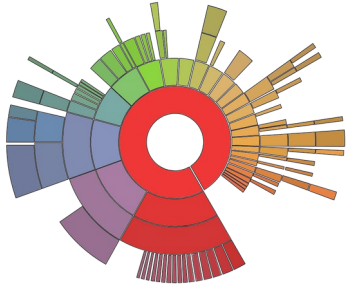
The Inria/IRISA DiverSE Research Group

July, 2025
GDR-IHM -- GT GSI

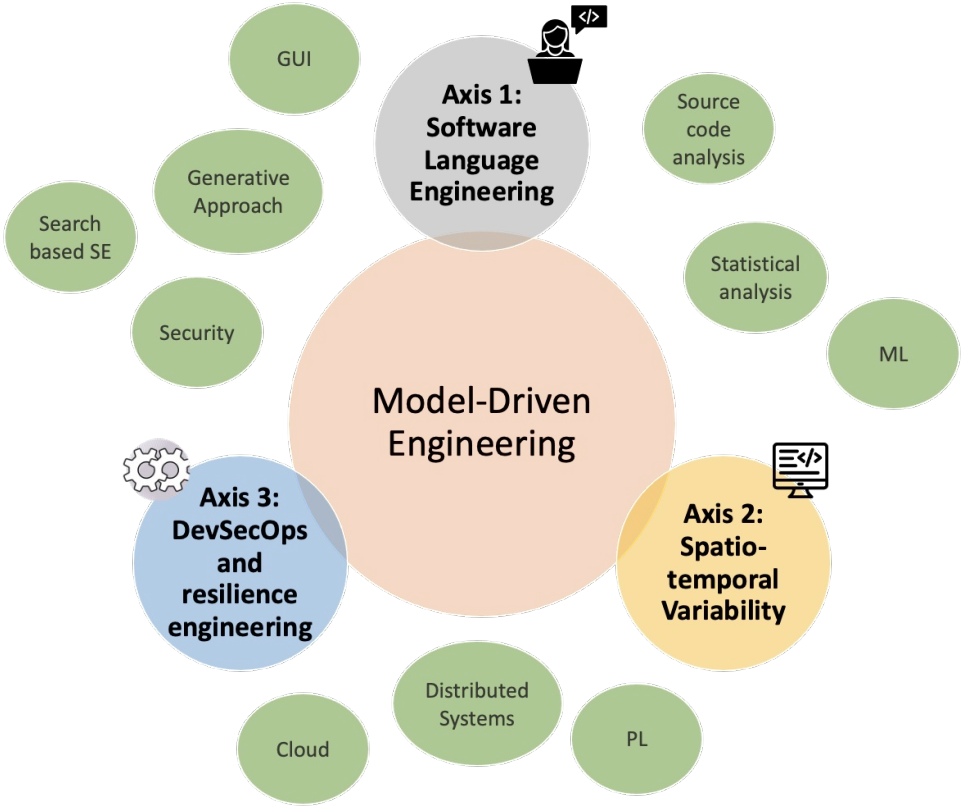
http://www.diverse-team.fr

THE DIVERSE GROUP



- Inria/IRISA project-team in Software Engineering
  - **Head**: Olivier Barais (Univ Rennes), **Co-head**: Benoit Combemale (Univ Rennes / Inria)
  - Strong background in Model-Driven software/systems Eng.
  - Software languages, architecture, evolution, simulation, variability and testing
  - Applied to smart, heterogeneous, and distributed CPS (e.g., IoT, Industry 4.0), scientific computing, cloud-native applications, etc.
  - 12 Prof. and Inria/CNRS researchers, 1 Inria RSE, ~20 PhD, 2 post-doc, 4 SE
- Deductive and empirical scientific approaches
- Open-source software development

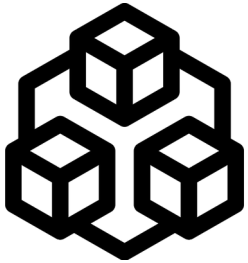
Research axes (2021-2025)



