

The Language

XChange

rule-based programming of
reactive behaviour on the Web

Outline

- Overview, Motivation
- Paradigms, Concepts
 - Xcerpt
 - Events
- Use Case
- Related approaches

Introduction

- XChange
 - A declarative approach to reactivity on the web
 - High-level and rule-based reactive Web language.
 - Allows programming both locally and distributed over several Web nodes.
- By the Programming and Modeling Languages research group of the Institute for Informatics, University of Munich
- Research as part of the REWERSE project funded by the EU Commission and Switzerland

[demo, 1]

Overview (1): Motivation

- Bridge the gap between the existing, passive Web and the dynamic Web
- Rule based programming brings in
 - declarativity
 - fine-grain modularity
 - higher abstraction
 - natural language like syntax
- Rule based programs are easier to write, understand and maintain, including for non-technical users.

[xchange, 57]

Overview (2)

- Reactive Web applications:
 - Respond to events such as
 - User requests
 - Changes to local or remote data
 - Messages from other applications
 - Common reactions are
 - Updating data or
 - Raising new events (local & remote)

[demo, 1]

Overview (3): ECA Rules

- Event-Condition-Action Rules
 - Are reactive rules, which
 - describe state changes
 - are also called active or dynamic rules
 - Are different to
 - Deduction rules (logical inference)
 - Normative or structural rules (consistency!)
 - Production rules
 - Production rules are reactive rules
 - of the form IF Condition DO Action

[rif, 5-9]

Overview (4): ECA Rules

- General form of ECA rules:
 - ON Event IF Condition DO Action
- ECA rules examples:
 - ON item out of stock
DO set item's status to "not available"
 - ON item out of stock
IF item is in stock at one of the shop's suppliers
DO reorder item and set status to "reordered"

[reactiveWeb, 187]

Overview (5): XChange introduction

- XChange: Event-Condition-Action Rules (ECA rules)
- A XChange program consists of one or more ECA rules
- A XChange ECA rule consists of
 - Event Query (Event)
 - Web Query (Condition)
 - Action
- A program runs locally at some Web site, accessing local and remote data as reaction to events.
- A reaction can trigger further reactions, local or remote.

[xchange, 57]

XChange concepts & paradigms (1)

- Event
 - A happening to which each Web site may decide to react in a particular way or not.
 - Examples:
 - Update to a Web resource
 - A query posed to Web resources
 - 8 o' clock every morning

[xchange, 8]

XChange concepts & paradigms (2)

- Event Query [xchange, 58]
 - Queries against event data
 - Double purpose:
 - Detecting events of interest and temporal combinations of them
 - Selecting data items from events' representation.
 - Variables are used as place holders for data items, which can be used in the other parts of XChange rules.
- Events are changes in the state of the world and event queries are queries against their representation.

XChange concepts & paradigms (3)

- Volatile vs. Persistent data
 - Volatile data: Event data communicated on the Web between XChange programs
 - Persistent data: Data of Web resources, such as HTML or XML documents.
- Event query: Query against volatile data.
- Web query: Query against persistent data.

[xchange, 58]

XChange concepts & paradigms (4)

- Web Queries
 - Represent the condition part
 - Gather data as variable bindings
 - Query directly existing data
 - or query views constructed by means of deductive rules
 - A XChange Web Query is a Xcerpt query.

[xchange, 94]

Xcerpt (1)

- Developed at the University of Munich
- Makes use of a **pattern-based approach** for selecting data items and constructing new data.
- Incomplete Patterns (handle data without schema or with irregular structure)
- Offers **Simulation Unification**
 - A novel unification method
 - Essential for event queries
- Is a **rule based** language (deduction rules)

[xchange, 30]

Xcerpt (2)

- Backward chaining
- Separation of Querying and Construction
- Reasoning Capabilities
- Language constructs: XML or compact
 - Building blocks are terms:
 - Query terms
 - Construct terms
 - Data terms (Web content)

[xchange, 30-31]

Xcerpt (3): Example

Data terms

```
accommodation {
  currency {"EUR"},
  city {"Paris"},
  country {"France"},
  hotel {
    name {"Princesse Isabelle"},
    category {"3 stars"},
    price-per-room {"112"}
  },
  hotel {
    name {"Corail"},
    category {"2 stars"},
    price-per-room {"65"},
    no-pets { }
  }
}
```

