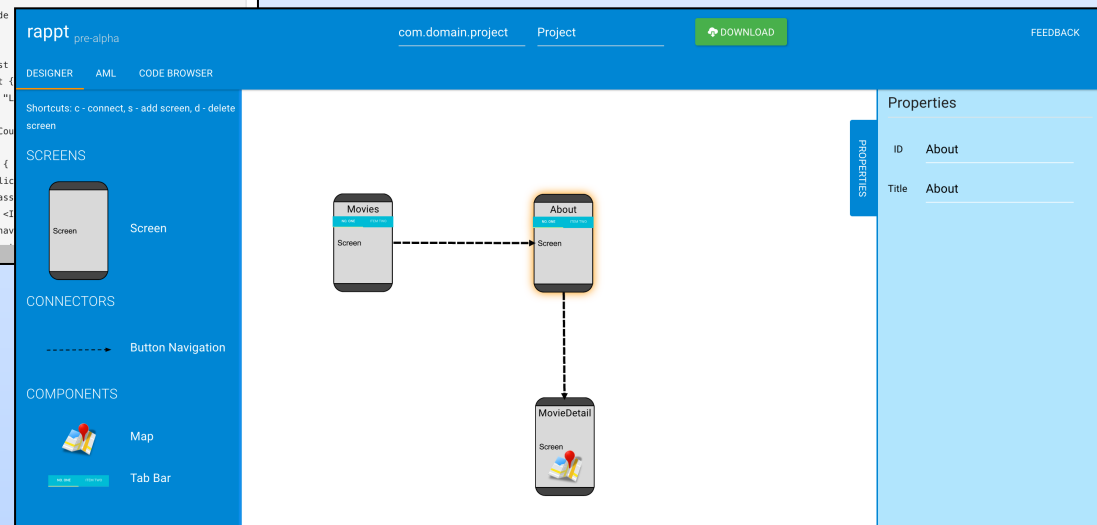


Supporting Multi-View Development for Mobile Applications

Scott Barnett, Iman Avazpour, Rajesh Vasa, John Grundy

A multi-view code synthesis tool for developing data-driven mobile apps at decreasing levels of abstraction.

```
1 app {
2   landing-page Main
3 }
4
5 api RestCountries "http://restcountries.eu/rest/v1" {
6   GET all "/" {list}
7   GET CountryCode "/alpha/{code}"
8 }
9
10 screen Main "List Example" {
11   group mainList {
12     label msgId "List of countries"
13     on-load {
14       call RestCountries.all
15     }
16     list listId {
17       on-item-click {
18         // To pass value to details screen add
19         // pass <ID> <field value in JSON response>
20         // to 'navigate-to' command
21         navigate-to CountryDetail pass code alpha2Code
22       }
23     }
24     row rowId {
25       label nameLabel name
26       label capitalLabelId capital
27     }
28   }
29 }
30
31 screen CountryDetail "Country" {
32   group mainScreen {
33     on-load {
34       // 'code' must be passed from another screen
35       call RestCountries.CountryCode passed code
36     }
37     label nameLabel "Name:"
38     label restNameId name
39     label capitalLabel "Capital City:"
40     label restCapitalId capital
41     label regionLabel "Region:"
42     label restRegionId region
43     label subRegionLabel "Sub Region:"
44     label restSubRegionId subregion
45   }
46 }
47 }
```



Problem

Simple data-driven mobile applications take too long to develop!



