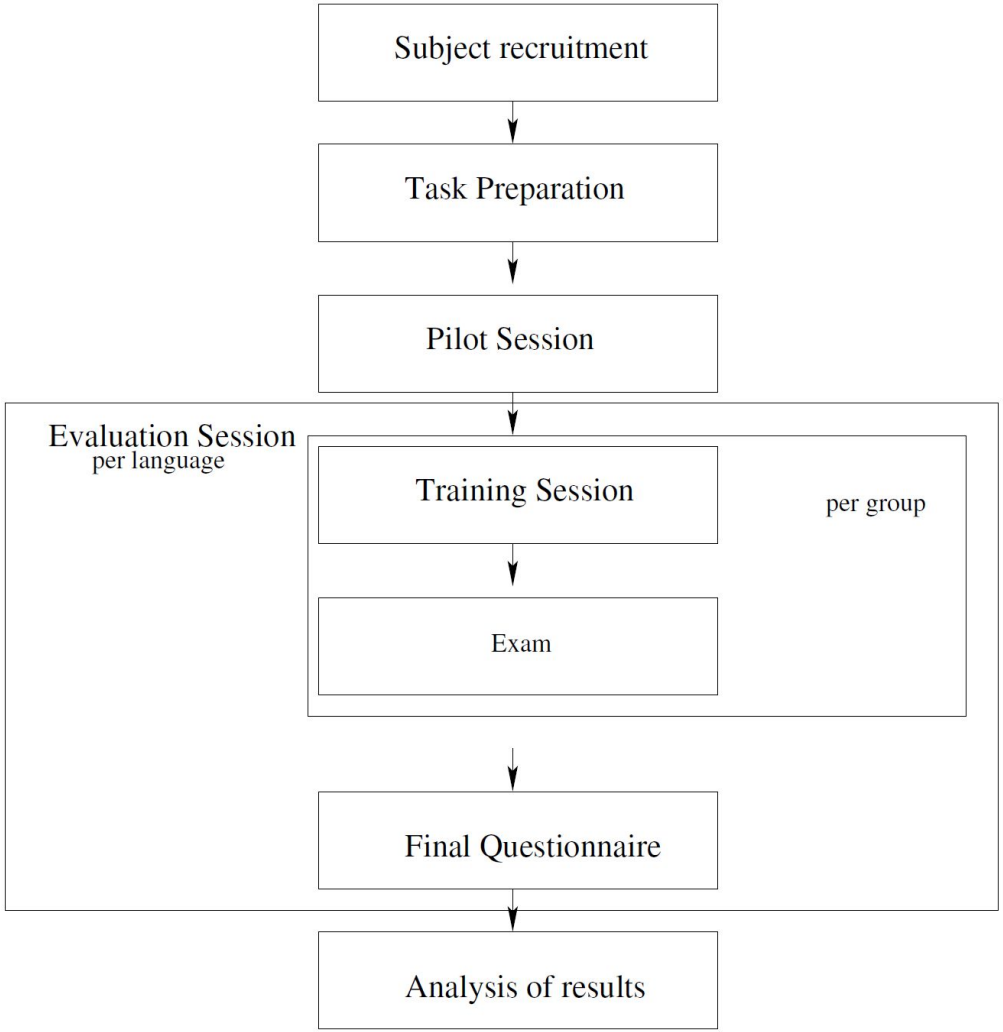
(H1) Using Pheasant increases the effectiveness in query specification

- Less error-prone than the alternative
- DSL allows non-programmers to correctly define queries
- Baseline approach was too difficult for non-programmers

