University of Leicester

Model-based Development of Web Services using Design-by-Contract

Lesster

Reiko Heckel
University of Leicester, UK

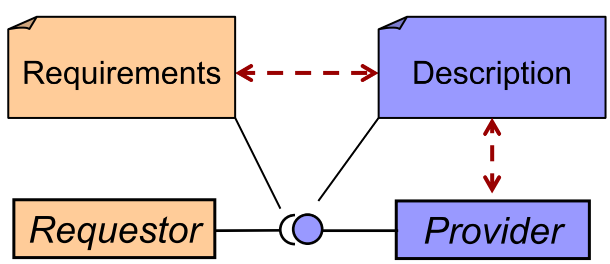
Joint work with M. Lohmann, A. Cherchago, J.H. Hausmann, Paderborn

TFS Colloqium, TU Berlin, 5. 12. 2005

1

University of Leicester

Consistency of Service Composition



1. External: between interface specifications
2. Internal: between interface specification and implementation

University of Leicester

Example: Car Rental Service

<<interface>>
RentalServiceRequired
reservCar(c:Customer, car:Car, ri:RentalInfo)
...

<<interface>>
RentalServiceProvided
makeReserv(c:Customer, car:Car, ri:RentalInfo): EContract
...

Matching *provider* and *requestor* specification within *registry* must ensure compatibility of

- Data types
  - Does Customer have the same meaning for requestor and provider?
- Operation signatures
  - Can provider operation be supplied with suitable parameters from a call of requestor operation?
- Behavior
  - Does provided operation actually carry out what is expected by a requestor?

University of Leicester

Data Types and Signatures

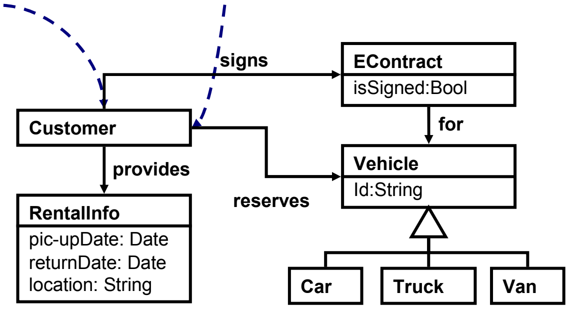
<<interface>>
RentalServiceRequired
reservCar(c:Customer, car:Car, ri:RentalInfo)
...

<<interface>>
RentalServiceProvided
makeReserv(c:Customer, car:Car, ri:RentalInfo): EContract
...

- Reorder and rename pars
- Skip input of requestor
- Ignore output of provider

Data types: parties use common domain model (ontology)

Operation signatures: Zaremski and Wing: *Signature matching: A tool for using software libraries.* TOSEM 1995.



University of Leicester

Behavior: Operation Contracts

Pre-condition:
Customer provides rental info and selects car

Effect:
Car is reserved for customer

Required

- Formal specification (logic, graph transformation, …) for automatic matching
- Integration into mainstream SW development methods (UML) for wider applicability

Outline

- Contracts as graph transformation rules
- Semantics of rules
- Semantic / syntactic compatibility, soundness