Axis 1: Software Language Engineering

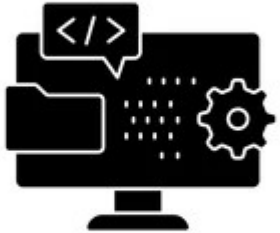
DSL Executability

- Modularity, composability
- MoCC, REPL
- Animation, debugging, monitoring, (co-)simulation



Modular & Distributed IDE

- Language protocol
- Microservice architecture
- IDE as Code
- Reconfigurable IDE



Design Lab

- **Web-based IDE**
- **Usability & interactivity**
- **Collaborative platforms**
- **Design-Space Exploration**

Self-Adaptable Language

- Integrate the domain feedback loop in language definition
- Approx. language semantics
- Smart Language/IDE

Live & Polyglot Development

- **Immediate feedback and direct manipulation**
- Program (state) co-evolution
- **Socio-technical coordination**

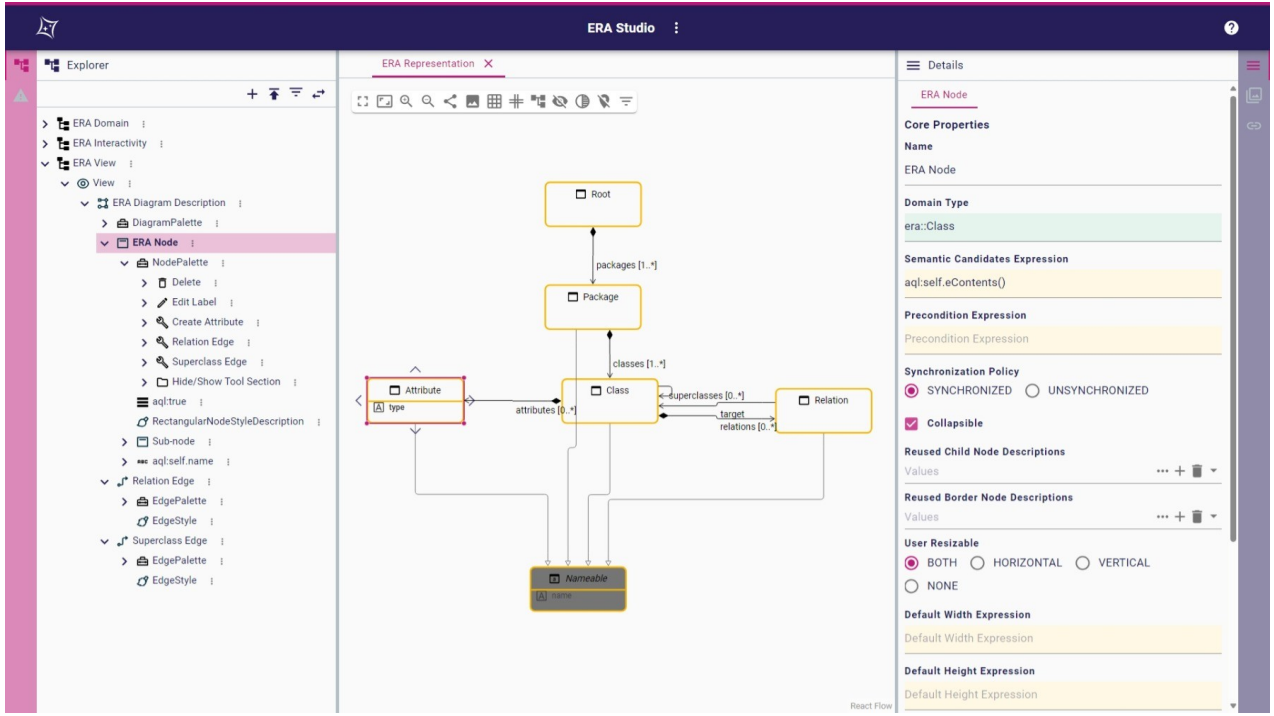
Engineering interactive systems in DiverSE