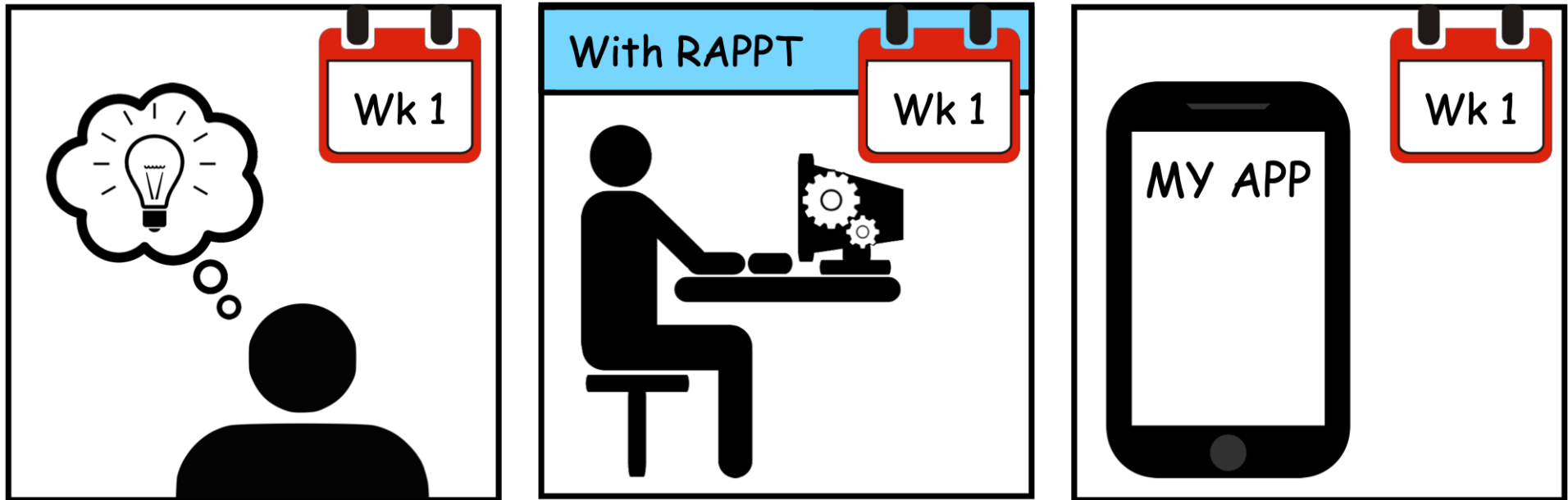
Problem continued



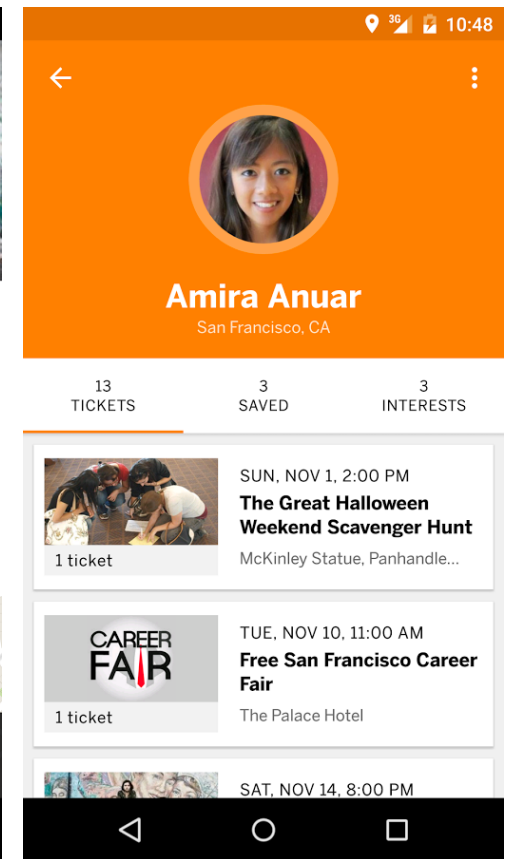
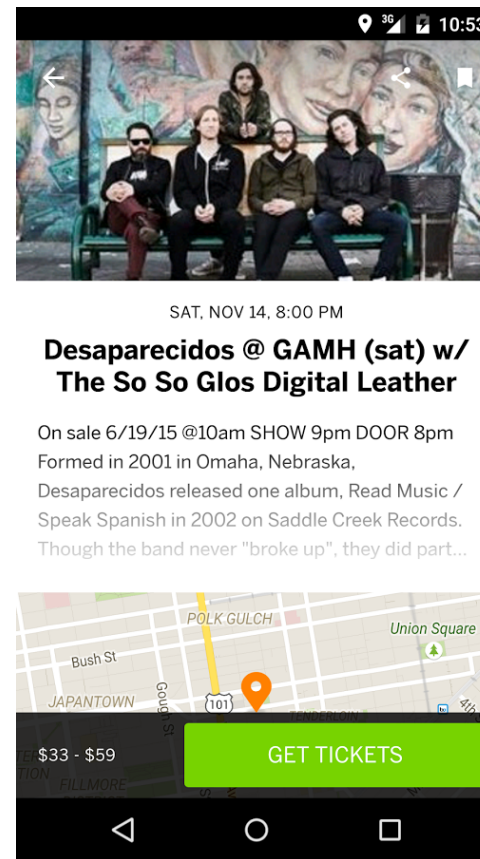
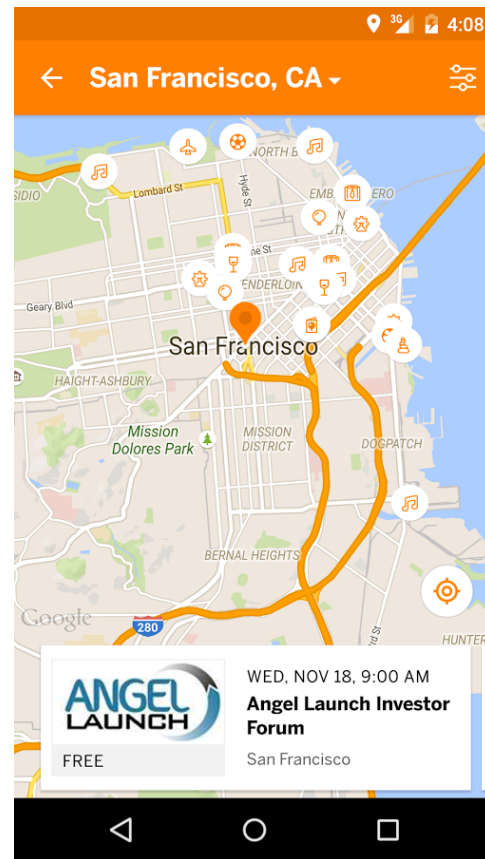
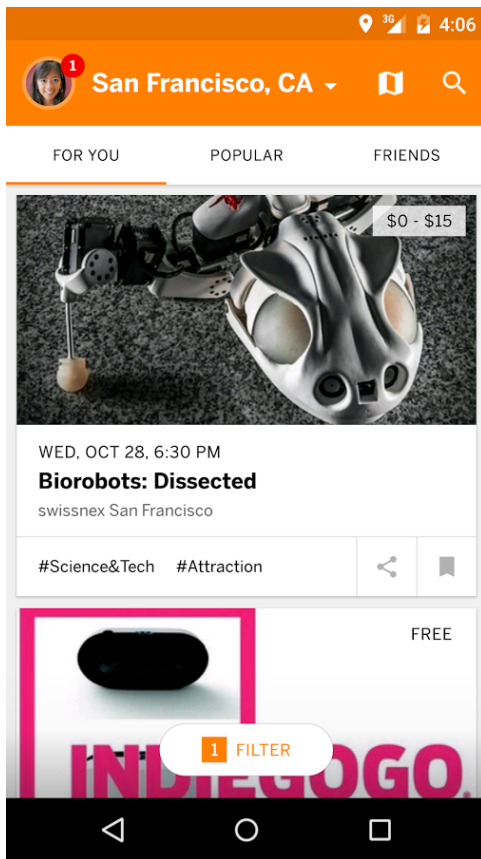
- Multiple levels of abstraction required for designing and building a mobile app (navigation, workflow, services etc.)
- Data-drive apps contain boilerplate code
- Re-development needed to migrate from a prototype to a production ready app
- Low level code implementation needed to deal with device specific variations

Solution

Better domain-specific tooling == faster development time
Target end users == professional app developers