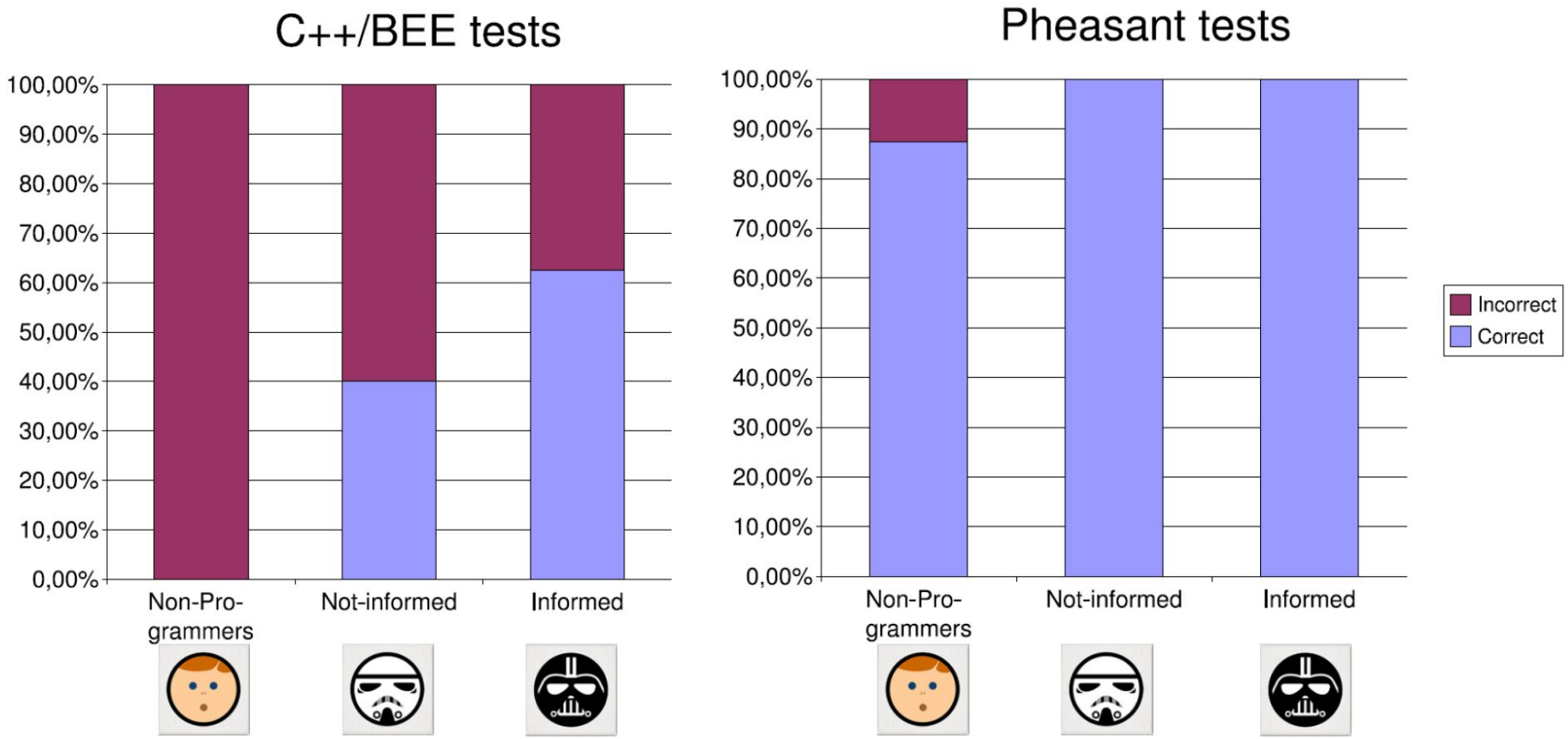
(H2) Using Pheasant increases the efficiency in query specification

- Considerable speedup in query definition, in all controlled groups of users, using Pheasant

