Livelits: Filling Typed Holes with Live GUIs

PROGRESS REPORT – TYDE 2019

Cyrus Omar	**University of Chicago → Michigan**
Nick Collins	University of Chicago
David Moon	University of Colorado Boulder → Michigan
Ian Voysey	Carnegie Mellon University
Ravi Chugh	University of Chicago

Text-like user interfaces are **often** great.

```
1 type 'a lazy_node =
2   | Empty
3   | Node of 'a * 'a lazy_list
4 and 'a lazy_list = 'a lazy_node lazy_t
5
6 let empty = lazy Empty
7
8 let con x zl = lazy (Node (x,zl))
9 let decon zl =
10   match force zl with
11     | Empty -> None
12     | Node (x,ztl) -> Some (x, ztl)
13
14 let rec to_list zl =
15   match decon zl with
16     | None -> []
17     | Some (x, ztl) -> x::to_list ztl
18
19 let rec zmap ~f zl = lazy (
20   match force zl with
21     | Empty -> Empty
22     | Node (x, ztl) -> Node (f x, zmap f ztl))
23
24 (* Below illustrates the interesting point of zmap and lazy_list *)
25 let f x = print_endline "f";x * 10
26 let g x = print_endline "g";string_of_int x
27 let l1 = [3;2;1]
28 let zl1 = con 1 empty |> con 2 |> con 3
29
30 (*
31   using normal map and normal list, if we want to map f, g to the list `map
   g (map f l1)`, then we get
32   f
33   f
34   f
35   g
36   g
37   g
38
39   because each map will finish a function mapping on the list, then next fun
-:---  lazy_list.ml   Top (26,43)   Git-master   (Tuareg Abbrev)
```

```
15   a.(q-s).(p) <- a.(q).(q-s);
16   a.(q).(q-s) <- a.(s).(q);
17   a.(s).(q) <- tmp
18
19 let rotate a =
20   let n = Array.length a in
21   let rec rot p q =
22     if p >= q then ()
23     else (
24       for s = p to q-1 do
25         unit_op a p s q;
26       done;
27       rot (p+1) (q-1)
28     )
29   in
30   rot 0 (n-1);
31   a
32
33 let a1 =
34   [|
35     [|1;2;3;4|];
36     [|1;2;3;4|];
37     [|1;2;3;4|];
38     [|1;2;3;4|];
39   |]
-:**-  rotate.ml      29% (27,21)   Git-master   (Tuareg Abbrev)
```

```
1 Welcome to utop version 1.15 (using OCaml version 4.02.0)!
2
3 Findlib has been successfully loaded. Additional directives:
4   #require "package";;      to load a package
5   #list;;                   to list the available packages
6   #camlp4o;;                to load camlp4 (standard syntax)
7   #camlp4r;;                to load camlp4 (revised syntax)
8   #predicates "p,q,...";;   to set these predicates
9   Topfind.reset();;         to force that packages will be reloaded
10  #thread;;                 to enable threads
11
12 utop[0]>
U:**-  *utop*         All (12,9)     (utop: idle)
```

Text-like user interfaces are **often** great.

```
let qsort : List(Num) → List(Num) =
  λxs.
    case xs
    | [] ⇒ []
    | y::ys ⇒
      let (smaller, bigger) = partition 49 46 in
      let (qs, qb) = 53 in
      append qs (y::qb)
    end
in

qsort (4::2::6::5::3::1::7::[])
```

Text-like user interfaces are **often, but not always,** great.

```
let bgcolor : Color =
```

Text-like user interfaces are **often, but not always,** great.

```
let bgcolor : Color = RGBA(0, 148, 55, 255)
```

What if we could **fill holes** of types like these **by manipulating GUIs**?

```
let bgcolor : Color =
```

Graphite

[Omar et al., ICSE 2012]

Active Code Completion

Cyrus Omar, YoungSeok Yoon, Thomas D. LaToza, Brad A. Myers
Carnegie Mellon University, Pittsburgh, PA, USA
{comar,youngseok,tlatoza,bam}@cs.cmu.edu

***Abstract*—Code completion menus have replaced standalone API browsers for most developers because they are more tightly integrated into the development workflow. Refinements to the code completion menu that incorporate additional sources of information have similarly been shown to be valuable, even relative to standalone counterparts offering similar functionality. In this paper, we describe *active code completion*, an architecture that allows library developers to introduce interactive and highly-specialized code generation interfaces, called *palettes*, directly into the editor. Using several empirical methods, we examine the contexts in which such a system could be useful, describe the design constraints governing the system architecture as well as particular code completion interfaces, and design one such system, named Graphite, for the Eclipse Java development environment. Using Graphite, we implement a palette for writing regular expressions as our**

For example, users of the Calcite tool completed 40% more tasks in a lab study (unfortunately, a Jadeite control group was not included.)

