Livells: 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.
Text-like user interfaces are often great.

```ocaml
let qsort : List(Num) → List(Num) =
  λxs.
    case xs
    | [] → []
    | y::ys →
      let (smaller, bigger) = partition in
      let (qs, qb) = 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.

```plaintext
let bgcolor : Color = |
```
Text-like user interfaces are often, but not always, great.

```javascript
let bgcolor : Color = RGBA(0, 148, 55, 255)
```
What if we could **fill holes** of types like these **by manipulating** GUls?

```plaintext
let bgcolor : Color = |
```
Active Code Completion

Cyrus Omar, YoungSeok Yoon, Thomas D. LaToza, Brad A. Myers
Carnegie Mellon University, Pittsburgh, PA, USA
{cmar, youngseok, latoza, bradm}@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 Calceite 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. This makes developing and integrating a broad array of highly-specialized developer tools directly
import java.awt.Color;

public class ColorTest {
    public static void main(String[] args) {
        Color color = 
    }

    Create a color using a color palette.

    Press ‘Space’ to show Template Proposals
Graphite: Color Example

import java.awt.Color;

public class ColorTest {
    public static void main(String[] args) {
        Color color =
    }
}
import java.awt.Color;

public class ColorTest {
    public static void main(String[] args) {
        Color color = new Color(0, 148, 255, 255);
    }
}
Graphite: Regex Example

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

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = ^d+(\.d+)?\s?\(F|C\)$
        Should match... | Should NOT match...
        | 12:05  | 37  |
        | 37F  | 37  |
        | 42.1 F  | 37Q  |
        | .8C  | -10C  |
        = matched by pattern
```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^[-]?([\d+]([\d]*(\.[\d]+)))?[\d]*(F|C)\$");
        /*
        * Should match:
        *   37F
        *   42.1 F
        *   .8C
        *   -10C
        *
        * Should NOT match:
        *   12:05
        *   37
        *   37Q
        */
    }
}
• Large online developer survey (~450 participants)
  - Quantitative and qualitative feedback about mockups
  - Solicitation of use cases
Design Methodology

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

- Implementation
Design Methodology

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

- Implementation

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

[Omar et al., ICSE 2012]
Design Methodology

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

- Implementation

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

<table>
<thead>
<tr>
<th>CLASS</th>
<th>Nearly every time</th>
<th>Most of the time</th>
<th>Some of the time</th>
<th>Rarely</th>
<th>Never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>9.6%</td>
<td>22.1%</td>
<td><strong>32.4%</strong></td>
<td>28.2%</td>
<td>7.7%</td>
</tr>
<tr>
<td>RegExp</td>
<td><strong>36.6%</strong></td>
<td>29.5%</td>
<td>21.8%</td>
<td>7.3%</td>
<td>4.8%</td>
</tr>
<tr>
<td>SQL</td>
<td>18.2%</td>
<td>19.3%</td>
<td><strong>30.9%</strong></td>
<td>20.4%</td>
<td>11.4%</td>
</tr>
</tbody>
</table>
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]

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

```java
@GraphitePalette(url="...")
class MyClass { ... }
```
Summary

Palettes
(Graphite)

Data establishing usefulness ✓
Wide variety of use cases ✓ ✓ ✓
Extensible ✓
<table>
<thead>
<tr>
<th>Feature</th>
<th>✔</th>
<th>✓</th>
<th>✗</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data establishing usefulness</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Wide variety of use cases</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Extensible</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Persistent</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Compositional</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Live</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>

Palettes (Graphite)
## Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Palettes (Graphite)</th>
<th>Livelits (Hazel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data establishing usefulness</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wide variety of use cases</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extensible</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Persistent</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Compositional</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Live</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = \n
        // Should match...
        Should match... | Should NOT match...
        37F            | 12:05
        42.1 F        | 37
        .8C            | 37Q
        -10C          |

        = matched by pattern

        Pattern Description
        ^\d+(-\d+)?\s?(F|C)$ Must match at the beginning of the line
        [\w\d] Must match at the end of the line
        [a-z] Set definition, matches the letter a or b or c
        [a-z][0-9] Set definition, matches a or b or c followed by 0 or 9
        [^a-z] 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|y Finds X or Y
        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\f]
import java.util.regex.Pattern;

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

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]$\);

        return p.matcher(s).matches();
    }
}
```
let bgcolor : Color = $color
livelit $color at Color {
  type model = ...
  type action = ...
  val init_model = ...
  val update = (model, action) => ...
  val view = (model) => ...
  val expand = (model) => ...
}
$html `( <div>
    <h3>Chemical Structure of Sucrose</h3>
    <$> $smiles `({mono_glucose}-0-{mono_fructose})`  |> Smiles.to_svg</$>
)</div>)`
let grades : List(StudentRecord) = $grade_table

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>91</td>
<td>_</td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td>_</td>
</tr>
</tbody>
</table>
let grades : List(StudentRecord) = $grade_table

= 91 * curve

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>91 * curve</td>
<td></td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
let grades : List(StudentRecord) = $grade_table

= $slider(0, 100)

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>91 * curve</td>
<td>80</td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
let grades : List(StudentRecord) = $grade_table

= 91 * curve

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>95</td>
<td>80</td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>
let grades : List(StudentRecord) = $grade_table

= 91 * curve

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td>80</td>
</tr>
</tbody>
</table>

(based on [Omar et al., POPL 2019])
Ongoing Work: Live Palettes in Hazel

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

data = weighted_averages
```

Warning! Averages around A are too close.
## Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Palettes (Graphite)</th>
<th>Livelits (Hazel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data establishing usefulness</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wide variety of use cases</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extensible</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Persistent</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Compositional</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Live</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
In Progress

Theory (Agda)

Implementation

Layout Variations

Case Studies

Livelits for Authoring
let grades : List(StudentRecord) = $grade_table

= 91 * curve

<table>
<thead>
<tr>
<th>name</th>
<th>hw1</th>
<th>hw2</th>
<th>hw3</th>
<th>midterm</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Alice&quot;</td>
<td>88</td>
<td>77</td>
<td>94</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>&quot;Bob&quot;</td>
<td>91</td>
<td>74</td>
<td>88</td>
<td>97</td>
<td>80</td>
</tr>
</tbody>
</table>
- Live Evaluation
- Direct Manipulation

- Type-Driven Feedback
- Type-Driven Automation

Next Generation Environments

Type-Theoretic Foundations