Query terms

```
accommodation {{
  hotel {{ }}
}}
accommodation {{
  city {"Paris"},
  var V -> hotel {{ }}
}}
```

[xchange, 30-31]

XChange concepts & paradigms (5)

- Update Patterns (1)
 - XChange follows a pattern based approach
 - primarily developed for updating XML data
 - XML data is represented with “Data terms”
 - Data terms can be
 - Local
 - or remote at a web resource
(having another XChange program running)

[xchange, 94-95]

XChange concepts & paradigms (6)

- Update Patterns (2)
 - Simple example: Update Term Specifying Insertion

```
a {{  
  b {{ }},  
  insert d { e{}, f{} },  
  h {{ }}  
}}
```

Before update

```
a {  
  b { i{}, j{} },  
  h { k{} }  
}
```

After update

```
a {  
  b { i{}, j{} },  
  d { e{}, f{} }  
  h { k{} }  
}
```

[xchange, 96-97]

XChange concepts & paradigms (7)

- Elementary updates
- Complex updates
 - are an ordered or unordered,
 - conjunction or disjunction of updates.
 - Conjunction of updates:
 - XChange executes all specified updates.
 - Disjunction of updates:
 - XChange executes the specified updates until one was successful.

[xchange, 109-112]

XChange concepts & paradigms (8)

- Transactions
 - Are executed in an “all-or-nothing manner”
 - Are groups of
 - Update specifications
 - Explicit event specifications
 - Transactions have to obey the ACID properties.
 - Handling distributed transactions poses new challenges.
 - XChange recognizes the need for transactions, but doesn't implement them as of now.

[xchange, 112-113]

XChange: Events (1)

- Atomic events
(update, query, system events, remote events,...)
 - Example: A atomic event query
 - detects event messages notifying a phone conference
 - Subject and time are bound to the variables S and T.

```
xchange:event {{
  xchange:sender {"http://organiser.de/secretary/"},
  phone-conference {{
    subject { var S },
    time { var T }
  }}
}}
```

[xchange, 70]

XChange: Events (2)

- Composite events
 - Use more than one atomic events.
 - Are defined as answers to composite event queries
 - Offer
 - temporal restrictions
 - event compositions
 - Conjunctions (ordered or unordered)
 - Inclusive Disjunctions
 - Exclusions (no instance for a finite time interval)

[xchange, 74-79]

XChange: Events (3)

- Event Messages
 - communicate event data between (same or different) Web sites
 - contain information about the sender web site and the event
 - Communication follows a push strategy, so web sites inform other web sites about events.
 - Subscription might be implicit, e.g. booking a flight implicitly subscribes to related events

[xchange, 64]

XChange: Rules

- XChange supports different types of rules:
 - Reactive rules (Event query - Web query - Action)
 - Event Raising Rules
are means for notifying reactive (XChange-aware) Web sites
 - Transaction rules
are means for updating persistent data on the Web
 - Deductive rules (Xcerpt rules)
are means for constructing views over heterogeneous data sources

[xchange, 114-118]

XChange: Transaction rule example (1)

- Travel organiser of Mrs. Smith uses this rule:
 - If the return flight of Mrs. Smith is cancelled
 - Then look for and book another suitable flight.
- Shape of the rule:

```
TRANSACTION
  <make flight reservation>
ON
  <event query detecting flight cancellations notifications>
FROM
  <Web query looking for another suitable flight>
END
```

[xchange, 116]

XChange: Transaction rule example (2)

```
TRANSACTION
  in { resource { "http://airline.com/reservations/" },
      reservations {{
        insert reservation { var F, name { "Christina Smith" } }
      }}
  }
ON
  xchange:event {{
    xchange:sender { "http://airline.com" },
    cancellation-notification {{
      flight {{ number { "AI2021" },
                date { "2005-08-21" } }}
    }}
  }}
FROM
  in { resource { "http://airline.com" },
      flights {{
        var F -> flight {{
          from { "Paris" }, to { "Munich" },
          date { "2005-08-21" }
        }}
      }}
  }
END
```

Use Case: ECSS (1)

- A distributed scientific community of historians called “Eighteenth Century Studies Society” (ECSS)
- ECSS consists of
 - participating universities
 - thematic working groups
 - project management
- The parts have their own autonomous web sites, but they cooperate to react and distribute changes.