In all of these systems, the code completion interface has remained primarily menu-based. When an item is selected, code is inserted immediately, without further input from the developer. These systems are also difficult to extend: a fixed strategy determines the completions that are available, so library providers cannot directly specify new domain-specific or contextually-relevant logic. In this paper we propose a technique called *active code completion* that eliminates these restrictions[1]. This makes developing and integrating a broad array of highly-specialized developer tools directly

Graphite: Color Example

[Omar et al., ICSE 2012]

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color =
8      }
9
10 }
11
```

Show interactive palette - Color
color : Color
args : String []
main(String[] args) : void - ColorTest
ColorTest

Create a color using a color palette.

Press '^Space' to show Template Proposals

Press 'Tab' from proposal table or click for focus

Outline
import declarations
ColorTest
main(String[]) : void

Graphite: Color Example

[Omar et al., ICSE 2012]

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color =
8      }
9
10 }
11
```

RGB
R: 0
G: 148
B: 255
Hex: 0094FF
HSV
H: 205
S: 100
V: 100
Transparency - Alpha
255

Outline
import declarations
ColorTest
main(String[]) : voic

Graphite: Color Example

[Omar et al., ICSE 2012]

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color = new Color(0, 148, 255, 255);
8      }
9
10 }
11
```

Outline

import declarations
ColorTest
main(String[]) : void

Graphite: Regex Example

[Omar et al., ICSE 2012]

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^\d+(\.\d+)?\s?(F|C)$

☐ Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Graphite: Regex Example

[Omar et al., ICSE 2012]

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should match:
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NOT match:
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Design Methodology

[Omar et al., ICSE 2012]

- Large online developer survey (**~450 participants**)
  - Quantitative and qualitative feedback about mockups
  - Solicitation of use cases

Design Methodology

[Omar et al., ICSE 2012]

- Large online developer survey (**~450 participants**)
  - Quantitative and qualitative feedback about mockups
  - Solicitation of use cases

- Implementation

Design Methodology

[Omar et al., ICSE 2012]

- Large online developer survey (**~450 participants**)
  - Quantitative and qualitative feedback about mockups
  - Solicitation of use cases

- Implementation

- Small pilot study (n=7, regex palette)

Design Methodology

[Omar et al., ICSE 2012]

- Large online developer survey (**~450 participants**)
  - Quantitative and qualitative feedback about mockups
  - Solicitation of use cases
- Implementation
- Small pilot study (n=7, regex palette)

Usefulness

[Omar et al., ICSE 2012]

"Consider situations where you need to instantiate the [specified] class. What portion of the time, in these situations, do you think you would use this feature?"

CLASS	Nearly every time	Most of the time	Some of the time	Rarely	Never
Color	9.6%	22.1%	**32.4%**	28.2%	7.7%
RegExp	**36.6%**	29.5%	21.8%	7.3%	4.8%
SQL	18.2%	19.3%	**30.9%**	20.4%	11.4%

Suggested Use Cases

[Omar et al., ICSE 2012]

containers (dictionary, matrix)
URLs, paths with lookup
editors for embedded languages (e.g. HTML)
audio transformations
3D transformations
number / string / date formatting previews
GUI widgets
fonts
shapes
GUI layouts
shortcut keys
custom documentation

Suggested Use Cases

[Omar et al., ICSE 2012]

```
@GraphitePalette(url="…")
class MyClass { … }
```

containers (dictionary, matrix)
URLs, paths with lookup
editors for embedded languages (e.g. HTML)
audio transformations
3D transformations
number / string / date formatting previews
GUI widgets
fonts
shapes
GUI layouts
shortcut keys
custom documentation

Summary

	Palettes (Graphite)
Data establishing usefulness	✓
Wide variety of use cases	✓
Extensible	✓

Summary

	Palettes (Graphite)
Data establishing usefulness	✓
Wide variety of use cases	✓
Extensible	✓
Persistent	✗
Compositional	✗
Live	✗

Summary

	Palettes (Graphite)	**Livelits** (Hazel)
Data establishing usefulness	✓	✓
Wide variety of use cases	✓	✓
Extensible	✓	✓
Persistent	✗	✓
Compositional	✗	✓
Live	✗	✓

Persistence

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^\d+(\.\d+)?\s?(F|C)$

Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Persistence

```java
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should match:
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NOT match:
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Persistence

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should ma
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NO
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Use the regular expression workbench...
Pattern – java.util.regex
p : Pattern
s : String
isTemperature(String s) : boolean – Matc
Matcher – edu.cmu.cs.comar
runnable – runnable

Press '^Space' to show Template Proposals

Displays a workbench that allows you to enter a regular expression pattern and test it against positive and negative examples. Automatically handles escape sequences!

