EasyInterface: A Toolkit for Rapid Development of GUIs for Research Prototype Tools

Jesús Doménech[1], Samir Genaim[1](✉), Einar Broch Johnsen[2], and Rudolf Schlatte[2]

[1] Complutense University of Madrid, Madrid, Spain
genaim@gmail.com
[2] University of Oslo, Oslo, Norway

Abstract. EasyInterface is an open-source toolkit to develop web-based graphical user interfaces (GUIs) for research prototype tools. This toolkit enables researchers to make their tool prototypes available to the community and integrating them in a common environment, rapidly and without being familiar with web programming or GUI libraries.

1 Introduction

During the lifetime of a research project, research prototype tools are often developed which share many common aspects. For example, in the Envisage [2] project, we developed various tools for processing ABS programs: static analyzers, compilers, simulators, etc. Both as individual researchers and as groups, we often develop several related tools over time to pursue a specific line of research.

Providing the community with easy access to research prototype tools is crucial to promote the research, get feedback, and increase the tools' lifetime beyond the duration of a specific project. This can be achieved by building GUIs that facilitate trying tools; in particular, tools with *web-interfaces* can be tried without the overhead of first downloading and installing the tools.

In practice, we typically avoid developing GUIs until tools are fairly stable. Since prototype tools change continuously, in particular during a research project, they will often not be available to the community during early development. Both programming plug-ins for sophisticated frameworks such as Eclipse Scout and building simpler GUIs from scratch are tedious tasks, in particular for web-interfaces. It typically gets low priority when developing a research prototype. Often we opt for copying the GUI of one tool and modifying it to fit the needs of a new related tool. Apart from code duplication, these tools will "live" separately, although we might benefit from having them in a common GUI.

EasyInterface is a toolkit that aims at simplifying the process of building and maintaining GUIs for (but not limited to) research prototype tools.

This work was partially funded by the EU project FP7-ICT-610582 ENVISAGE: Engineering Virtualized Services, the Spanish MINECO projects TIN2012-38137 and TIN2015-69175-C4-2-R, and the CM project S2013/ICE-3006.

M. Huisman and J. Rubin (Eds.): FASE 2017, LNCS 10202, pp. 379–383, 2017.
DOI: 10.1007/978-3-662-54494-5_22

Avoiding complex programming, it provides an easy, declarative way to make existing (command-line) tools available via different environments such as a web-interface, within Eclipse, etc. It also defines a text-based output language that can be used to improve the way results are presented to the user without requiring any particular knowledge of GUI/Web programming; e.g., if the output of a tool is (a structured version of) "*highlight line number 10 of file ex.c*" and "*when the user clicks on line 10, open a dialog box with the text ...*", the web-interface will interpret this and convert it to corresponding visual effects. An advantage of using such an output language is that it will be understood by all the front-end environments of EasyInterface, e.g., the web-interface and the Eclipse plug-in (which is still under development). EasyInterface is open source and available at http://github.com/abstools/easyinterface. Detailed description of EasyInterface, including a step by step example on how to integrate tools and discussion of related work, is available in the user manual [1].

2 General Overview

The overall architecture of EasyInterface is depicted in Fig. 1. Its two main components are (*i*) *server side*: a machine with several tools (the circles `Tool1`, etc.) executable from the command-line, and with output going to the standard output. These are the tools that we want to make available for the outside world; and (*ii*) *client side*: several clients that communicate with the server to execute a tool. Tools may run on the server machine or on other machines; e.g., the web-interface can be installed as a web-page on the server, and accessed from anywhere with a web browser. Clients can connect to several servers simultaneously.

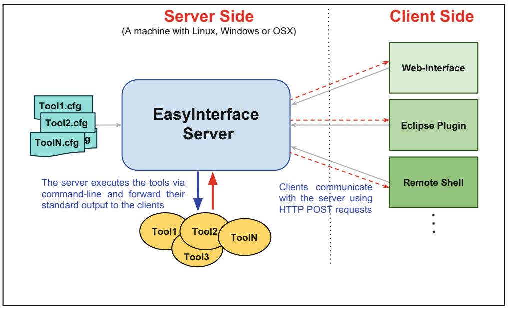

Server Side
(A machine with Linux, Windows or OSX)
Client Side
Tool1.cfg
Tool2.cfg
ToolN.cfg
EasyInterface Server
Web-Interface
Eclipse Plugin
Remote Shell
The server executes the tools via command-line and forward their standard output to the clients
Clients communicate with the server using HTTP POST requests
Tool1
Tool2
ToolN
Tool3

Fig. 1. EasyInterface architecture.

