Task Oriented Programming

Rinus Plasmeijer
Peter Achten - Pieter Koopman
(Radboud University Nijmegen)

Jan Martin Jansen (NLDA - Dutch NAVY)

Bas Lijnse (RUN - Dutch Coastguard)
Jurrien Stutterheim (RUN - TNO) - Wessel van Staal (RUN - TNO)

Laszlo Domoszlai (ELTE)
Internet Enables Us to Work Together Distributively

- Computer
- Laptop
- Smart Phone
- Tablet

Internet

- Cloud / Database
- Application Software
- Handle Data Storage
- Data Retrieval

- “Cool” User Interfaces
- Software on Client
- Handle Communication
- HandleMutiple Users
- Support Sharing of Data

Coordinate Collaboration

Coordination
Why Task Oriented Programming?

Tasks are a common notion in daily life / in any organization

- a task to do, assigned to one person
- group of people working together: collection of subtasks
  - statically known <-> dynamically determined
- managing tasks is also a task

Computers often play an important role when performing a Task

- Filling in a web form
- Answering an email
- Writing a piece of text, filling in a spreadsheet, ...
- Using an “App”

Many programming concepts can be seen as Tasks to be executed by a computer

- Function call, procedure call, method invocation, calling a web-service, a query
- process, thread, "App"
**TOP Approach**

1. **Formally describe the way of working by defining the tasks to do**
   - We are able to describe arbitrary complex, “dynamic” ways of working
   - **Declarative**: no worry about the technical details

2. **Automatically generate the entire system out of it: description = implementation**
   - No additional software to write

3. **Direct feedback to and response from customer**
   - **Rapid Prototyping**, saves lots of time and money
   - **Agile Development** with designer and end-user
   - Also suited for **Simulation** and **Training**

**Huge Application Area**

- (e) Health Care, Insurance Market, Systems for Economical Market
- (e) Government, Legal Systems, Enterprise Resource Planning, …
- Social Media type of applications, …
- Soft real-time distributed systems with close collaborations between men-machine
Task Oriented Programming with

- Domain Specific Programming Language, embedded
  - Specify Basic Tasks
  - Special Sequential and Parallel Combinators for composing tasks

- Abstracts from: (thanks to advanced Functional Programming techniques)
  - graphical user interfaces & handling of user-interaction
    - Fully automatically generated from the type
  - (client-server) communication
  - informing tasks about the progress made by others
  - informing tasks when shared information is being changed

- Resulting application: Web-Service coordinating the tasks to be done...
  - Run on computers, laptops, tablets, smart phones
  - Tasks can run on server (native code) as well as client (javascript)
  - One source code for everything, strongly typed
Standard iTASK Client
Small Example

twoTasks :: User User (Shared a) → Task a | iTask a
twoTasks user1 user2 sv
  = user1 @: updateSharedInformation sv -||- user2 @: viewSharedInformation sv
iTasks Architecture
iTasks Architecture
iTasks Architecture

- Computer
  - iTasks Client
  - iTask User

- Laptop
  - iTasks Client

- Phone
  - iTasks Client

- Tablet
  - iTasks Client

- Sensor
  - iTasks Client

- Web Service
  - iTask Web Server
  - iTask Combinator
  - Library
  - My iTask Specification

- Web Service
  - iTask Web Server
  - iTask Combinator
  - Library

- Cloud / Database
  - RPC
  - iTask User
  - My iTask Specification
Case Study: Coast Guard Search And Rescue
Incidone – Coast Guard Search and Rescue Support

Kitesurfer injured Scheveningen
Report of kitesurfer with possible injuries in water at Scheveningen

Coordinate incident

Situation
Coordinate the incident as you consider appropriate. You may use the form below to keep everything.
Title: Kitesurfer injured Scheveningen
Summary: Report of kitesurfer

Refinements
- Title
- Description
- Contact

Ievoli Amaranth
Contact no: 3
Type: Vessel
Name: Ievoli Amaranth

Map of the area showing the location of the incident.
What did we learn?