University of Leicester

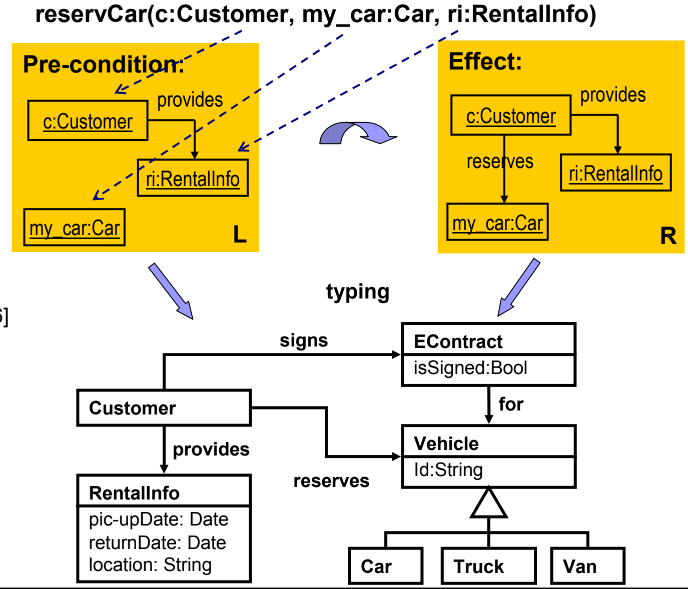
Contracts as Graph Transformation Rules

Signature: **reservCar(c:Customer, my_car:Car, ri:RentalInfo)**

Behavior:
GT rule

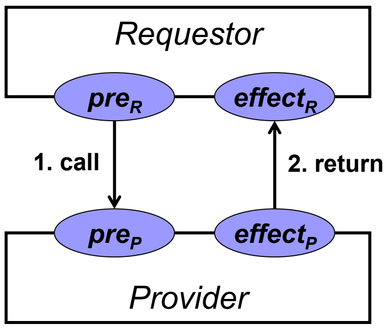
Typed DPO
[Corradini et al 96]

Data types:
type graph



University of Leicester

What is the right notion of compatibilty? That depends on …



how services should interact:

1. Requestor guarantees $\boldsymbol{pre_R}$
   → Provider assumes $\boldsymbol{pre_P}$
2. Provider guarantees $\boldsymbol{effect_P}$
   → Requestor assumes $\boldsymbol{effect_R}$

… a *contravariant* relation.

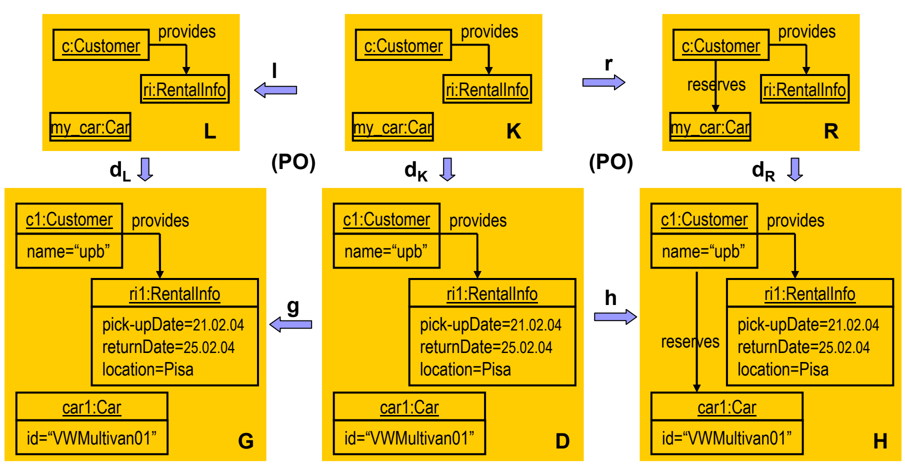
what it should mean, that:

- an *assumption* is correct
- a *guarantee* is fulfilled

… a question about the *semantics of contracts.*

University of Leicester

Operational Semantics: The DPO Approach



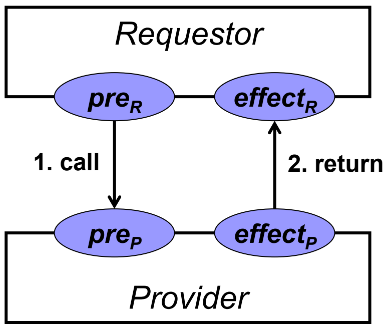
- *L* is embedded into graph ***G***.
- The elements of ***G*** matched by ***L- l(K)*** are removed.
- The elements matched by ***R - r(K)*** are added to *D*.

The changes to *G* are exactly those specified by the rule

University of Leicester

Loose Semantics of Contracts

Requestor has only loose idea of behavior of the other service



Provider has complete info, but may prefer not to publish everything

→ Contracts are incomplete specifications of service behavior

$$\begin{tikzcd} L \arrow[d, "d_L"'] \arrow[dr, phantom, "(PB)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "d_K"] \arrow[dr, phantom, "(PB)"] & R \arrow[d, "d_R"] \\ G & D \arrow[l, "g"] \arrow[r, "h"'] & H \end{tikzcd}$$

Formally: Double-Pullback (DPB), allows unspecified

- Deletion: *at least* elements of G matched by L - l(K) are removed
- Creation: *at least* elements matched by R - r(K) are added to D