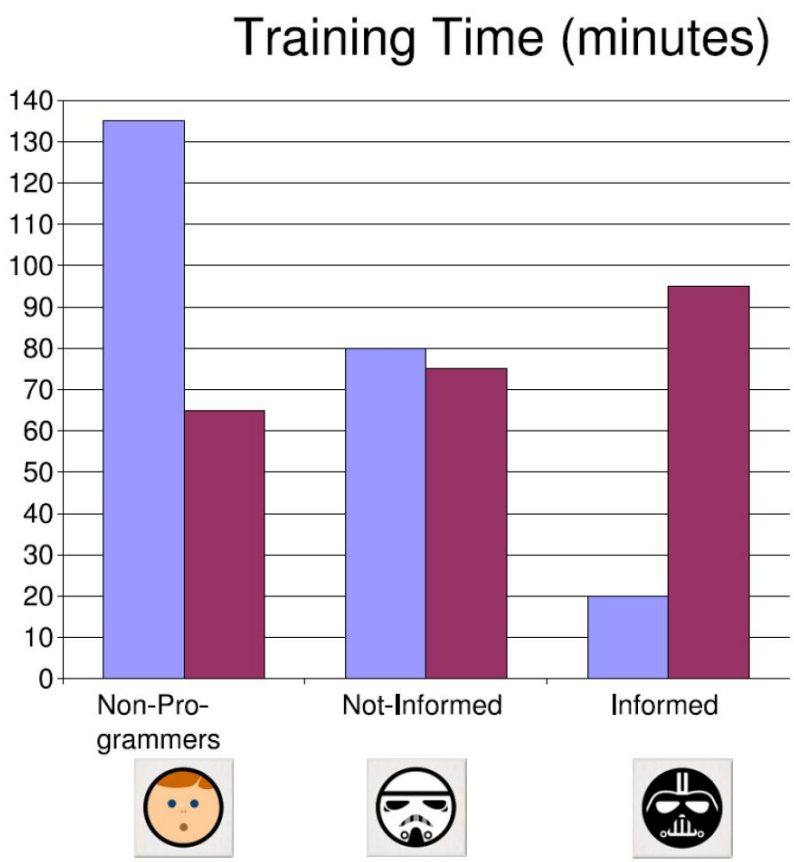
(H3) Users find more usable Pheasant than the alternative