- We must be able to **constantly observe** the current state / value of a task **while** the work goes on
  - the iTask system needs to be a reactive system...

- **Share**: we need one concept for storing information (memory, files, databases, sensors, ...)
  - automatic inform tasks when shared information has been changed
  - present (shared) information to every task differently, in a user definable dashboard view

- Managing is also a task...
  - tasks which can monitor and coordinate other tasks

And of course...
- Powerful enough to make state-of-the-art multi-user web-applications
- Also suited for tablets, mobile phone, evaluation on the client, distributed evaluation, ...
- Easy to programme: declarative -> hide all technical details
- Semantics well defined
CurrentWork

- **Real** real-world applications
  - Prototype New System for the Coast Guard (Bas Lijnse, PostDoc)
  - Investigate new efficient ways of collaboration
    (TNO Jurrien Schutterheim, PhD; Wessel van Staal, Master thesis)
- **Applicability**
  - efficiency, scalability, security, version management, collaboration existing systems...
  - distributed servers / client side evaluation (Laszlo Domoszlai, ELTE)
  - enable Graphical TOP specification for non-programmers
  - simulation (agent based) + training
- **Semantics**
  - Reasoning ? Proving ? Testing ?
Questions ?
Papers on iTasks

First paper on iTasks:

Extensions:
- iTasks for a change - Type-safe run-time change in dynamically evolving workflows (PEPM 2011)
- GiN: a graphical language and tool for defining iTask workflows (TFP 2011)
- iTask as a new paradigm for building GUI applications (IFL 2010)
- Getting a grip on tasks that coordinate tasks (LDTA 2011)

Semantics:
- An Executable and Testable Semantics for iTasks (IFL 2008)
  Task Oriented Programming in a Pure Functional Language (PPDP 2012)

Client site evaluation of tasks:
- Transparant Ajax and Client-Site Evaluation of iTasks (IFL 2007)
- iEditors: Extending iTask with Interactive Plug-ins (IFL 2008)

Applicability:
- A Conference Management System based on the iData Toolkit (IFL 2007)
- Web Based Dynamic Workflow Systems for C2 of Military Operations (ICCRTS 2010)
- Managing COPD exacerbations with telemedicine (AIME 2010)
- Towards Dynamic Workflows for Crisis Management (ISCRAM 2010)
- Capturing the Netherlands Coast Guard's SAR Workflow with iTasks (ISCRAM 2011)
- A Task-Oriented Incident Coordination Tool (ISCRAM 2012)
1. How do we formally describe the way of working?

⇒ **Task Oriented Programming**

New style of programming: *Tasks* as central concept

- **Intuitive:** Tasks are a common notion in daily life / in any organization

- **Expressive:** One can indeed **formalize** arbitrary, complex, “dynamic” ways of working
  - Data Driven
  - Tasks can observe each other state while the works takes place
  - Management is expressible as a task as well
  - Reliable: strongly typed
  - Tasks are: compositional, re-usable, higher-order
  - Well defined semantics based on a small number of basic concepts
2. Can we generate an entire system out of it?

- **Proof-of-Principle** implementation of TOP
  Using very advanced programming language techniques of

- Generates a **multi-user web-service** coordinating the described way of working
  
  ✫ **Supports modern devices**

  ✤ **Declarative:** no worry about the technical details
  
  Automatic generation and handling of: interactive web applications, GUI-elements (windows, menus, buttons), shared information, storage and retrieval of information, persistence, communication between distributed tasks, encoding / decoding, …

  ✤**Not yet directly applicable in real world environments…**
  Lots of additional work is needed one cannot do in research environments…
  performance, scalability, easy to deploy, maintain & upgrade, …
  access to common data bases, exchange formats, …
  documentation, tutorials, …