The server side addresses the problem of *providing a uniform way to remotely execute locally installed tools*. This problem is solved by the server, which consists of PHP programs (on top of an HTTP server). The server supports *declarative specifications* of how local tools can be executed and which parameters they take, using simple configuration files. For example, the XML snippet to the right is a configuration file for a tool

```
<app id="myapp" visible="true">
 ...
 <execinfo method="cmdline">
  <cmdlineapp>
    /path-to/myapp  _ei_parameters
  </cmdlineapp>
 </execinfo>
 <parameters prefix = "-" check="false">
  ...
  <selectone name="c">
   <option value="1" />
   <option value="2" />
  </selectone>
 </parameters>
</app>
```

called "myapp". The cmdlineapp tag is a template describing how to execute the tool from the command-line. The template parameter _ei_parameters is replaced by an appropriate value before execution. The server also supports template parameters for, e.g., passing files, temporal working directories, session identifiers, etc. The parameters tag includes a list of parameters accepted by the tool. For example, the parameter "c" above takes one of the values 1 or 2.

Once the configuration file is installed on the server, we can access the tool using an HTTP POST request that includes JSON-formatted data like the one on the right. When receiving such a request, the server generates a shell command according to the specification in the configuration file (e.g., "/path-to/myapp -c 1"), executes it and redirects the standard output to the client. The server also supports (*i*) tools that generate output in the background, we let clients fetch output (partially) when it is ready; and (*ii*) tools that generate files, we let clients download them later when needed. In all cases, the server can *restrict the resources* available to a tool (e.g., the processing time), and *guarantees the safety* of the generated command; i.e., clients cannot manipulate the server to execute other programs installed on the server. In addition to tools, the server can include example files, so users can easily try the tools.

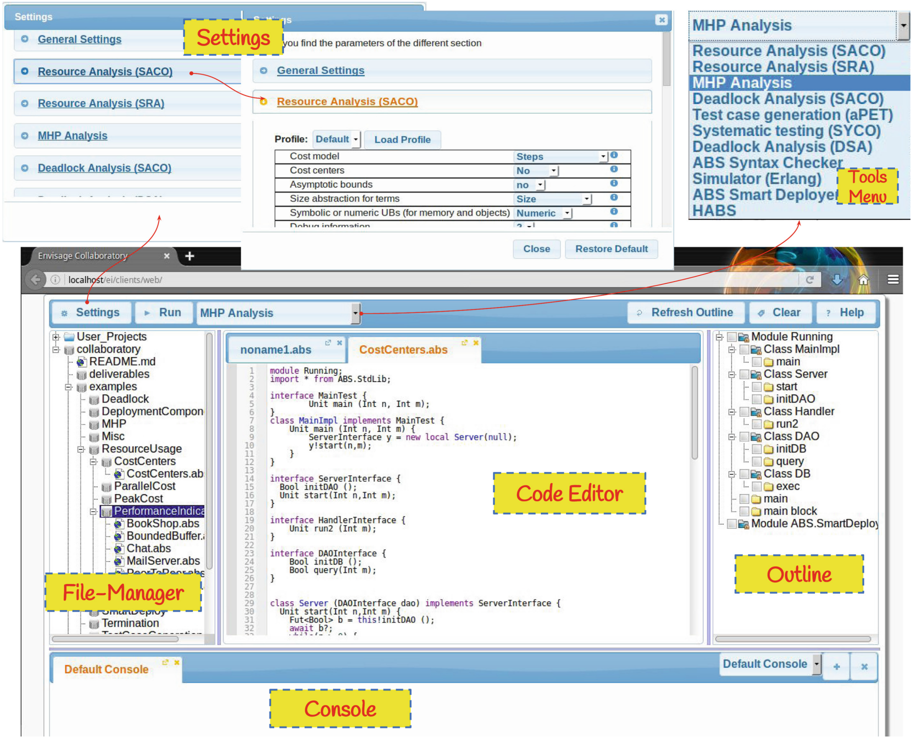
```
{
 command: "execute",
 app_id: "myapp",
 parameters: {
    c: ["1"],
    ...
 },
 ...
}
```