Engineering interactive systems in DiverSE

New approaches to build development environments (IDE)

New foundations to build interactive system

Approaches to build IDEs



Sirius Web language workbench — Design of the concrete and abstract language of a Class-Relationship-Attribute (CRA) language

Approaches to build IDEs

- Several studies show that the usability is a key point of graphical modeling workbenches [1-3]
  - But such usability is costly to develop…
  - So, we lack of usability in modeling workbenches

[1] Omar Badreddin et al. 2018. A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice
[2] John Hutchinson et al. 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure [3] Charlotte Verbruggen et al. 2023. Practitioners' experiences with model-driven engineering: a meta-review

Approaches to build IDEs

- Interactive features: solution to the lack of usability

[4] Arnaud Blouin et al. 2015. Assessing the Use of Slicing-based Visualizing Techniques on the Understanding of Large Metamodels
[5] Roberto Rodriguez-Echeverria et al. 2018. Towards a Language Server Protocol Infrastructure for Graphical Modeling
[6] Alexander Egyed et al. 2008. Generating and evaluating choices for fixing in consistencies in UML design models

Approaches to build IDEs

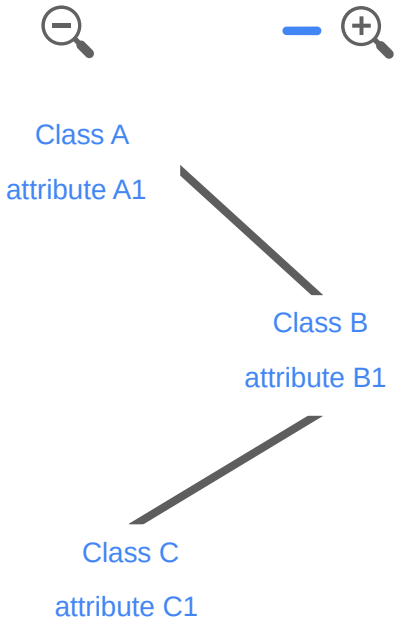
- Interactive features: solution to the lack of usability

[4] Arnaud Blouin et al. 2015. Assessing the Use of Slicing-based Visualizing Techniques on the Understanding of Large Metamodels
[5] Roberto Rodriguez-Echeverria et al. 2018. Towards a Language Server Protocol Infrastructure for Graphical Modeling
[6] Alexander Egyed et al. 2008. Generating and evaluating choices for fixing in consistencies in UML design models