[demo, 1]

Use Case: ECSS (2)

- Each web node has its own XChange program consisting of several rules
- The sites maintain XML data about members, publications, meetings, library books and newsletters.
- Data is often shared:
 - Member's personal data is present at her home university, the management site and in the working groups she participates in.
- To keep the data consistent XChange rules are used.

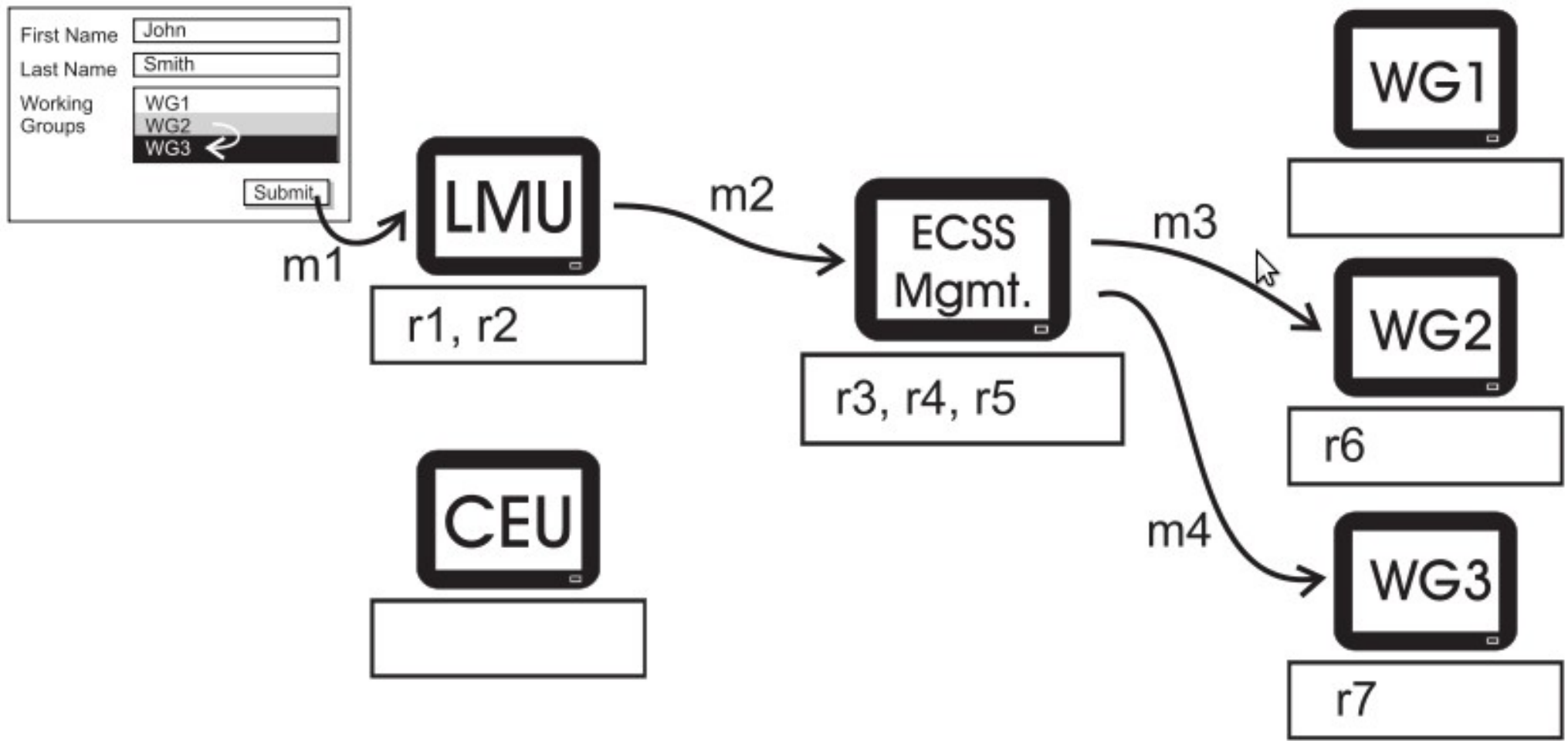
[demo, 1-2]

Use Case: ECSS (3)

- Changes are made at the member's home institution, sending an event to the LMU site.
- At the LMU site a rule reacts and updates the local data accordingly. Another rule forwards an event to the management site.
- The management site updates its local data and propagates events to the affected working groups.
- The working groups react and update their data.