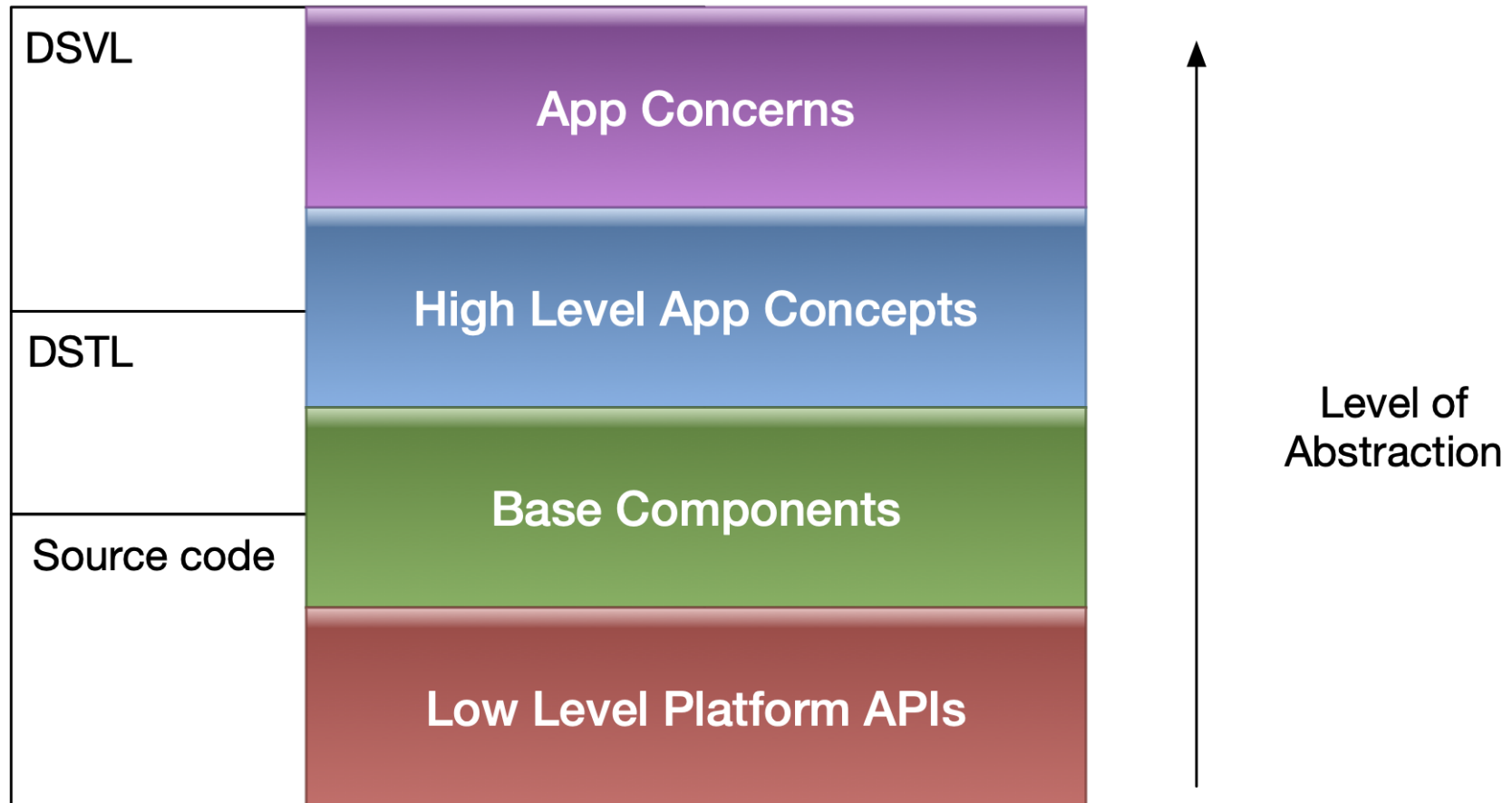


Goal: Generate 80% of this app!