(faithful) transition vs. transformation

University of Leicester

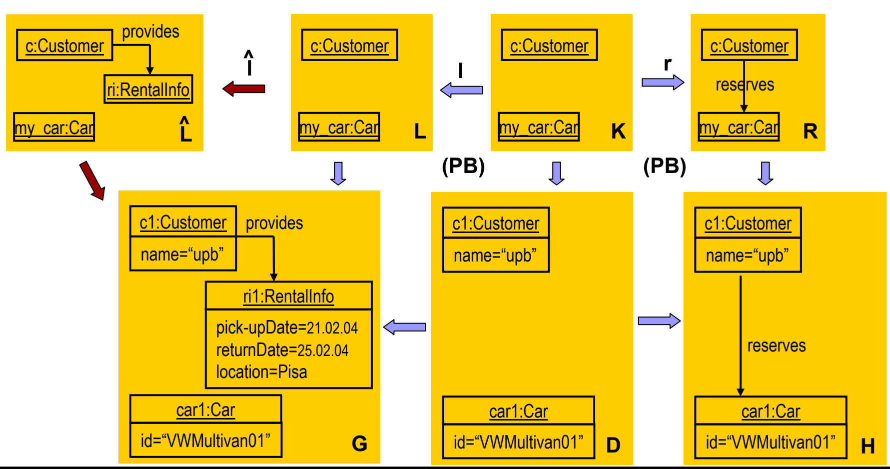
Contracts as Rules, revisited
→ Positive Application Conditions

Precondition: what must be

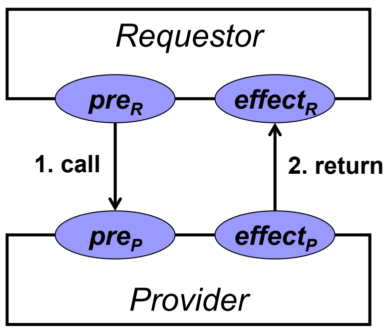
- present before, no matter what happens later

Effect: what must be

- deleted
- preserved
- created



What is the right notion of compatibility? That depends on …



how services should interact:

1. Requestor guarantees ***pre_R***
   → Provider assumes ***pre_P***
2. Provider guarantees ***$effect_P$***
   → Requestor assumes ***$effect_R$***

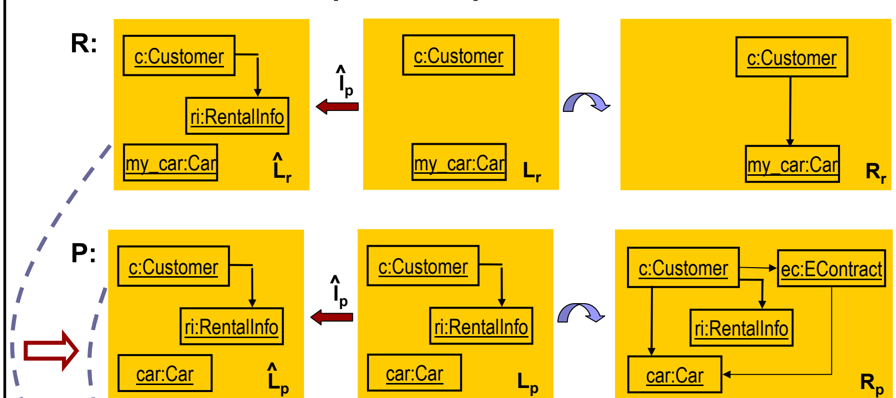
… a *contravariant* relation.

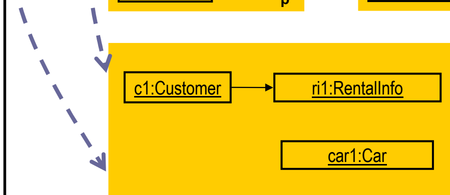
what it should mean, that:

- an *assumption* is correct
- a *guarantee* is fulfilled ✓

… a question about the *semantics of contracts.*

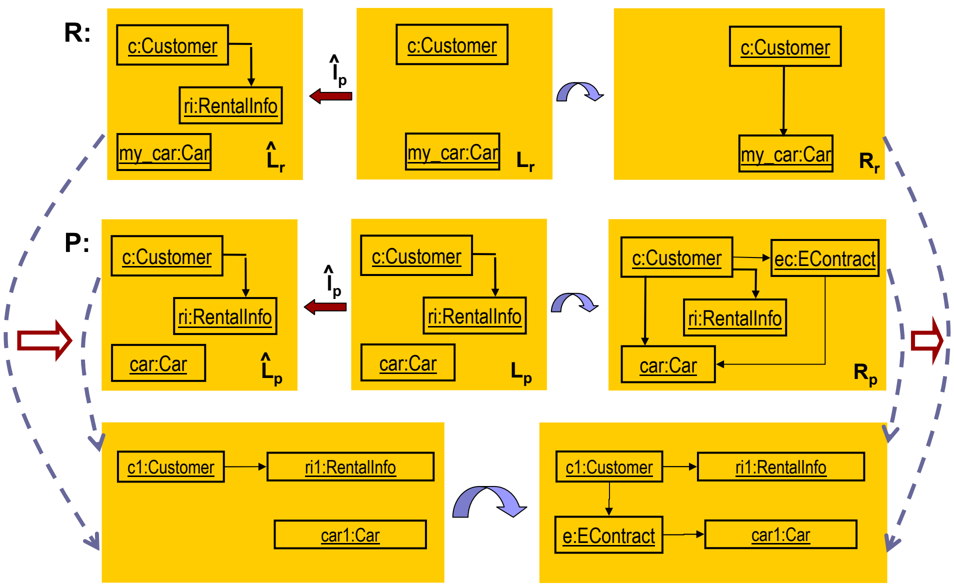
Semantic Compatibility



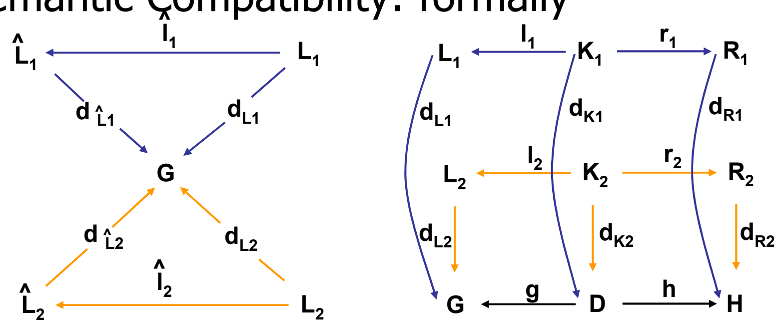


1. ***pre_R → pre_P :*** applicability of requestor rule **implies** applicability of provider rule
2. ***$effect_P$ → $effect_R$:*** transition via provider rule **is also** transition via requestor rule.

Semantic Compatibility



Semantic Compatibility: formally



$(p_1, \hat{L}_1)$ semantically matches $(p_2, \hat{L}_2)$, in symbols $(p_2, \hat{L}_2) \models_{match} (p_1, \hat{L}_1)$, iff:

(i) for all graphs G, if there exists $d_{\hat{L}_1} : \hat{L}_1 \to G$ s.t. $d_{L_1} := d_{\hat{L}_1} \circ \hat{l}_1$ satisfies IC of p_1, then there exists $d_{\hat{L}_2} : \hat{L}_2 \to G$ s.t. $d_{L_2} := d_{\hat{L}_2} \circ \hat{l}_1$ satisfies IC of p_2, and

(ii) for all spans $t : (G \xleftarrow{g} D \xrightarrow{h} H)$, if there exists a transition $G \overset{p_2/d_2}{\rightsquigarrow} H$, then there exists a transition $G \overset{p_1/d_1}{\rightsquigarrow} H$ using the same bottom span t.

University of Leicester

What do we have?

Semantic compatibility relation $|=$

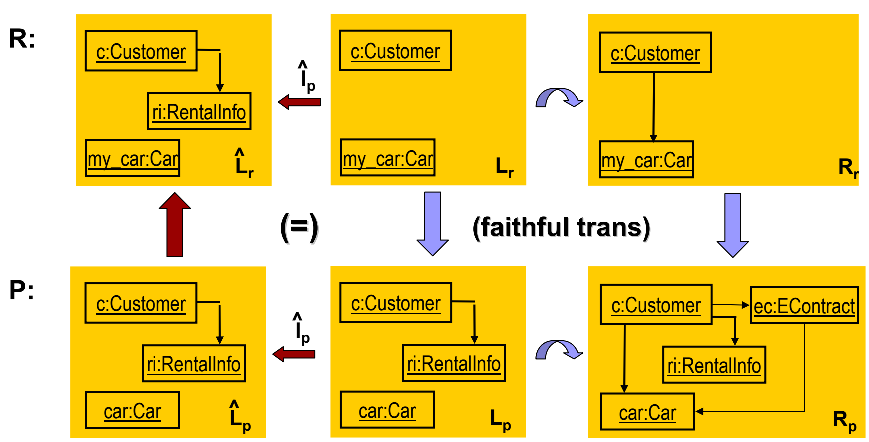
- quantified over all graphs and transitions
- cannot be verified directly