Approaches to build IDEs

- Interactive features: solution to the lack of usability

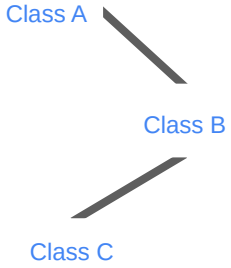
  Example: Semantic zooming



Approaches to build IDEs

- Interactive features: solution to the lack of usability

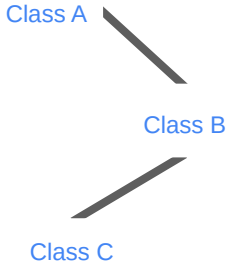
  Example: Semantic zooming



Approaches to build IDEs

- Interactive features: solution to the lack of usability

  Example: Semantic zooming

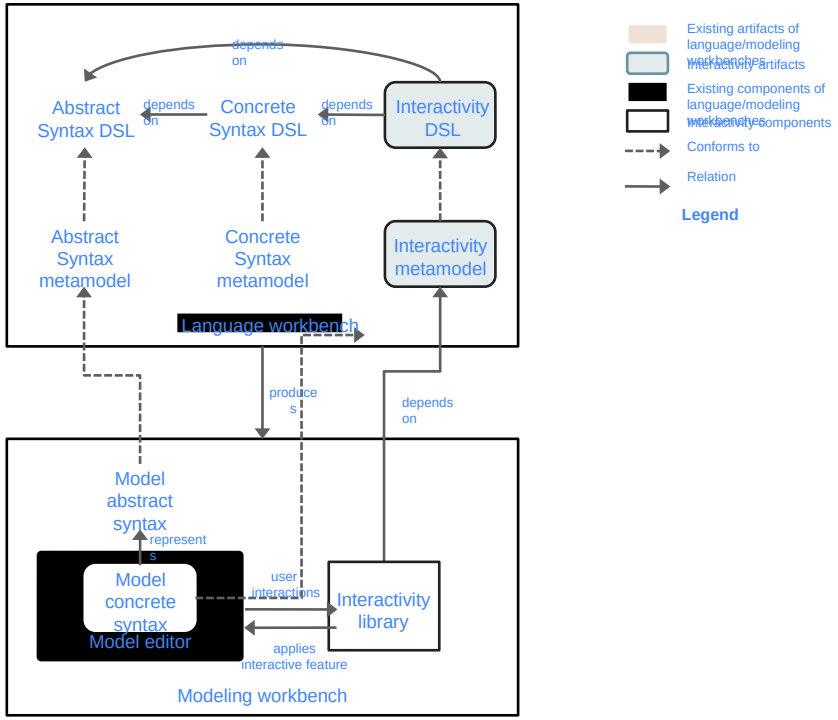


[4] Arnaud Blouin et al. 2015. Assessing the Use of Slicing-based Visualizing Techniques on the Understanding of Large Metamodels

- Some of them are
  - So… We have to define them for each DSL (costly)

Approaches to build IDEs

- New meta-DSL: **Interactivity DSL**, to describe semantic-aware interactive feature
- Aware of the other concerns of the developed DSL
- Integrated inside language workbenches
- Includes an **interactivity library** that depends on its technology stack and uses its API