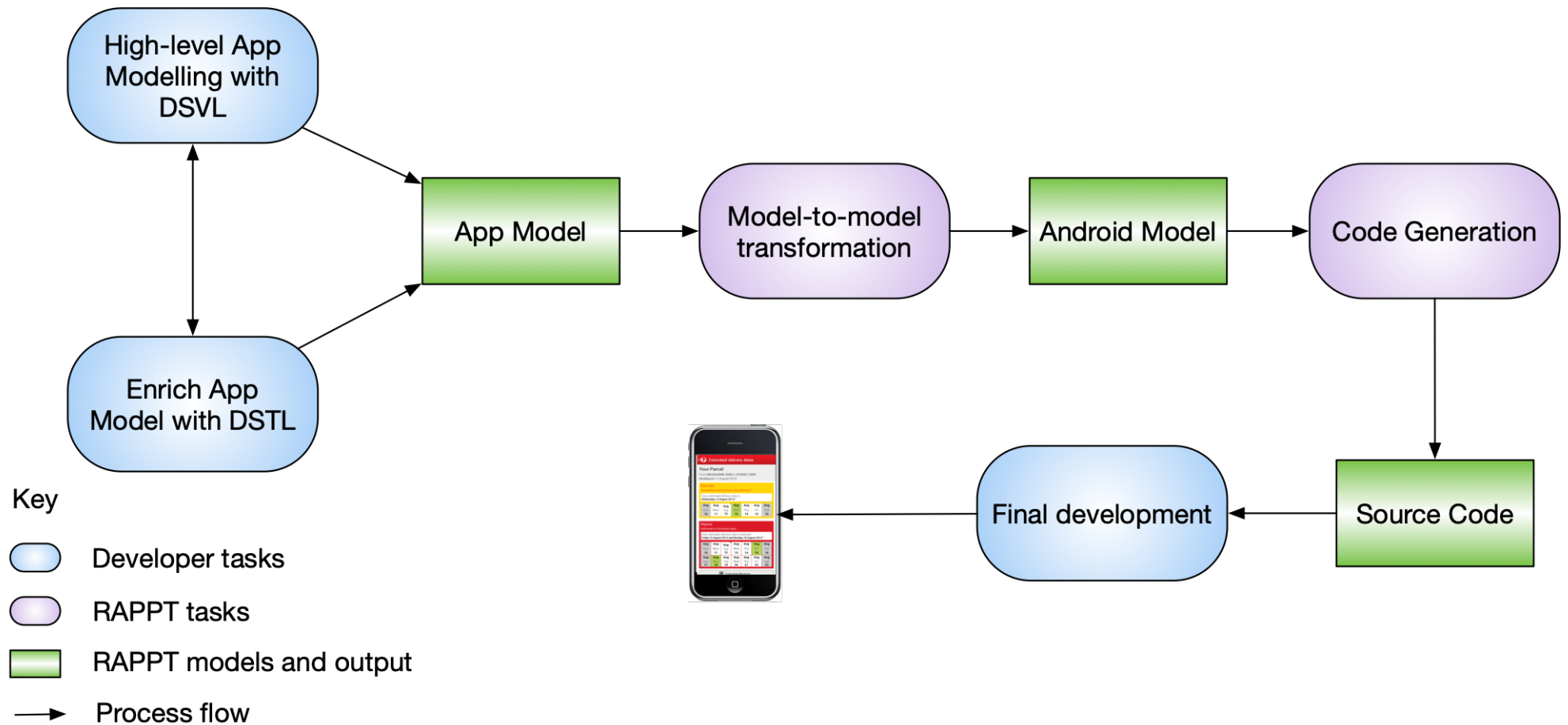
<https://play.google.com/store/apps/details?id=com.eventbrite.attendee&hl=en>

Levels of abstraction:



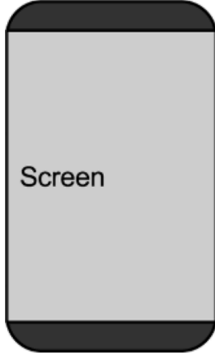
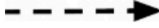


DSVL: Domain Specific Visual Language
DSTL: Domain Specific Textual Language

Our approach



Domain Specific Visual Language

Example visual elements that make up RAPPT's Visual Language.