Objective: syntactic matching relation $|--$

- Soundness: p_2 $|--$ p_1 *implies* p_2 $|=$ p_1
- Completeness: p_2 $|=$ p_1 *implies* p_2 $|--$ p_1

University of Leicester

Syntactic Matching Relation



pre$_R$ → pre$_P$: requestor must provide all information necessary for the execution of the provider operation,

effect$_P$ → effect$_R$: effect of the provided operation must include those expected by the requestor.

University of Leicester

Syntactic Matching: formally

$$\begin{tikzcd} \hat{L}_1 & L_1 \arrow[l, "\hat{l}_1"'] \\ \hat{L}_2 \arrow[u, "h_{\hat{L}}"] \arrow[r, phantom, "(=)"] & L_2 \arrow[l, "\hat{l}_2"] \arrow[u, "h_L"'] \end{tikzcd}$$

faithful transition

$$\begin{tikzcd} L_1 \arrow[d, "h_L"'] \arrow[r, phantom, "(PB)"] & K_1 \arrow[l, "l_1"'] \arrow[r, "r_1"] \arrow[d, "h_K"] \arrow[r, phantom, "(PB)"'] & R_1 \arrow[d, "h_R"] \\ L_2 & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] & R_2 \\ G & D \arrow[l, "g"'] \arrow[r, "h"] & H \end{tikzcd}$$

$(p_1, \hat{L}_1)$ syntactically matches $(p_2, \hat{L}_2)$, in symbols $(p_2 : s_2, \hat{L}_2) \vdash_{match} (p_1 : s_1, \hat{L}_1)$, iff:

(i) there exists an injective graph homomorphism $h_{\hat{L}} : \hat{L}_2 \to \hat{L}_1$ s.t. $h_{\hat{L}} \circ \hat{l}_2$ satisfies IC of p_2, and

(ii) there exist graph homomorphisms $h_L : L_1 \to L_2$, $h_K : K_1 \to K_2$, and $h_R : R_1 \to R_2$ s.t. the diagrams (a), (b), and the diagram on the left commute, and the diagrams (a) and (b) represent a faithful transition.

University of Leicester

What do we have?