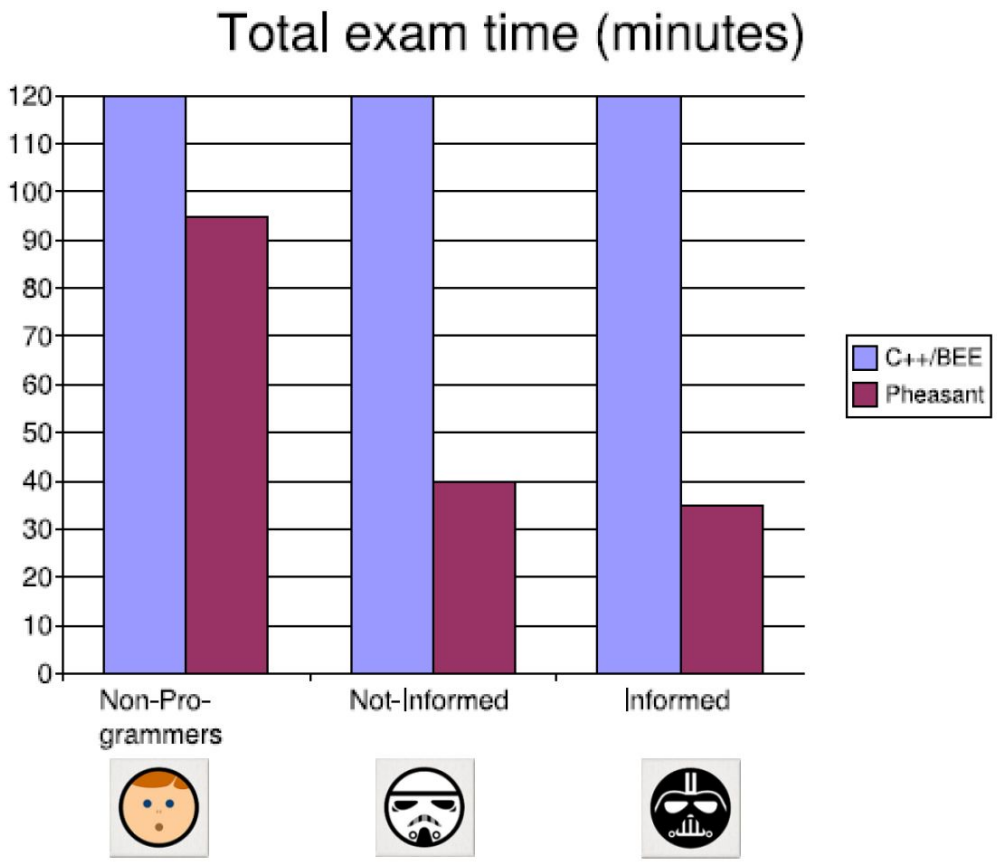

Subject recruitment
Task Preparation
Pilot Session
Evaluation Session
per language
Training Session
per group
Exam
Final Questionnaire
Analysis of results

(H1) Effectiveness – errors



(H2) Efficiency – resulting time





MODELS 2018

Improving the Developer Experience with a Low-Code Process Modelling Language

Henrique Henriques, Hugo Lourenço
OutSystems
Linda-a-Velha, Portugal
(henrique.henriques|hugo.lourenco)@outsystems.com

Vasco Amaral, Miguel Goulão
NOVA LINCS, DI, FCT/UNL
Lisboa, Portugal
(vma|mgoul)@fct.unl.pt

ABSTRACT

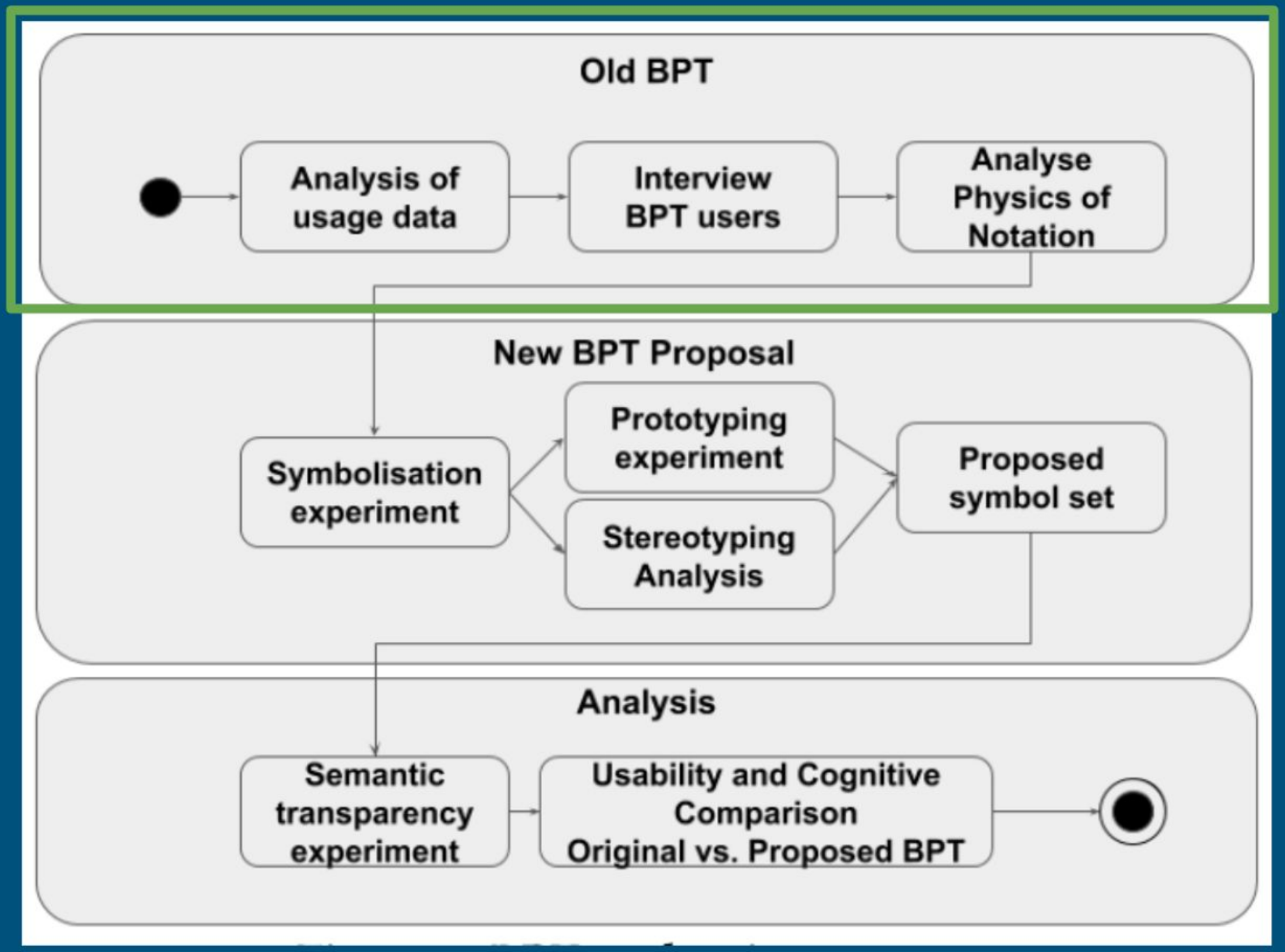
Context: The OutSystems Platform is a development environment composed of several domain-specific languages, used to specify, quickly build and validate web and mobile applications. The languages allow users to model different perspectives such as interfaces and data models, define custom business logic and construct process models. **Problem**: The DSL for process modelling (Business Process Technology (BPT)), does not have the desired adoption rate and used as having usability problems hampering its adoption. matic given the language maintenance costs. **Method**: tion of techniques, including interviews, a critthe "Physics of Notation" and empirical System Usability Scale (SUS) and the develop a new version of BPT