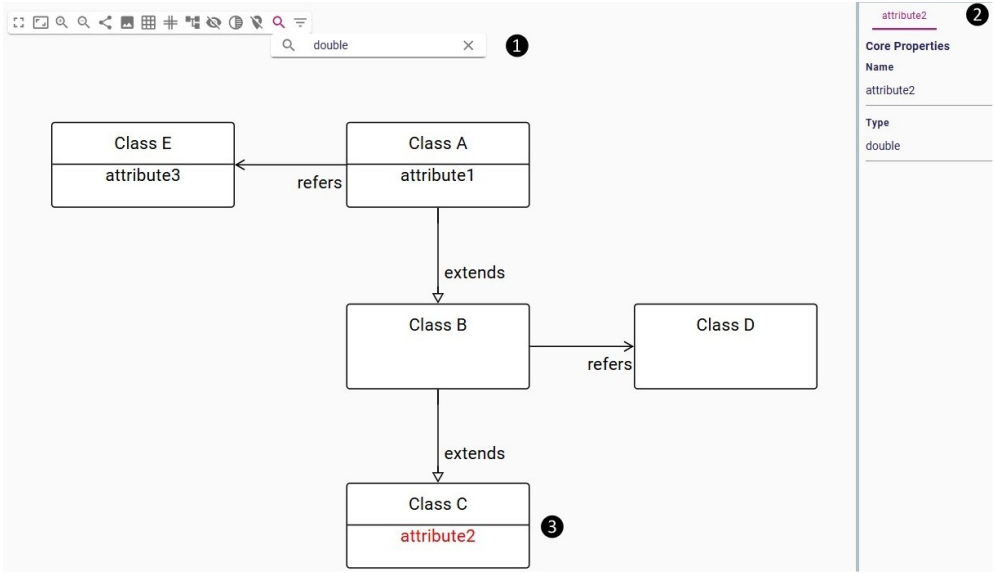


Approaches to build IDEs

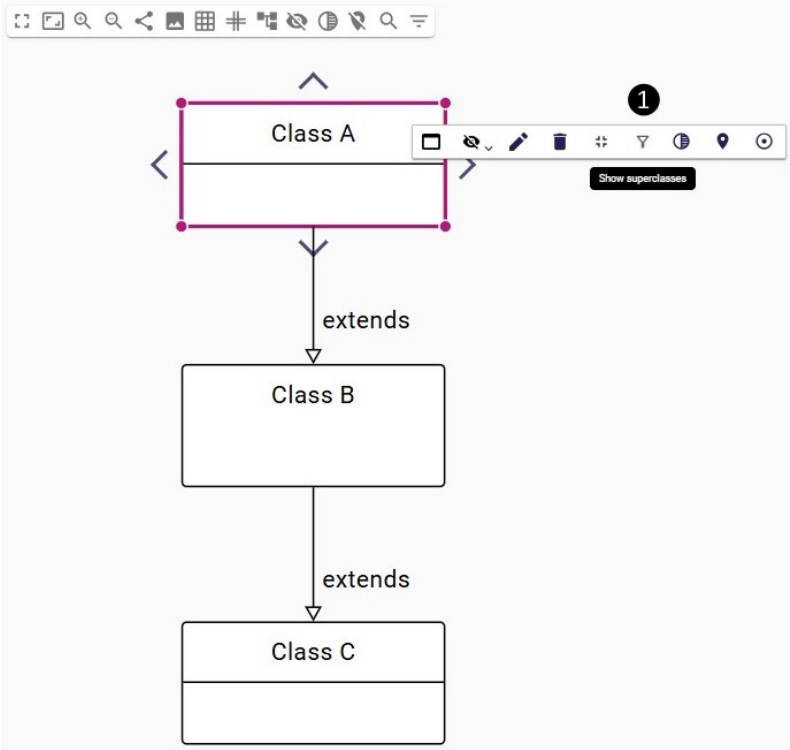
```
import abstract-syntax ‘uml-as.ecore' as umlas
import concrete-syntax ‘uml-cs.ecore' as umlcs
```

Approaches to build IDEs

```
import abstract-syntax ‘uml-as.ecore' as umlas
import concrete-syntax ‘uml-cs.ecore' as umlcs
search: [umlas.packages.classes.name, umlas.packages.name,
```



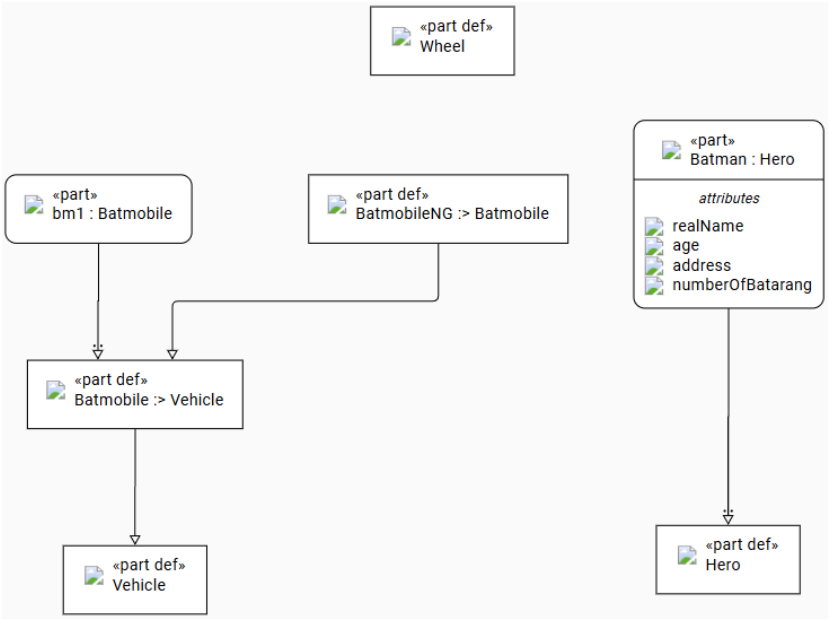
Approaches to build IDEs




```
import abstract-syntax ‘uml-as.ecore' as umlas
import concrete-syntax ‘uml-cs.ecore' as umlcs
search: [umlas.packages.classes.name, umlas.packages.name,
‘*’]
dynamic-filter:
    name: inheritance
    focus: umlas.packages.classes
    radius: [1..*]
    filter: show-inheritance
```