Concept	Notation	Description
Screen		Represents a screen displayed on a mobile device as seen by the end user.
Button navigation		Represents navigation from one screen to another by clicking on the UI component Button.
Map		Displays a Google Map ⁶
Tabbar		Represents the Mobile navigation UI pattern, Tabbar.

RAPPT's DSVL Interface

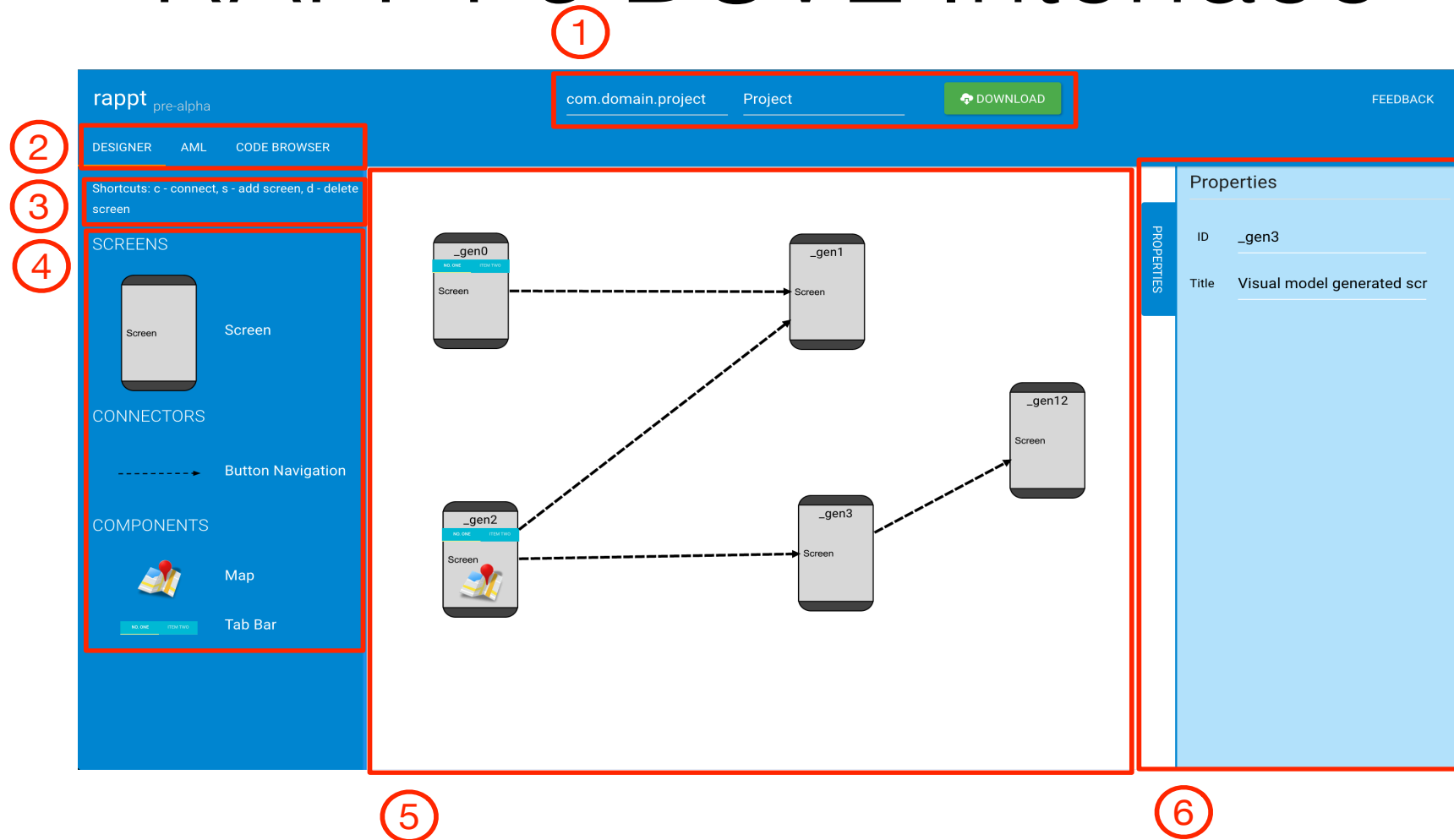


Figure C.1: Screenshot for RAPPT's Designer. 1. Package, Project and Download, 2. Navigational Tabs, 3. Shortcuts, 4. Widget Pane, 5. Visual Editing Pane and 6. Properties Pane.

Domain Specific Textual Language

1. Event handler for screen loading
2. Call to RESTful API
3. Landing page for the app
4. RESTful API definition

```
screen MovieDetailScreen "Movie detail" {
  group movieDetailGroup {
    ① on-load {
      ② call MovieDB.movieDetail passed idParam
    }
    image backdropId backdrop_path:image
    label title2ID title
    image posterImageId poster_path:image
    label overViewId overview
    label popularityLabel "Popularity:"
    label restPopularity popularity
  }
}

app { ③
  landing-page MoviesScreen
}

api MovieDB "https://api.themoviedb.org/3" {
  ④ api-key api_key "<API KEY>"
  GET popularMovies "/movie/popular"
  GET movieDetail "/movie/{id}"
}
```

RAPPT Generated Code

```
app {
  landingPage = MoviesScreen
  tabbar navDrawerId {
    tab movieScreenTab "Movies" navigate-to MoviesScreen
    tab aboutTab "About" navigate-to AboutScreen
  }
}

api MovieDB {
  base = "https://api.themoviedb.org/3"
  auth(type : token) {
    apiKey = "API KEY GOES HERE"
    tokenParam = "api_key"
  }
  endPoints {
    popularMovies = GET (endPoint:"/movie/popular")
    movieDetail = GET (endPoint:"/movie/{id}")
  }
}

screen MoviesScreen {
  title = "Movies"
  model {
    dataSource = source(endPoint: MovieDB.popularMovies) {
      loadedOriginalLabelId <= original_title
      imgId <= poster_path:image