1 INTRODUCTION

Modelling Languages are increasingly adopted in industry. Improving the developer experience with those languages has a potential economic impact both by facilitating their adoption and by making developers more productive. Moody's seminal work on the "Physics of Notations" [22] has raised awareness to the importance of effective visual notations. However, there is scarce evidence and examples of industry strength studies highlighting these benefits. The *OutSystems Platform* is used to create web and mobile applications with a set of integrated Domain-Specific Languages (DSLs). These DSLs are visual modelling languages that allow developing applications at a high abstraction level, hiding low-level details about creating and publishing those applications. This allows signif-
icantly faster development times and a higher quality result when

outsystems

BPT - Evaluation Process



Analysis of Usage Data

Analysed a repository of real application models:

- 5145 OutSystems application models available in the OutSystems platform, only 179 (around 3%) used BPT.
- total of 353 BPT models, of which 120 used process metadata. The 353 BPT models had an average of 18.7 nodes per BPT

Interview

Development teams liked BPT but were “scared” to do so (low level nuances)

Preferred to fallback to what they were used to (Timers).

Parallelism was hard to model and so was identifying synchronisation bugs.

New team members could not start working with BPT without specialised training.

Some clients explicitly requested the use of BPT.

Maintaining a project with BPT was difficult and costly.