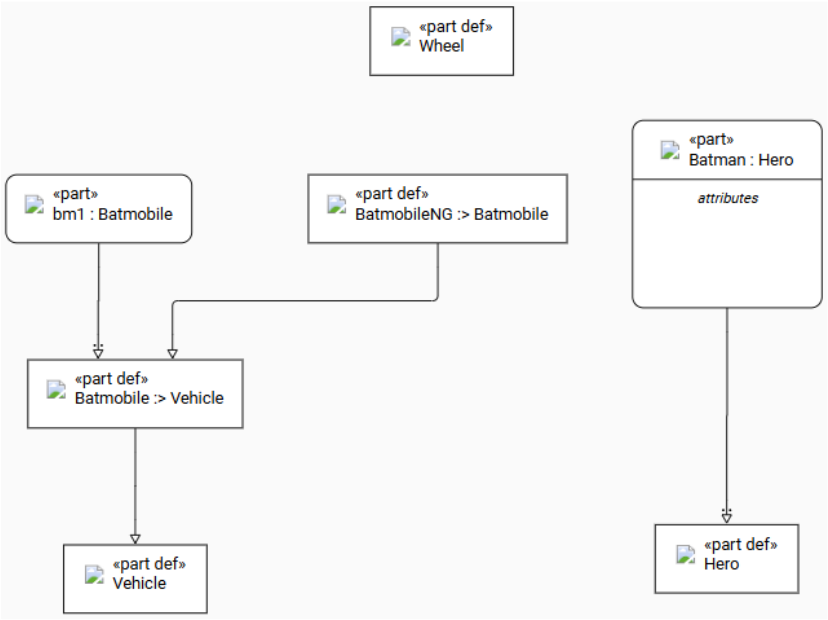
```
filters:
    show-inheritance:
        show: umlas.packages.classes
        show: umlas.package.class.superclasses
```

Approaches to build IDEs




```
import abstract-syntax ‘uml-as.ecore' as umlas
import concrete-syntax ‘uml-cs.ecore' as umlcs
search: [umlas.packages.classes.name, umlas.packages.name,
‘*’]
dynamic-filter:
    name: inheritance
    focus: umlas.packages.classes
    radius: [1..*]
    filter: show-inheritance
semantic-zoom:
    [0%-75%[:
        filter: without-attributes
    [150%-200%[:
        filter: without-packages
filters:
    show-inheritance:
        show: umlas.packages.classes
        show: umlas.package.class.superclasses
    without-attributes:
        hide: umlas.packages.classes.attributes
        setstyle umlas.packages.classes.name:
            font-size 200%
    without-package:
        hide: umlas.packages
```

Approaches to build IDEs




```
import abstract-syntax ‘uml-as.ecore' as umlas
import concrete-syntax ‘uml-cs.ecore' as umlcs
search: [umlas.packages.classes.name, umlas.packages.name,
‘*’]
dynamic-filter:
    name: inheritance
    focus: umlas.packages.classes
    radius: [1..*]
    filter: show-inheritance
semantic-zoom:
    [0%-75%[:
        filter: without-attributes
    [150%-200%[:
        filter: without-packages
filters:
    show-inheritance:
        show: umlas.packages.classes
        show: umlas.package.class.superclasses
    without-attributes:
        hide: umlas.packages.classes.attributes
        setstyle umlas.packages.classes.name:
            font-size 200%
    without-package:
        hide: umlas.packages
```

Engineering interactive systems in DiverSE

New approaches to build development environments (IDE)

New foundations to build interactive system

New foundations to build interactive system

- Validation & Verification / Security
  - What types of **faults** affect GUI?

