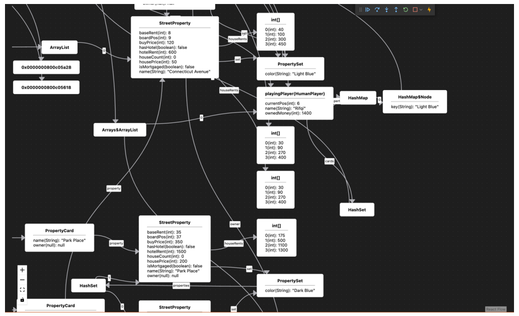
A Domain-Specific Language for Customizing Visual Debugger Views

Rifqi Adlan Apriyadi   Hidehiko Masuhara   Youyou Cong
Tokyo Institute of Technology

Motivation



Visual debuggers use object diagrams to visualize the runtime state.

Issues:

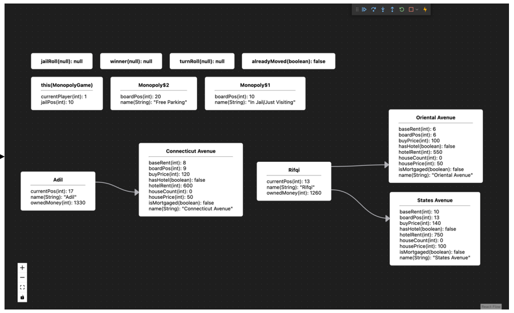
x **Visual Clutter** from too many nodes/edges[1]

x **Representation Gap** in visualization from the difference between a concept and its implementation[2]

Goal

To empower users with customizability
→ Get a <u>more focused</u> view

```
//...Other customizations...
c:Property {
  if (isNull f:owner) omit nodeOf here;
  else add newEdge (nodeOf f:owner)
    (nodeOf here);
}
```



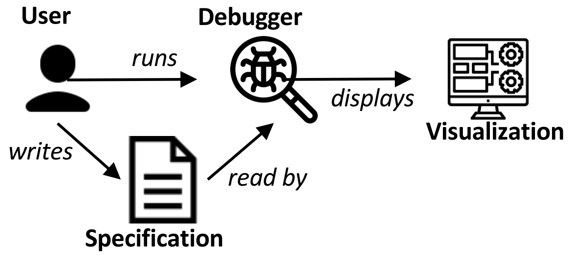
Approach

Customization

Customize the existence of nodes/edges or the contents therein of the visualized diagram based on runtime state via a **<u>Specification Language</u>**.

✓ **Visual Clutter**: Omit unnecessary information
✓ **Representation Gap**: Close the gap to resemble the concept on paper

<u>Usage</u>



<u>Concept: Location</u>

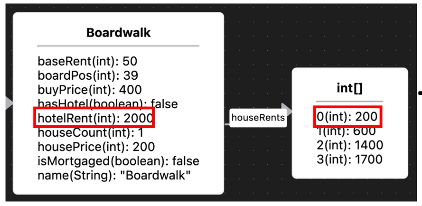
Classes
Fields
Methods
Local

```
c:Property {
  //...
  f:owner {//...}
  m:setMortgaged(boolean) {
    l:status {//...}
  }
}
```

All Property objects
All objects that is owner in Property
When halt in setMortgaged(boolean)
status local variable in method

x Problem 1: Static Customization

Cannot customize based on value
e.g.: Cannot display current rent:



Boardwalk
baseRent(int): 50
boardPos(int): 39
buyPrice(int): 400
hasHotel(boolean): false
hotelRent(int): 2000
houseCount(int): 1
housePrice(int): 200
isMortgaged(boolean): false
name(String): "Boardwalk"

houseRents

int[]
0(int): 200
1(int): 600
2(int): 1400
3(int): 1700

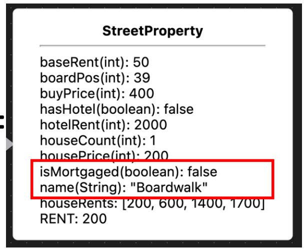
+

```
// Omit houseRents node and show rent
c:StreetProperty {
  setImmutable f:houseRents;
  num[] houseRents = valueOf f:houseRents;
  houseRents.insert(0, valueOf f:baseRent);
  num houseCount = valueOf f:houseCount;
  (nodeOf here).addRow
    ("RENT: " + houseRents[houseCount]);
}
```

=

✓ Solution 1: Contextual Customization

Retrieve values of runtime variables



StreetProperty
baseRent(int): 50
boardPos(int): 39
buyPrice(int): 400
hasHotel(boolean): false
hotelRent(int): 2000
houseCount(int): 1
housePrice(int): 200
isMortgaged(boolean): false
name(String): "Boardwalk"
houseRents: [200, 600, 1400, 1700]
RENT: 200

<u>Practicality in Debugging</u>

<u>Bug Localization</u>:
Use **simple high-level** customization functions

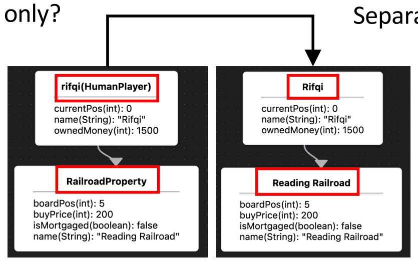
<u>Cause Identification</u>:
Use **detailed lower-level** customizations

<u>Solution Implementation</u>:
Use **very detailed** customizations

x Problem 2: Single-Point Entry

Specifications in respect to the main class only?

```
c:MonopolyGame {
  for (p : f:players)
    (nodeOf p).setTitle(f:name string);
  for (p : f:properties)
    (nodeOf p).setTitle(f:name string);
}
```



rifqi(HumanPlayer)
currentPos(int): 0
name(String): "Rifqi"
ownedMoney(int): 1500

RailroadProperty
boardPos(int): 5
buyPrice(int): 200
isMortgaged(boolean): false
name(String): "Reading Railroad"

Rifqi
currentPos(int): 0
name(String): "Rifqi"
ownedMoney(int): 1500

Reading Railroad
boardPos(int): 5
buyPrice(int): 200
isMortgaged(boolean): false
name(String): "Reading Railroad"

✓ Solution 2: Modularity

Separation of concerns in customization.

```
Node[] nodes = [];
c:Player {nodes.append(node);}
c:Property {nodes.append(node);}
// Do something extra with nodes
```

Challenges and Future Work

- Streamlined specification
  → Complex object value retrieval
- Specification reusability
  → Location Polymorphism

References

1. Lowering Visual Clutter in Large Component Diagrams (IV'12)
2. Possible Improvements in UML Behavior Diagrams (ITOEC'17)