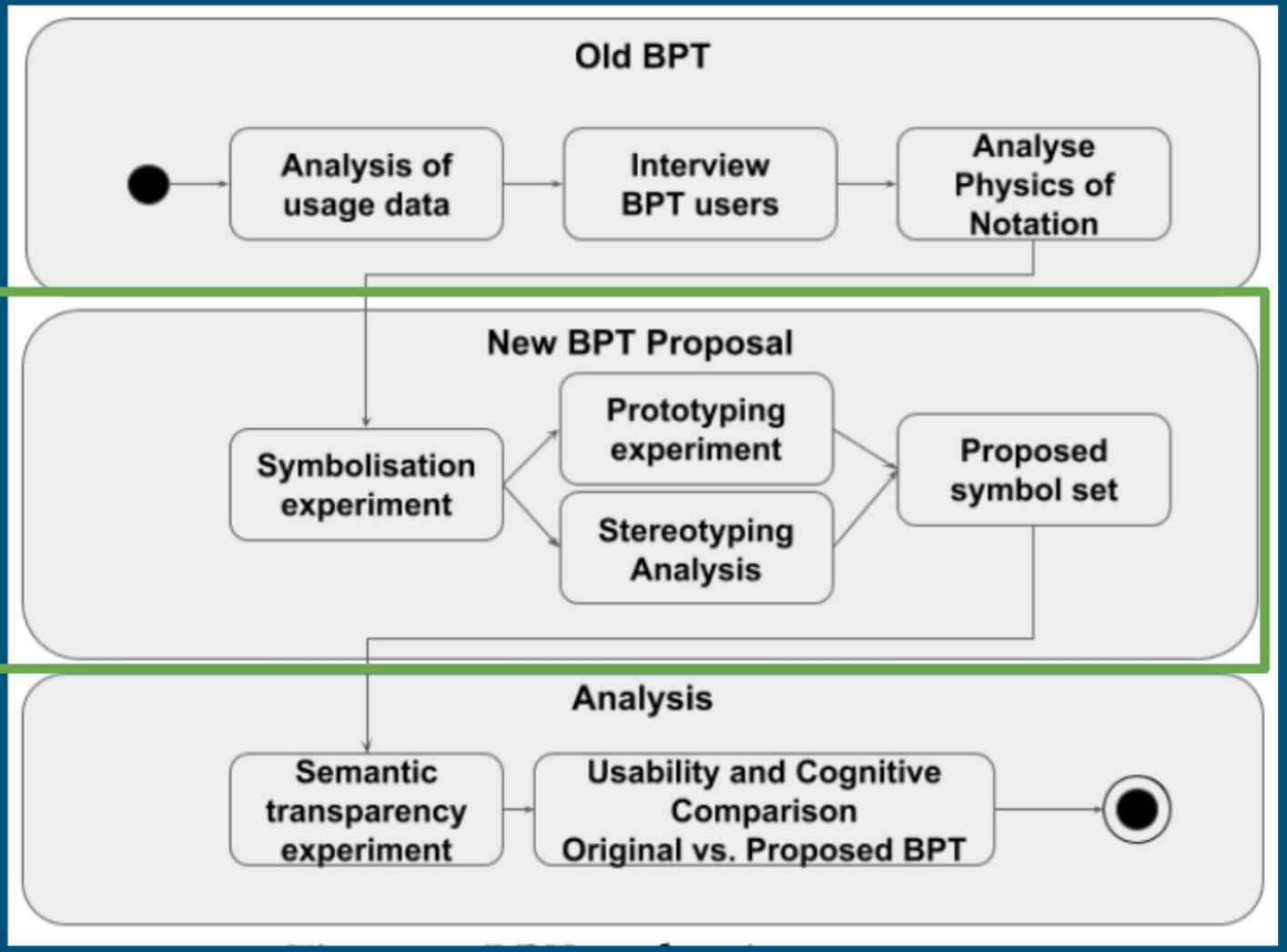
BPT was often used outside of its domain. Used for event handling small (processes normally around three or four nodes) instead process modelling.

They start automatically in response to an event (like an API call), perform a small automatic action and then end. The problem is that the language runtime was not designed for this behaviour and as such does not perform well.