Lelli, Valéria et al. "Classifying and qualifying GUI defects. ICST 2015

Fault categories	ID	Faults
GUI Structure and Aesthetics	GSA1	Incorrect layout of widgets (*e.g.* alignment, dimension, orientation, depth)
	GSA2	Incorrect state of widgets (*e.g.* visible, activated, selected, focused, modal, editable, expandable)
	GSA3	Incorrect appearance of widgets (*e.g.* font, color, icon, label)
Data Presentation	DT1	Incorrect data rendering (*e.g.* scaling factors, rotating, converting)
	DT2	Incorrect data properties (*e.g.* selectable, focused)
	DT3	Incorrect data type or format (*e.g.* degree *vs* radian, float *vs* double)
Interaction Behavior	IB1	Incorrect behavior of a user interaction
Action	ACT1	Incorrect action results
	ACT2	No action executed
	ACT1	Incorrect action executed
Reversibility	RVSB1	Incorrect results of undo or redo operations
	RVSB2	Reverting the current interaction in progress works incorrectly
	RVSB3	Reverting the current action in progress works incorrectly
Feedback	FDBK1	Feedback provided by widgets to reflect the current state of an action in progress works incorrectly.
	FDBK2	The temporary feedback provided all along the execution of long interactions is incorrect.

New foundations to build interactive system

- Validation & Verification / Security
  - What types of **attacks** affect GUI?

Cavalli and al. SoK: "Web Front-end Security", under review at S&P

Category	Attack/Vulnerabilities/ Issues	Description	Victim	Front-end Responsibility	Rationale	References
Front-end language specific vulnerabilities	Prototype pollution	Common ancestors of objects are modified by user-provided input, thus overwriting intended values	User/ Business owner	Very high	JavaScript features involved. Dangerous payload provided client-side	[11], [12], [13]
	DOM clobbering	Naming collision between JavaScript variables and named HTML markups (with `id` and `name` attribute)	User/ Business owner	Very high	HTML and JavaScript features involved. Dangerous payload provided client-side	[14]
Visual deception-based attacks	Clickjacking	User interactions (clicks, key presses, drag events, etc) are intercepted to manipulate invisible content	User	Very high	User interface, pointer and client events are exploited	[15], [16], [17] [18], [19], [20] [21]
	Tabnabbing	Top-level navigation (URL address) is replaced from tab A to tab B, or vice versa (reverse tabnabbing) tricking the user into providing credentials	User	Very high	User interface and browser tabs are exploited	[22], [23]
	Phishing	User is tricked into providing sensitive data through visual deception techniques	User	Very high	User interface is exploited	[24], [25], [26]

Etc.

New foundations to build interactive system

- Validation & Verification / Security
  - What types of **defects / smells** affect GUI?

Blouin et al. "User interface design smell: Automatic detection and refactoring of Blob listeners." IST 2018
Lelli et al. "Automatic detection of GUI design smells: The case of blob listener." EICS 2016

Blob listener (UI smell):
an event handler that manages 3 commands or more

```
class AController implements ActionListener {
  JButton b1;
  JButton b2;
  JMenuItem m3;

  @Override public void actionPerformed(ActionEvent e) {
     Object src = e.getSource();
     if(src==b1){
        // Command 1
     }else if(src==b2)
        // Command 2
     }else if(src instanceof AbstractButton &&
         ((AbstractButton)src).getActionCommand().equals(
         m3.getActionCommand()))
        // Command 3
     }
}}
```

New foundations to build interactive system

- Programming user interactions

  Blouin and Jézéquel. "Interacto: A Modern User Interaction Processing Model." TSE 2021
  Blouin. "A Type System for Flexible User Interactions Handling." EICS 2024

