Persistent Contextual Values as Inter-process Layers

Markus Raab
Vienna University of Technology
Institute of Computer Languages, Austria
Email: markus.raab@complang.tuwien.ac.at
Goals

- **Context-aware**
  e.g. battery status

- **Customizable**
  adapt to user

- **Mobile**
  consistent context changes across apps
  performance/battery life
Context-Oriented Programming

- originates from object-oriented programming
- layers represents context
- can be activated anywhere in the program
  - dynamic scope

```java
void rcvPhoneCall() {
    e.context().with(<PhoneCall>())({
        vibrate();
    });
}
```
Contextual Values

- “Trivial generalization of thread-local values”
- layers and dynamic scoping as in context-oriented programming
- access performance and usage identical to variables

```cpp
void visit(Person & p) {
    p.context().with<CountryAustriaLayer>()
        .with<LanguageGermanLayer>()([&] {
            cout << "visit " << ++p.visits
            << " in " << p.context
            << " : " << p.greeting
        });
    cout << p.greeting
}
```

“Hallo!”

“Griaß enk!”

Different Context, Same Thread
Program Execution Environment

- **consists of**: Configuration Files, Commandline Arguments, ...

- Program Execution Env. is defined using a specification

```
[/%language%/%country%/%dialect%/person/greeting]
type=String
```

```
[/%country%/person/visits]
type=Integer
default=0
```

```
[/%location%/country]
type=String
```

- `/:` denotes hierarchy of contextual values
- `%:` placeholders for layers

needed for **customization**

- initialize and persist every contextual value
Problem

- no synchronization between processes
- dependencies between activations
- implementation of layer tedious

```cpp
void rcvPhoneCall() {
    e.context().with()<PhoneCall>()([&]{
        vibrate();
    });
    // vibrate();
}
```

```cpp
class PhoneCall {
    // ???
};
```
Usage

Contribution with Elektra

Code Generator

Specif"ication

Persistent Contextual Values (\(=\)Layers)

Code

Written by User

Code generates Persistent Contextual Values (=Layers)

uses runs on
Solution

- **directly activate CVs (every CV works as layer!)**

```c++
void greet (Person & p, Country & country,
            Location & location) {
  p.activate(country);
  p.activate(location);
  cout << p.greeting << endl;
}
```

- **sync CVs between processes**

```c++
void userInteraction(Accuracy const & a) {
  a.context().sync(); // a might change
  for (long i=0; i<a; ++i) {
    /* a does not change here */
  }
}
```
Activation via Assignment

- **assignments** on CVs trigger layer activations
- **sync** triggers all necessary assignments
- implication: we do not need extra layers anymore

```csharp
void assignLanguage(Language & lang) {
    lang.context().activate(lang);
    lang = "";
    // layer lang deactivated
    lang = "spanish";
    // layer switch to spanish
    lang.context().deactivate(lang);
    lang = "english";
    // layer still deactivated
}
```
Evaluation
Benchmark

- access: no overhead
- 4 benchmarks

```cpp
void benchmarkReload(vector<CV> & cv) {
    vector<kdb::KDB> kdb;
    kdb.resize(1000);
    t.start();
    for (long i = 0; i < 1000; ++i)
    {
        kdb[i].get(c.values(), "\test");
        c.sync();
        x ^= tcv + tcv;
    }
    t.stop();
}
```
Case Study Webserver

```
1 httpperf --hog --timeout=1 --num-conn=50000
2 --rate=2200 --num-call=1 --server=127.0.0.1
```
Source Code

- Source Code released as free software within Elektra
  - >70 predefined plugins
  - support for hundreds kinds of configuration files
  - integrate standard software
  - specification is configuration (e.g. in XML, JSON)

- http://www.libelektra.org
  - version 0.8.18 released at 2016-09-16
Conclusion

- combination of performance, context awareness and customization
- CVs with code generation in multi-threaded and multi-process applications
- CVs can be *shared* across applications
- implementation is **free software** and can be downloaded from [http://www.libelektra.org](http://www.libelektra.org)
- supports mobile development in C++, Java, and more
- **benchmark**: overhead increases linearly with CVs
- **case study**: only with dominant layer activations performance decreases
Thank you for your attention!

Markus Raab
Vienna University of Technology
Institute of Computer Languages, Austria
Email: markus.raab@complang.tuwien.ac.at
Example: Hardware Abstraction

- hardware as context

```
(hw/pi/pi/gpio/folder = /sys/class/gpio/
/hw/pi/pi/gpio/tamper = gpio7
/hw/pi/elitebook/gpio/folder = ~/context/pi
/hw/pi/elitebook/gpio/tamper = tamper.txt
```

(this is a configuration file, not a specification!
But they are both part of Program Execution Environment)

- layer activations for sensor states

```cpp
select(fd+1, 0, 0, &fds, 0);
t.c().activate<Tamper>();
```

```cpp
t.c().syncLayers();
if (t) out<< "tamper!!!";
```
Benchmark Setup

- **hp ® EliteBook 8570w ™**
  - CPU Intel ® Core i7-3740QM @ 2.70GHz
  - 7939 MB Ram
- GNU/Linux Debian Jessy 8.4
- gcc compiler Debian 4.9.2-10
  - with the options -std=c++11, -O2
- measured the time using `gettimeofday`
- median of eleven executions
Some Related Work

context variables (check on every usage)

ensure-active-layers (global layer activation)
P. Costanza, R. Hirschfeld, and W. De Meuter, “Efficient layer activation for switching context-dependent behavior,” in Modular Programming Languages

partial evaluation avoids usage of libxml2
M. Jung, R. Laue, and S. A. Huss, “A case study on partial evaluation in embedded software design,” in SEUS 2005

hybrid mediator-observer pattern
O. Riva, C. di Flora, S. Russo, and K. Raatikainen, “Unearthing design patterns to support context-awareness,” in Pervasive Computing and Communications Workshops
Example: Battery low

c1.activate<BatteryLow>();

Thread 1

c1.deactivate<BatteryLow>();

// Security unchanged

c2.syncLayers();

// BatteryLow active

c2.activate<Security>(cv);

Thread 2

// BatteryLow inactive
Big Picture

- **Persistence**
  - get value derived from env
  - • Runtime Checks
  - • Validation
  - • Upgrades
  - • Integration

- **Specification**
  - provides

- **C-Code**
  - (e.g. option parsing)

- **C++-Code**
  - generate

- **Docu, Manuals, GUIs**

- **Contextual Values Connection**

- **/%/%/%/person/greeting=Hi!**
- **/German/%%/person/greeting=Guten Tag!**
- **/German/Austria/%/person/greeting=Servus!**
- **/German/Austria/t/person/greeting=Griaß enk!**