Physics of Notation

Critical analysis of BPT, following the Physics of Notations (PoN) principles

BPT - Evaluation Process

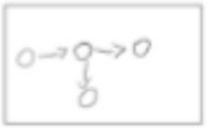


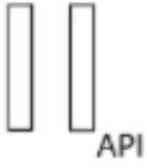
Prototyping

							
Wait for timeout	Fork	Subprocess	Start	Send email	Wait for DB event	End	Decision

						
Custom logic	Alternative flow	Join	Terminate	Wait for user	Broadcast DB or API event	Wait for API

Stereotype symbols

							
Wait for timeout	Fork	Subprocess*	Start	Send email	Wait for DB event	End	Decision

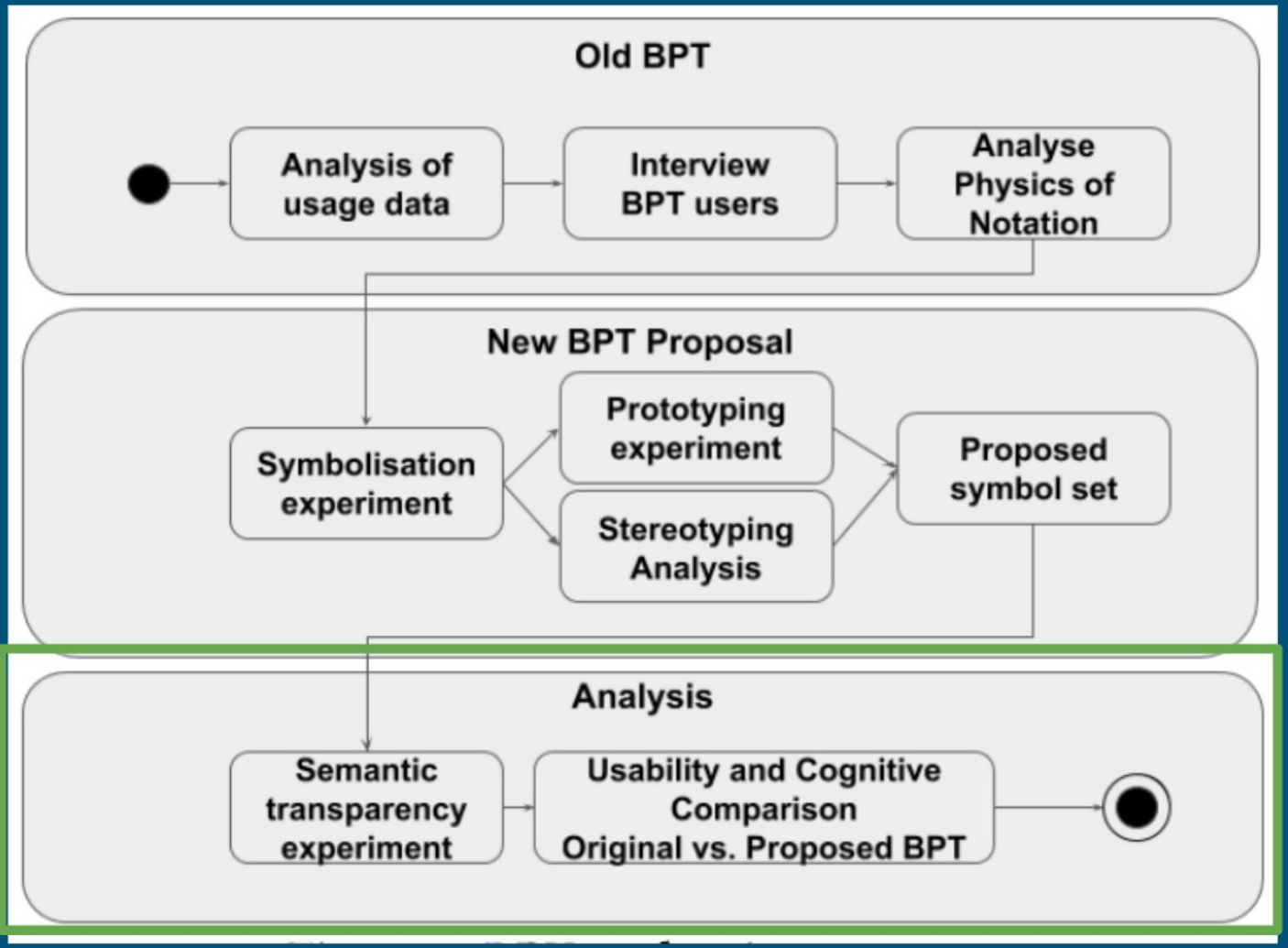
						
Custom logic	Alternative flow	Join	Terminate	Wait for user	Broadcast DB or API event	Wait for API