✓ Semantic compatibility: relation |=

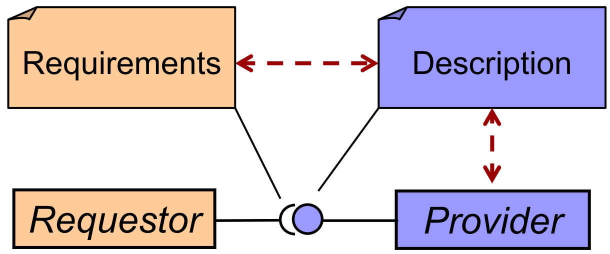
✓ Syntactic matching: relation |--

✓ Soundness: ***p*₂ |-- *p*₁** *implies* ***p*₂ |= *p*₁**

✓ Completeness: ***p*₂ |= *p*₁** *implies* ***p*₂ |-- *p*₁**

University of Leicester

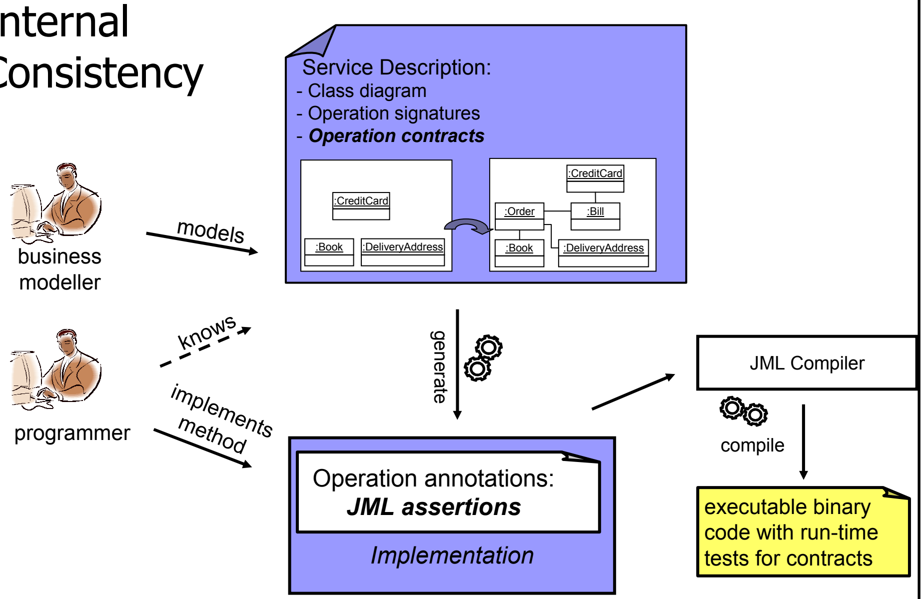
Consistency of Service Composition



- ✓ External: between interface specifications
- 2. Internal: between interface specification and implementation

University of Leicester

Internal Consistency



JML from Graphical Contracts

Semantic idea: Assume rule ***r*** specifying method ***m*** .

- If ***r*** is applicable to ***G*** , then ***m*** invoked in ***G*** (with appropriate parameters) terminates without exception.
- If invocation yields ***H***, there exists a graph transition from ***G*** to H via ***r***.

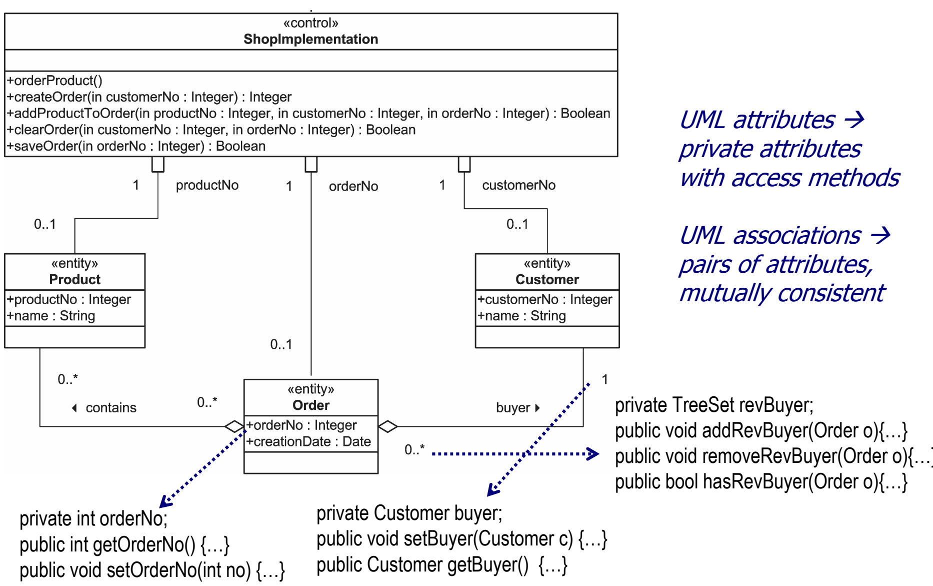
After manually refining the models (business → analysis view), translate

1. class diagram → Java class frames
2. rules → JML
   - patterns
   - rules

Class diagrams → Java class frames

UML attributes → private attributes with access methods

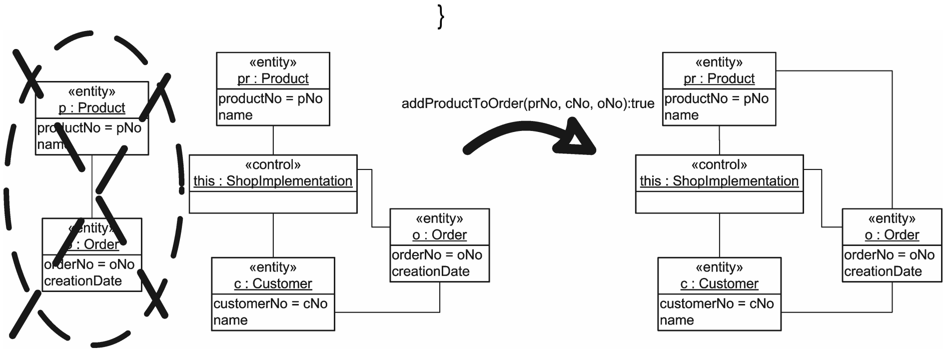
UML associations → pairs of attributes, mutually consistent