Press 'Tab' from proposal table or click for focus

Persistence

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should ma
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NO
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

^-?(\d+|(\d*(\.\d+)))?\s?(F|C)$

☐ Ignore Case

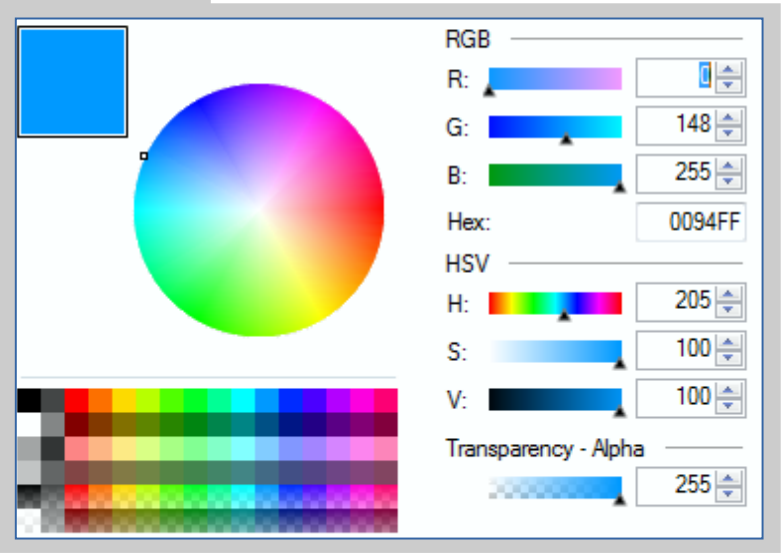
Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Persistence

```
let bgcolor : Color = $color
```



Livelit Definitions

```
livelit $color at Color {
  type model = …
  type action = …
  val init_model = …
  val update = (model, action) => …
  val view = (model) => …
  val expand = (model) => …
}
```

Related: **Typed Literal Macros** [Omar and Aldrich, ICFP 2018]

```
$html `( <div>
           <h3>Chemical Structure of Sucrose</h3>
           <$>$smiles `({mono_glucose}-O-{mono_fructose})` |> Smiles.to_svg</$>
         </div> )`
```

Available for Reason (Facebook's OCaml front-end)

Compositionality

```
let grades : List(StudentRecord) = $grade_table
```

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	91	_
"Bob"	91	74	88	97	_

Compositionality

```
let grades : List(StudentRecord) = $grade_table
  = 91 * curve
```

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	91 * curve	_
"Bob"	91	74	88	97	_

Compositionality

```
let grades : List(StudentRecord) = $grade_table
  = $slider(0, 100)
```

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	91 * curve	80
"Bob"	91	74	88	97	_

Liveness

```
let grades : List(StudentRecord) = $grade_table
```

= 91 * curve

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	95	80
"Bob"	91	74	88	97	_

Liveness

(based on [Omar et al., POPL 2019])

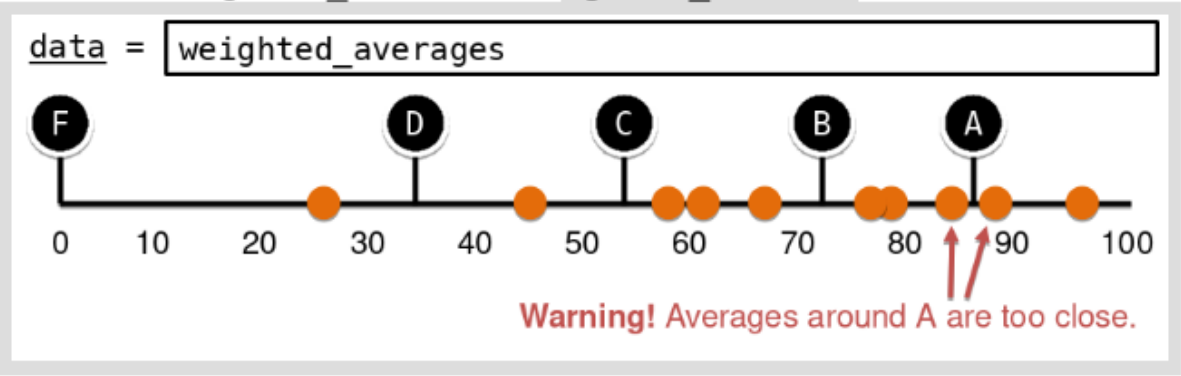
```
let grades : List(StudentRecord) = $grade_table
```

```
= 91 * curve
```

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	95	80
"Bob"	91	74	88	97	_

Ongoing Work: Live Palettes in Hazel

```
let cutoffs: grade_cutoffs = $grade_cutoffs;
```



Summary

	Palettes (Graphite)	**Livelits** (Hazel)
Data establishing usefulness	✓	✓
Wide variety of use cases	✓	✓
Extensible	✓	✓
Persistent	✗	✓
Compositional	✗	✓
Live	✗	✓

In Progress

Theory (Agda)

Implementation

Layout Variations

Case Studies

Livelits for Authoring

Thanks!

hazel.org

```
let grades : List(StudentRecord) = $grade_table
```

= 91 * curve

name	hw1	hw2	hw3	midterm	final
"Alice"	88	77	94	95	80
"Bob"	91	74	88	97	_

- **Live Evaluation**
- **Direct Manipulation**

- **Type-Driven Feedback**
- **Type-Driven Automation**