Targeted on-line data extraction with SystemXtract

Johannes Schützel
University of Rostock Modeling & Simulation Group

SIMUTools Conference
August 26, 2015
Organization of this talk

1. Motivation
2. SystemXtract
   1. Application context
   2. Specification language
   3. Extraction mechanism
3. Performance and potentials
We need targeted on-line data extraction for multiple reasons

non-targeted
- interesting data is hard to spot

off-line
- requires space and time for temporary storage
We need targeted on-line data extraction for multiple reasons

**non-targeted**
- interesting data is hard to spot

**off-line**
- requires space and time for temporary storage

**targeted**
- extracted data = data of interest
- less storage space required

**on-line**
- data is available live during simulation
We need targeted on-line data extraction for multiple reasons:

- Extracted data = data of interest
- Less storage space required
- Interesting data is hard to spot
- Data is available live during simulation
- Requires space and time for temporary storage

Targeted on-line data extraction requires two things:

1. Specification language
2. Extraction mechanism
The language design must fit to how data is organized

Messages with values

device state change: secondary -> activated (reason 'none') [90 100 0]
SystemXtract facilitates the specification of Targets

```python
extract to stdout {
    targets [
        FOO,
        BAR with (lvl = INFO),
        BAZ with (msg ~ "found")
    ]
}
```
SystemXtract facilitates the specification of Targets

extract to stdout {
  targets [
    FOO,
    BAR with (lvl = INFO),
    BAZ with (msg ~ "found")
  ]
}

list of 3 targets
SystemXtract facilitates the specification of Targets

```
extract to stdout {
    targets [
        FOO,
        BAR with (lvl = INFO),
        BAZ with (msg ~ "found")
    ]
}
```

Value matching and constraining

```
... GAK with (msg ~ "* s/n: $INT:s$ $INT:n$ *", s/n > 2.5)
```
Johannes Schützel: Targeted on-line data extraction with SystemXtract

SystemXtract facilitates the specification of Targets

```plaintext
extract to stdout {
    targets [
        FOO,
        BAR with (lvl = INFO),
        BAZ with (msg ~ "found")
    ]
}
```

- everything from module FOO
- everything from BAR with level INFO
- everything from BAZ whose message begins with "found"

Value matching and constraining:

```plaintext
... GAK with (msg ~ "* s/n: $INT:s$ $INT:n$ *", s/n > 2.5)
```

- match bind to s
- match bind to n
- constrain

Value matching and constraining
SystemXtract facilitates the specification of **Extraction Phases**

```plaintext
extract to stdout {
    targets [ * ]
    start at (BAR.msg ~ "found $INT:x$", x > 50)
    stop after (100)
}
```
SystemXtract facilitates the specification of Extraction Phases

```
extract to stdout {
    targets [ * ]
    start at (BAR.msg ~ "found $INT:x$", x > 50)
    stop after (100)
}
```

everything
SystemXtract facilitates the specification of Extraction Phases

extract to stdout {
  targets [ * ]
  start at (BAR.msg ~ "found $INT:x$", x > 50)
  stop after (100)
}

Phases:

- start point — $\infty$
- $t_0$ — stop point
- start point — stop point
- $x$ time units after a start event
- $x$ time units ahead of a stop event

start/stop points may be times or events

everything from whenever observing this to 100 milliseconds afterwards
Extraction can be realized by filtering

Data Item

<table>
<thead>
<tr>
<th>Time-stamp</th>
<th>Origin</th>
<th>Level</th>
<th>Message</th>
</tr>
</thead>
</table>

Instrumented Model

Sensor Driver Software


Data Streamer

TCP/IP

Model

SOCK.IP

S Sock ACC Gyr Mag.