      listId <= results:list
    }
  }
  view {
    group moviesGroup {
      list listId {
        on-item-click {
          navigate-to MovieDetailScreen(param: id)
        }
        row rowId {
          loadedOriginalLabelId = label( binding: dataSource)
          imgId = image( binding: dataSource)
        }
      }
    }
  }
}

screen MovieDetailScreen(string param) {
  title = "Movie detail"
  model {
    dataSource = source (endPoint: MovieDB.movieDetail) {
      param => id
      backdropId <= backdrop_path:image
      overViewId <= overview
      homePageId <= homepage
      restPopularity <= popularity
    }
  }
  view {
    group movieDetailGroup {
      backdropId = image( binding: dataSource)
      overViewId = label( binding: dataSource)
      homePageId = label( binding: dataSource)
      restPopularity = label( binding: dataSource)
    }
  }
}

screen AboutScreen {
  title = "About"
  view {
    group aboutGroup {
      developerId = label(text: "Developed by Scott")
      copyrightId = label(text: "Copyright 2014")
    }
  }
}
```

- DSVL Generated Code
- Samples Code
- Manual edited code

Figure C.4: Complete source code for the MovieDB app showing code generated by the DSVL, code reused from copy-pasting AML samples and manually edited DSTL code.

Results from an evaluation with 20 users

- 95% of users felt RAPPT was beneficial for mobile app development (agree or strongly agree)
- 80% felt RAPPT was more efficient than starting with a standard Android project (agree or strongly agree)
- More abstractions required and additional support for avoiding errors in the user interface

Summary

- Building mobile apps is hard
- Professional app developers under-researched area for support
- They want range of abstractions from high to low level, “professional quality” generated templates/partial code ability to edit/polish generated code
- We use mixed DSVL/DSTLs to support this