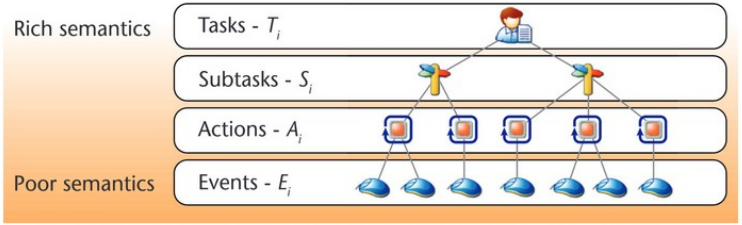
Lacks separation of concerns and reuse

- Processing UI events mixes:
  - User interaction definition
  - UI command production
- User interaction and command not first-class concerns

```
let isDragLocked = false;
const mvCallback = evt => {
  node.attr({ x: evt.x, y: evt.y });
};
node.addEventListener('dblclick', evt => {
  if (evt.button === 0) {
    if (isDragLocked) {
      node.style.cursor = '';
      node.removeEventListener('mousemove', mvCallback);
    } else {
      node.style.cursor = 'hand';
      node.addEventListener('mousemove', mvCallback);
    }
      isDragLocked = !isDragLocked;
}});
```

Lacks advanced features for developers

- No tangible undo/redo mechanism
- No native logging mechanism
- Manual optimisation (eg. throttling)

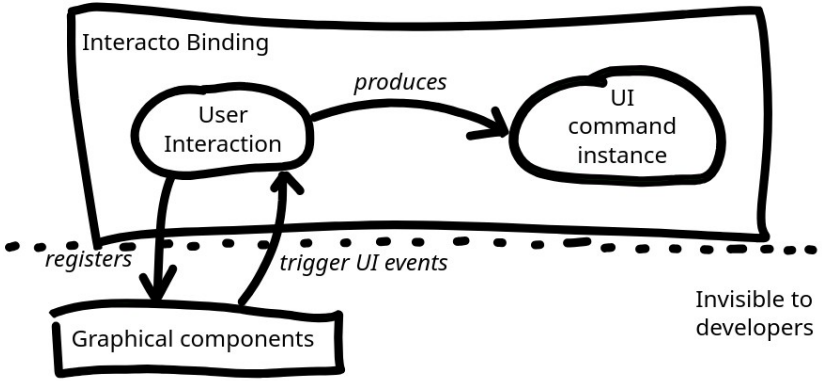


Xu, et al. "Analytic provenance for sensemaking: A research agenda." IEEE computer graphics and applications 35.3 (2015): 56-64

New foundations to build interactive system

- Programming user interactions

Blouin and Jézéquel. "Interacto: A Modern User Interaction Processing Model." TSE 2021
Blouin. "A Type System for Flexible User Interactions Handling." EICS 2024




```
interactoCtx
  // Use the drag-lock user interaction
  .dragLockBinder()
  // On each child of 'canvas', dynamically
  .onDynamic(this.canvas)
  // To produce and execute 'Translate' command instances
  .toProduce(i => new Translate(i.src.target, this.canvas))
  // When the interaction starts, change the cursor
  .first((_, i) => i.src.target.style.cursor = 'pointer')
  // When the interaction stops/cancelled, set the default cursor
  .endOrCancel(i => i.src.target.style.cursor = 'default')
  // When the interaction updates, update the translation vector
  // the ongoing command will use
  .then((c, i) => {
    c.vectorX = i.diffClientX;
    c.vectorY = i.diffClientY;
  })
  // That, only if the user uses the primary mouse button
  .when(i => i.tgt.button === 0)
  .continuousExecution()
  .bind();
```

- Developers configure an Interacto binding
  - Focuses on transforming a user interaction execution into a UI command instance
- Produced undoable commands automatically stored in dedicated histories
  - Various undo/redo algorithms supported
- No more UI events

New foundations to build interactive system

- Programming user interactions

Blouin and Jézéquel. "Interacto: A Modern User Interaction Processing Model." TSE 2021
Blouin. "A Type System for Flexible User Interactions Handling." EICS 2024

Future work

- Programming/modeling styles and interfaces
  - Live programming
  - Exploratoring programming
- Forges for socio-technical synchronization
- IDEs for augmented development
- Digital Twins
- UI V&V (e.g. XR)


Thank you!
DiverSE
https://www.diverse-team.fr