Components, Objects, and Contracts

Olaf Owe and Gerardo Schneider and Martin Steffen

IfI UiO

3. September 2007

Components, Objects, and Contracts

Olaf Owe and Gerardo Schneider and Martin Steffen

IfI UiO

3. September 2007

The title of my talk is “Components, Objects, and Contracts, which is perhaps slightly non-descript. joint work . I do not have the time to go into details. But in a nutshell : the goal of this research is to enhance (object-oriented) components with a notion of contract.

Of course, the term contract is not new in connection with components or software development. The most well-known use of contracts is in connection with the design-by-contract methodology (in context of the Eiffel programming language).

Contracts & e-contracts

> "A contract is a binding agreement between two or more persons that is enforceable by law." [Webster online]

Contracts & e-contracts

This deed of **Agreement** is made between:
1. **[name]**, from now on referred to as **Provider** and
2. the **Client**.

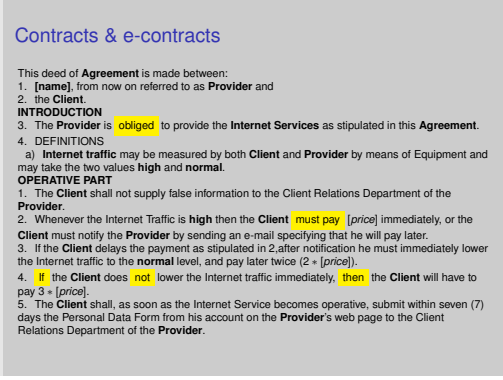
INTRODUCTION

3. The **Provider** is obliged to provide the **Internet Services** as stipulated in this **Agreement**.
4. DEFINITIONS

a) **Internet traffic** may be measured by both **Client** and **Provider** by means of Equipment and may take the two values **high** and **normal**.

OPERATIVE PART

1. The **Client** shall not supply false information to the Client Relations Department of the **Provider**.
2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.
3. If the **Client** delays the payment as stipulated in 2,after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ($2 * [price]$).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay $3 * [price]$.
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider**'s web page to the Client Relations Department of the **Provider**.



I will not expect you to read all this, but as so much. The contract is split into two parts, a definitorial part (agreeing on words etc) and the part of the contact proper, i.e., stating what the involved parties are supposed to do.

Contracts & e-contracts

> "A contract is a binding agreement between two or more persons that is enforceable by law." [Webster online]

Definition

A contract is a document which engages several parties in a transaction and stipulates their obligations, rights, and prohibitions, as well as penalties in case of contract violations.

Goal

- develop a notion of component model
- interface description by deontic contracts
- formal model for e-contracts
- formal semantics
- executable
- using Creol language