S

O

C

K

I

F

A

P

I

F

A

C

C

E

G

Y

R

O

M

A

G
Extraction can be realized by filtering

Data Item

<table>
<thead>
<tr>
<th>Time-stamp</th>
<th>Origin</th>
<th>Level</th>
<th>Message</th>
</tr>
</thead>
</table>

Instrumented Model

Stream
Extraction can be realized by filtering

Data Item

<table>
<thead>
<tr>
<th>Time-stamp</th>
<th>Origin</th>
<th>Level</th>
<th>Message</th>
</tr>
</thead>
</table>

Specification

```plaintext
extract to stdout {
  targets [
    ACCEL,
    GYRO with (lvl = INFO),
    MAG with (msg ~ "found *")
  ]
}
```
Filtering can be realized by simple filter nodes

```java
class OriginFilter {
    String origin;

    //...

    onInput(DataItem d) {
        if (d.getOrigin().equals(origin))
            forward(d);
    }
}
```
Specifications can be mapped to filter graphs

Mapping of Targets

targets [
  FOO,
  BAR with (lvl=INFO),
  BAZ with (msg~"found *")
  GAK with (msg~"* $INT:x$", x > 9)
]
Specifications can be mapped to filter graphs

Mapping of Targets

targets [
  FOO,
  BAR with (lvl=INFO),
  BAZ with (msg~"found *")
  GAK with (msg~"* $INT:x$", x > 9)
]
Specifications can be mapped to filter graphs

Mapping of Targets

targets [  
  FOO,  
  BAR with \(\text{lvl}=\text{INFO}\),  
  BAZ with \(\text{msg} \sim \text{"found \ *\"} , \ x > 9\)  
  GAK with \(\text{msg} \sim \text{"\* $\text{INT}:x$", \ x > 9\})  
]
Specifications can be mapped to filter graphs

Mapping of Phases

- **start at (<Start>)**
- **stop at (<Stop>)**

- **start at (<Event>)**
- **stop after (100)**

- **start ahead (100)**
- **stop at (<Event>)**

... maps to ...

![Diagram](attachment://filter_graph.png)
Specifications can be mapped to filter graphs

Mapping of Phases

- start at \(<\text{Start}>\)
- stop at \(<\text{Stop}>\)
- start at \(<\text{Event}>\)
- stop after (100)
- start ahead (100)
- stop at \(<\text{Event}>\)

... maps to ...

- StartStopSwitch
- IntervalAfterEventSwitch 100
- IntervalAheadEventSwitch 100
Specifications can be mapped to filter graphs

```plaintext
extract to file "log.txt" {
  targets [ACCEL, GYRO, MAG, AP_IF, SOCK_IF with (msg~"I2C*")]
  start at (PMU.msg~"* status=2")
  stop at (PMU.msg~"* status=0")
}
```
SystemXtract is an independent tool

Simulation Application

Model

SystemXtract

Specification

Parser

Mapper

Input Handler

Parse Tree

Filter Graph

Output
First experiments show insignificant overhead

- **no filtering (base line)**: 121.41 µs
- **minimal filtering**: 122.81 µs (+0.2%)
- **12.5% filtering**: 124.2 µs (+0.4%)
SystemXtract is an independent tool

Simulation Application

Model

Parser

Mapper

Input Handler

Specification

SystemXtract

Parse Tree

Filter Graph

Output
There is still potential for improvement

<table>
<thead>
<tr>
<th>Pre-Runtime</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensive instrumentation of model code</td>
<td>Parsing</td>
</tr>
<tr>
<td>Compilation</td>
<td>Creation of filter graph</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
</tr>
<tr>
<td></td>
<td>Filtering</td>
</tr>
<tr>
<td></td>
<td>(Targets&amp;Phases)</td>
</tr>
</tbody>
</table>
There is still potential for improvement

**Pre-Runtime**
- Extensive instrumentation of model code
- Compilation
- Parsing
- Creation of filter graph

**Runtime**
- Simulation
- Filtering (Targets & Phases)
- Compilation

↕ Filtering (Targets & Phases)
Targeted on-line data extraction with SystemXtract

Johannes Schützel
johannes.schuetzel@uni-rostock.de

Sebastian Stieber
sebastian.stieber2@uni-rostock.de

Christian Haubelt
christian.haubelt@uni-rostock.de

Adelinde M. Uhrmacher
adelinde.uhrmacher@uni-rostock.de

1 Modeling and Simulation Group
Institute of Computer Science
University of Rostock

2 Institute for Applied Microelectronics
and Computer Engineering
University of Rostock