Set of proposed symbols

							
Wait for timeout	Fork	Subprocess	Start	Send email	Wait for DB event	End	Decision

						
Custom logic	Alternative flow	Join	Terminate	Wait for user	Broadcast DB or API event	Wait for API

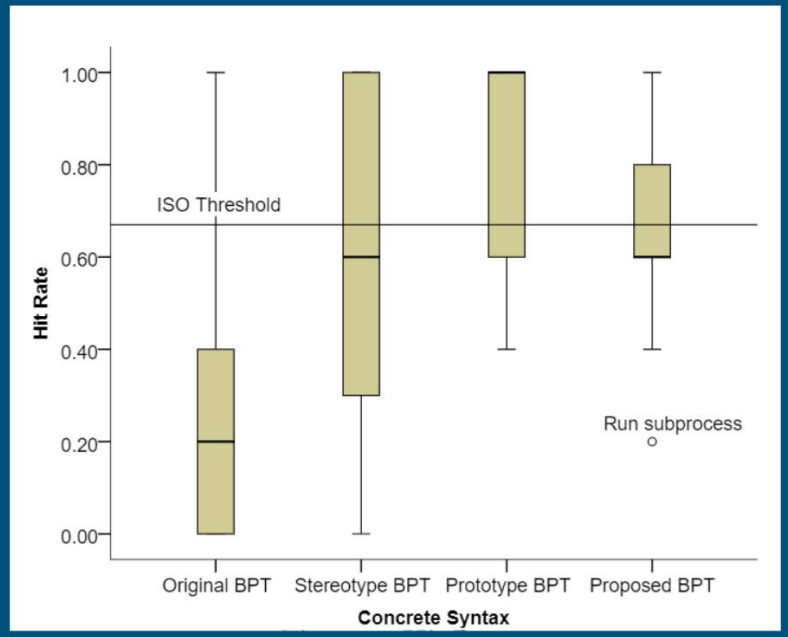
BPT - Evaluation Process



Semantic Transparency and Comparison

Table 1: Hit rate

Construct	Original	Stereotype	Prototype	Proposed
Wait for timeout	**1.00**	.60	**1.00**	**.80**
Run subprocess	.40	.20	.60	.20
Start process	.40	**1.00**	**.80**	.60
Send email	**.80**	**1.00**	**1.00**	**1.00**
Wait for DB event	.20	**1.00**	**1.00**	**.80**
End process	.40	.20	**1.00**	.60
Decision	.40	.0	.60	.60
Custom logic	.0	.0	.60	.40
Alternative flow	.0	.40	.40	.60
Fork	.0	**1.00**	.40	**1.00**
Join	.0	**.80**	.40	**1.00**
Terminate	.20	**1.00**	**1.00**	**.80**
Wait for user	.60	.60	**1.00**	**.80**
Wait for API call	.20	.60	**1.00**	.60
Broadcast DB event	.0	**.80**	**1.00**	.60
Mean Hit Rate	.31	.61	**.79**	**.69**
Standard Deviation	.31	.37	.26	.23



Thank you!

Exercise: answer to the following questions:

1) Who are the users of your DSL?
2) What is the context of use?
   a) When is it supposed to be used, and how?
   b) Is the language for internal use or external?
   c) What do the users want to express?
3) User Stories? (with user role, goal and acceptance criteria)
4) How would you plan your Empirical study?
5) What are the independent and dependent variables?
6) What metrics would you use?