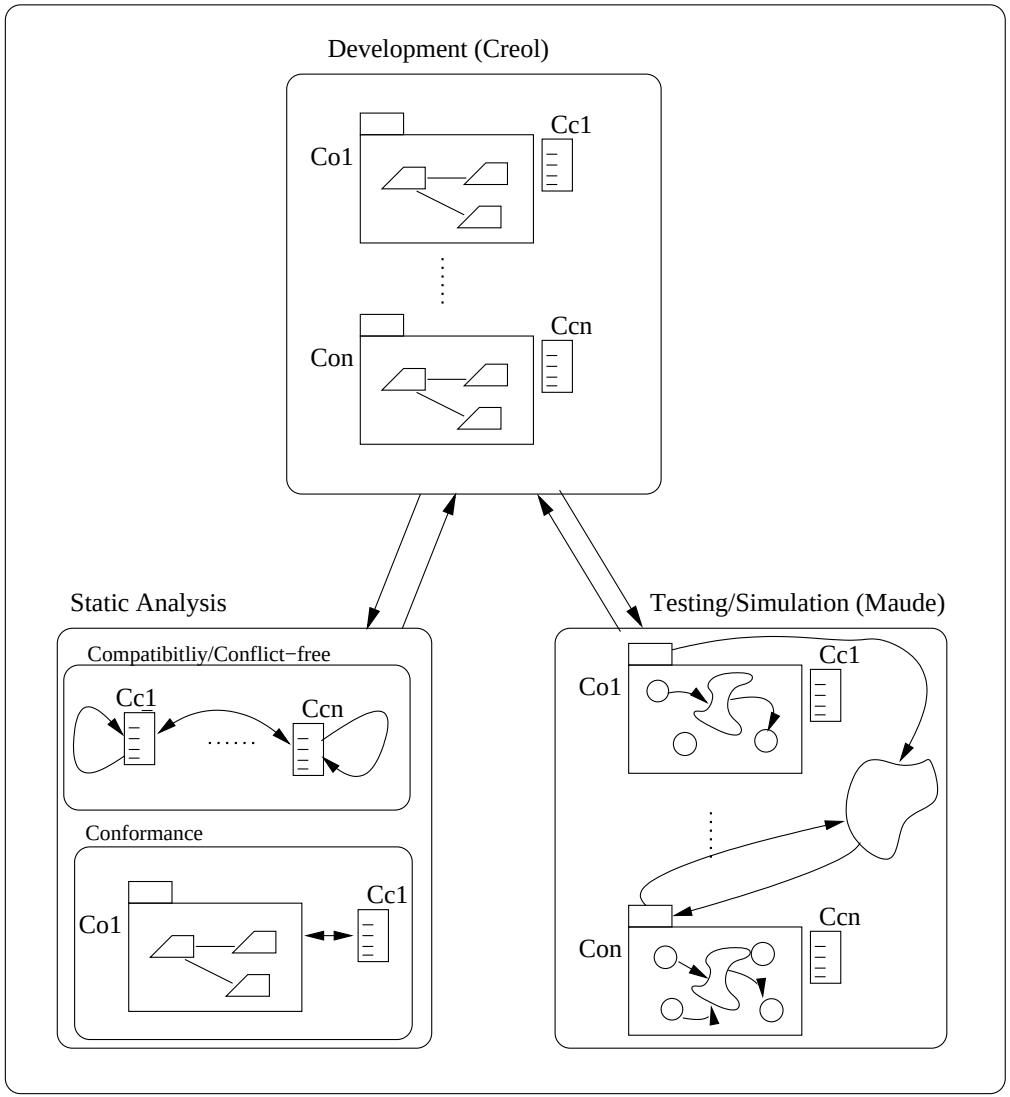
Creol: a concurrent object model

- executable oo modelling language concurrent objects
- formal semantics in rewriting logics /Maude
- strongly typed
- method invocations: synchronous or asynchronous
- recently: concurrent objects by (first-class) futures
- dynamic reprogramming : class definitions may *evolve at runtime*

Interfaces as types

- Object variables (pointers) are typed by interfaces
  (other variables are typed by data types)
- *Mutual dependency*: An interface may require a *cointerface*
  - Explicit keyword *caller*
  - Supports callbacks to the caller through the cointerface
  - Protocol-like behaviour
- Supports *strong typing*: no “method not understood” errors
- All object interaction is *controlled* by interfaces
  - *No explicit hiding* needed at the class level
  - Interfaces provide aspect-oriented specifications
  - A class may implement a number of interfaces

Contracts as behavioral interfaces



Contract specification language $\mathcal{CL}$

- formal specification language
- expressive enought to capture natural language contracts
  - contrary-to-duty (CTD)
  - contrary-to-permission (CTP)
- avoid certain paradoxes from deontic logic

Contract specification language $\mathcal{CL}$
- formal specification language
- expressive enought to capture natural language contracts
  - contrary-to-duty (CTD)
  - contrary-to-permission (CTP)
- avoid certain paradoxes from deontic logic

The “paradoxes” seem partly more like counter-intuitive conclusions in the interpretation of the formulas than. Counter-intuitive.

A glimpse of $\mathcal{CL}$

$$
\begin{array}{rcl}
\mathcal{C}\mathit{ontract} & := & \mathcal{D}\ ;\ \mathcal{C} \\
\mathcal{C} & := & \phi \mid \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\delta) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

- formal modal logic, combining aspects of
  - temporal,
  - deontic (O, P, F), and
  - dynamic logics
- formal semantics by translation into μ-calculus C_μ variant
- model checking using nuSMV
- sophisticated action algebra

A glimpse of $\mathcal{CL}$

$$
\begin{array}{rcl}
\mathcal{C}\text{ontract} & := & \mathcal{D}\ ;\ \mathcal{C} \\
\mathcal{C} & := & \phi \mid \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\delta) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

- formal modal logic, combining aspects of
  - temporal ,
  - deontic (O, P, F), and
  - dynamic logics
- formal semantics by translation into μ-calculus C_μ variant
- model checking using nuSMV
- sophisticated action algebra

The ϕ-thing is an assertion, i.e., a boolean formula.
Obligations, permissions, and prohibitions do not have an immediate truth value, but are more meant to specify restrictions. You may think of them as if they are wither respected or violated. Basically we can know about the truth value of e.g. $O(\alpha)$ obligation to do action α, only after the action has been done. For example, if another action different than α is executed (and not α which was obligatory) then one may conclude that the obligation was violated (the obligation doe not hold).
Permissions is treated rather special than obligation or prohibition.
α and δ are actions given in the definition part $\mathcal{D}$. $[\alpha]$ and $\langle\alpha\rangle$ are the action parameterised modalities of dynamic logic. $\mathcal{U}$, $\bigcirc$, and $\Box$ correspond to temporal logic operators. $\oplus$ is disjunction.

Conclusion & future work

- using Maude-engine for monitoring contracts
- conformance checking
- contracts-as-types

- FLACOS'07 – First Workshop on Formal Languages and Analysis of Contract-Oriented Software (in conjunction with NWPT'07): `http://www.ifi.uio.no/flacos07/`

Conclusion & future work

- using Maude-engine for monitoring contracts
- conformance checking
- contracts-as-types

- FLACOS'07 – First Workshop on Formal Languages and Analysis of Contract-Oriented Software (in conjunction with NWPT'07): `http://www.ifi.uio.no/flacos07/`

beginning of work

References I

[1] E. B. Johnsen, O. Owe, and I. C. Yu.
Creol: A type-safe object-oriented model for distributed concurrent systems.
Theoretical Computer Science, 365(1–2):23–66, Nov. 2006.

[2] Nordunet 3 project "Contract oriented software development for internet services.
`folk.uio.no/gerardo/nordunet3`, 2007.

[3] O. Owe, G. Schneider, and M. Steffen.
Components, objects, and contracts.
In *Sixths International Workshop on Specification and Verification of Component-Based Systems, Sept. 3–4, 2007, Catvat, Croatia*, Aug. 2007.

[4] O. Owe, G. Schneider, and M. Steffen.
Components, objects, and contracts.
Research Report 363, University of Oslo, Dept. of Computer Science, Aug. 2007.
A short version appeared in the proceedings of SAVCBS'07.

[5] C. Prisacariu and G. Schneider.
A formal language for electronic language.
In M. M. Bonsangue and E. B. Johnsen, editors, *FMOODS '07*, volume 4468 of *Lecture Notes in Computer Science*, pages 174–189. Springer-Verlag, June 2007.

[6] C. Prisacariu and G. Schneider.
Towards a formal definition of electronic contracts.
Research report 348, Department of Informatics, University of Oslo, Oslo, Norway, Jan. 2007.