[demo, 2]

Use Case: ECSS (4)



[demo, 2]

Use Case: ECSS (5)

- Needed rules

r1: ON change member

DO update LMU data

r2: ON change member

DO forward to management

r3: ON change member

DO update management data

r4: ON change member (w/WG3)

IF was not member of WG3

DO send add member to WG3

r5: ON change member (w/o WG2)

IF was member of WG2

DO send remove member to WG2

r6: ON remove member

DO update WG2 data

r7: ON add member

DO update WG3 data

[demo, 2]

XChange: Status

- Quite a few publications from 2004 to 2006.
- There is a prototype available written in Haskell
 - Last update 28.07.2008
 - It's readme states it “State of development”
 - Remote updates are not yet implemented
- Get it at <http://reactiveweb.org/xchange/prototype.html>

[<http://reactiveweb.org/xchange>]

Related approaches: REVERSE

- REVERSE:
 - a research "Network of Excellence" (NoE)
 - Reasoning on the Web with Rules and Semantics
 - Working group I5 (Evolution and Reactivity)
- Languages, Tools and Methodologies developed
 - XChange: Language to program reactivity on the Web.
 - r3: A semantic web rule engine for reactive rules of different formats.
 - MARS: Modular active rules for the semantic web framework.

[reverse3, 1,10]

r3: Resourceful Reactive Rules (1)

- <http://centria.di.fct.unl.pt/~reverse/wg-i5/r3/>
- deals with language *heterogeneity*!
- Autonomous web nodes can use *different formalisms* for ECA rules, and also different formalism for events, conditions and actions, depending on the requirements of their applications.
- r3 is general ECA language, combining those sub-languages for reacting and performing evolution.
- In contrast to that XChange is a concrete language for all ECA components.

[general, 103]

r3: Resourceful Reactive Rules (2)

- Resourceful
 - Resources provide the foundation for the Semantic Web
 - allows the manipulation of rules as pure semantic objects
 - reason about rules, leading to a truly adaptive Semantic Web!
 - Prototype provides a foundational ontology
 - Supports already xchange, xcerpt, xquery, xslt, xpath, and others

[<http://centria.di.fct.unl.pt/~reverse/wg-i5/r3/DOC/2006/eval/index.html>]