EasyInterface not only makes the server side execution of a tool easy, it provides client side GUIs that (1) connect to the server and ask for available tools; (2) let users select the tool to execute, set its parameters and provide a source program; (3) generate and send a request to the server; and (4) display the returned output. EasyInterface provides three generic clients: a *web-interface* similar to an IDE; an Eclipse IDE plug-in; and a remote command-line shell. The last two clients are under development, so we focus here on the web-interface.

The web-interface, shown in Fig. 2, is designed like an IDE where users can edit programs, etc. Next to the Run button there is a drop-down menu with all available tools obtained from the associated servers. In the settings window, the user can select values for the different parameters of each tool. These parameters are specified in the corresponding configuration files on the server side, and automatically converted to combo-boxes, etc., by the web-interface. When the user clicks the Run button, the web-interface sends a request to the associated server to execute the selected tool and prints the received output back in the console area of the web-interface.

Since the web-interface and Eclipse plug-in are GUI based clients, EasyInterface allows tools to generate output with some graphical effects, such as opening dialog-boxes, highlighting code lines, adding markers, etc.

```
<highlightlines dest="/path-to/sum.c">
  <lines> <line from="5" to="10"/> </lines>
</highlightlines>
...
<oncodelineclick dest="/path-to/sum.c" outclass="info">
  <lines><line from="17" /></lines>
  <eicommands>
    <dialogbox boxtitle="Hey!">
      <content format="text"> some message </content>
    </dialogbox>
  </eicommands>
</oncodelineclick>
```



Settings

you find the parameters of the different section

General Settings

Resource Analysis (SACO)

Profile: Default Load Profile

Cost model: Steps
Cost centers: No
Asymptotic bounds: no
Size abstraction for terms: Size
Symbolic or numeric UBs (for memory and objects): Numeric

Close Restore Default

```
module Running;
import * from ABS.StdLib;

interface MainTest {
        Unit main (Int n, Int m);
}
class MainImpl implements MainTest {
    Unit main (Int n, Int m) {
        ServerInterface y = new local Server(null);
        y!start(n,m);
    }
}

interface ServerInterface {
  Bool initDAO ();
  Unit start(Int n,Int m);
}

interface HandlerInterface {
    Unit run2 (Int m);
}

interface DAOInterface {
    Bool initDB ();
    Bool query(Int m);
}


class Server (DAOInterface dao) implements ServerInterface {
  Unit start(Int n,Int m) {
    Fut<Bool> b = this!initDAO ();
    await b?;
```

Fig. 2. EasyInterface web-interface client

To use this feature, tools must support the EasyInterface output language, shown in the XML snippet to the right. The tag highlightlines indicates that Lines 5–10 of file /path-to/sum.c should be highlighted. The tag oncodelineclick indicates that when clicking on Line 17, a dialog-box with a corresponding message should be opened. Note that a tool is only modified once to produce such output, with similar effect in all EasyInterface clients (including future ones).

3 Concluding Remarks

EasyInterface is a toolkit for the rapid development of GUIs for command-line research prototype tools. The toolkit has been successfully used in the Envisage project to integrate the tools from the different partners in a common web-based environment, including parsers, type-checkers, compilers, simulators, deadlock and worst-case cost analyzers, and a systematic testing framework (see http://abs-models.org). Our experience suggests that the methodology implied by EasyInterface for building GUIs is adequate for research prototype tools; as such tools change continuously, the corresponding GUIs can be modified immediately and with negligible effort. Future work includes plans to develop more clients, and libraries for different programming languages to facilitate generation of the output commands/actions instead of printing these directly.

References

1. Easyinterface User Manual. http://costa.ls.fi.upm.es/papers/costa/eiusermanual.pdf
2. Envisage: Engineering Virtualized Services. http://www.envisage-project.eu