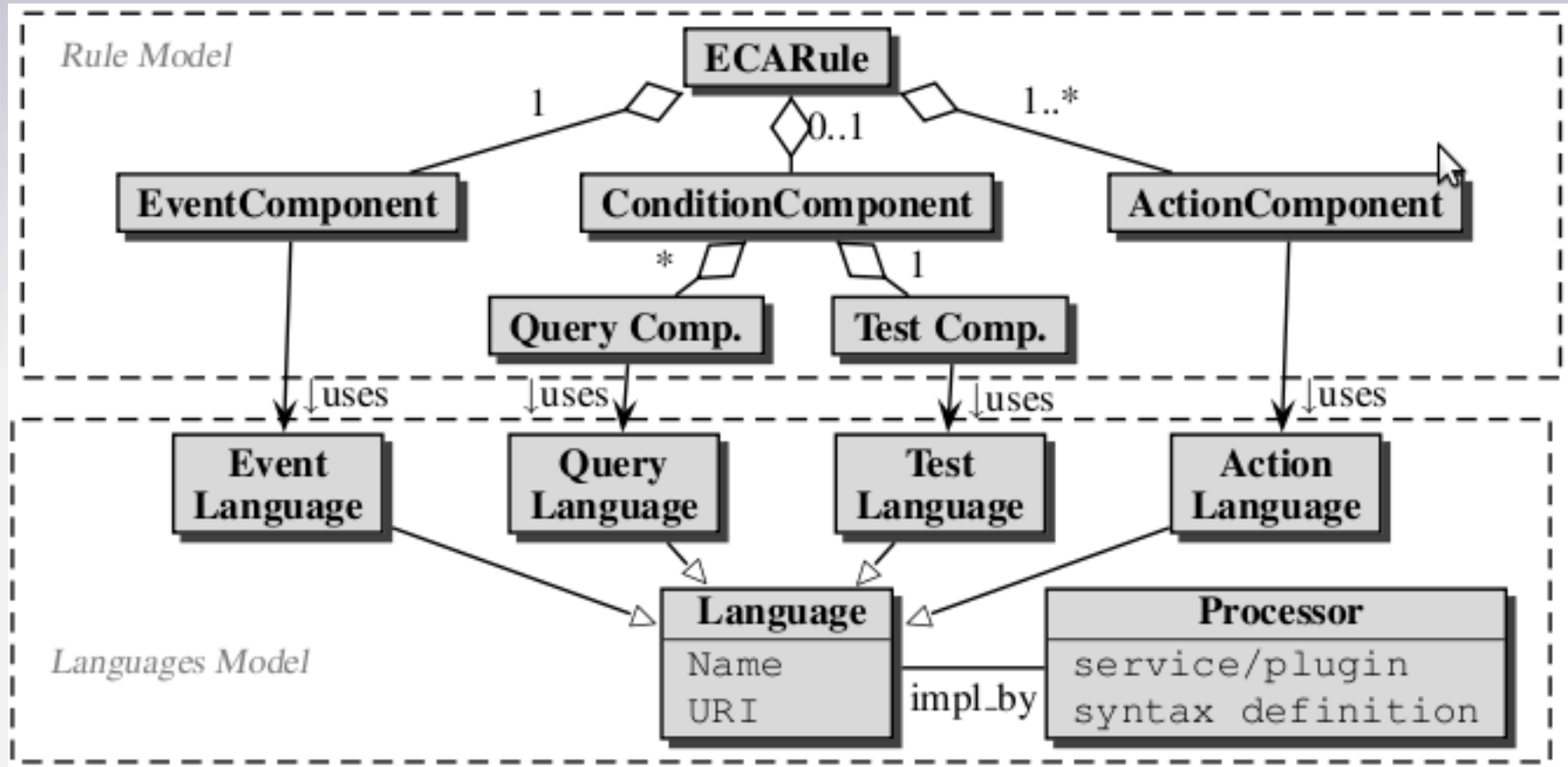
[r3ontheway, 2]

MARS (1)

- heterogeneous event, query, and action languages
- Condition component is divided into queries and a test component.
- User can register rules in the ECA-ML language at an ECA service that provides the infrastructure and global rules semantics
- ECA-ML has been designed by the MARS group

[marsdemo, 1]

MARS (2)



[marsdemo, 2]

MARS (3)

- For processing the ECA engine determines a language processor node and submits the task to the node.
- Service identification is done by a Language&Service registry.
- Information flow between components is possible by logical variables
- There is a prototype and a online interface for testing at <http://www.semwebtech.org/mars/frontend>

[marsdemo, 1]

Summing-up

- Thanks to REVERSE working group I5 (Evolution and Reactivity) there is a lot of research in active or ECA rules for the web.
- XChange is a homogeneous approach for reactive web rules.
- Heterogeneous solutions try to make different approaches compatible and combinable.
- The REVERSE project finished with 29.02.2008.

References (1)

[demo]

François Bry, Michael Eckert, Hendrik Grallert, Paula-Lavinia Pătrânjan: Reactive Web Rules: A Demonstration of XChange

[xchange]

Paula-Lavinia Pătrânjan: The Language XChange: A Declarative Approach to Reactivity on the Web.

[rif]

Harold Boley, Michael Kifer, Paula-Lavinia Pătrânjan and Axel Polleres: Rule Interchange on the Web.

[reactiveweb]

Bruno Berstel, Philippe Bonnard, François Bry, Michael Eckert, and Paula-Lavinia Pătrânjan: Reactive Rules on the Web.

References (2)

[rewise3]

Uta Schwertel, University of Munich: REWERSE

Reasoning on the Web with Rules and Semantics, Project
Presentation Year 3, 2007,

http://rewise.net/downloads_demos/presentations/rewise-y3-project-presentation.ppt

[general]

JosÈ Júlio Alferes, Ricardo Amador, Wolfgang May: A General
Language for Evolution and Reactivity in the Semantic Web

[r3ontheway]

JosÈ Júlio Alferes, Ricardo Amador: r3: On the way to Resourceful
Reactive Rules

[marsdemo]

Erik Behrends, Oliver Fritzen, Wolfgang May, Franz Schenk, Daniel
Schubert: A Framework and Components for ECA Rules in the
Web (Demo)