University of Leicester

Contracts → JML

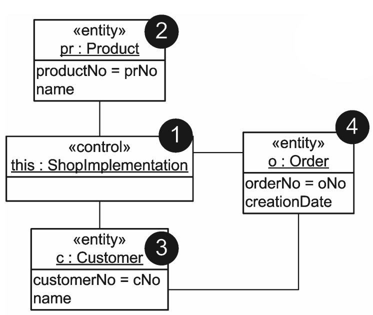
```
public class ShopImplementation {
        …
/*          @ public normal_behavior
            @ requires JML-PRE;
            @ ensures JML-POST;
*/
public boolean addProductToOrder(
    int productNo, int customerNo, int orderNo) {…}
        …
}
```



University of Leicester

Contracts → JML: Patterns

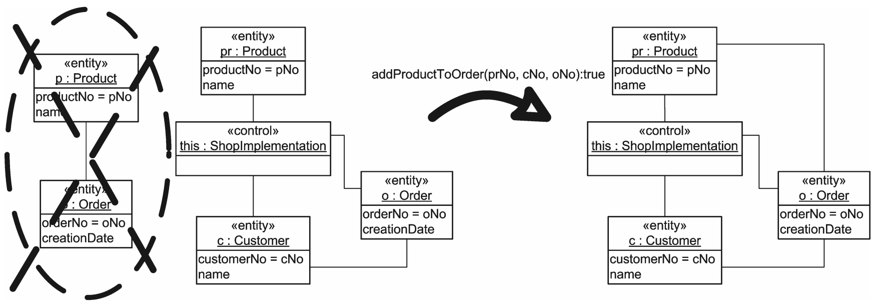
- starting at *this* navigate to as yet unbound objects, check attributes and links and bind them
- select navigation paths to achieve earliest possible failure




```
@ public normal_behavior
@ requires (\exists Product p;
@           product.contains(p);
@           p.getNo() == productNo);
@           && (\exists Customer c;
@           customer.contains(c);
@           c.getNo() == customerNo
@       && (\exists Order o;
@       order.contains(o);
@       o.getOrderNo() == orderNo
@       && o.getCustomer() == c
@       && o.containsProduct(p) == false)));
```

University of Leicester

Contracts → JML: Rules



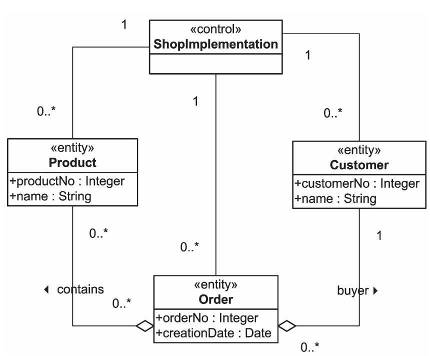
- like let, evaluated in pre state
- works for deterministic matches

```
@ public normal_behavior
@
@ old Product p = getProductByNo(productNo);
@ old Customer c = getCustomerByNo(customerNo);
@ old Order o = c.findOrderByNo(orderNo);
@
@ requires p != null;
@ requires c != null;
@ requires o != null;
@ requires o.getCustomer() == c
@       && o.containsProduct(p) == false;
```

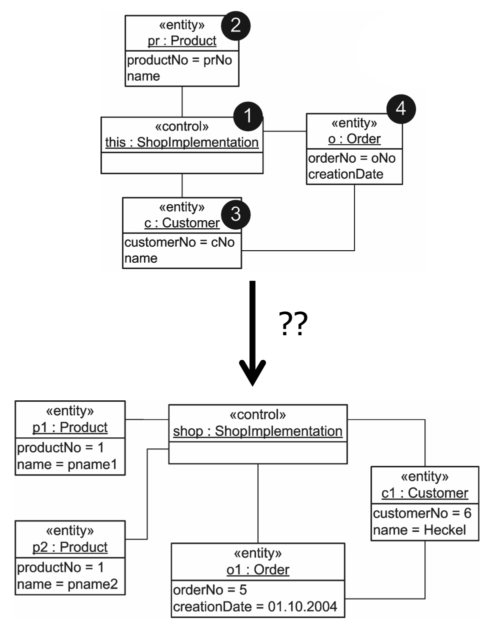
```
@ ensures p != null;
@ ensures c != null;
@ ensures o != null;
@ ensures \not_modified(p, c);
@ ensures o.getCustomer() == c;
@ ensures o.getProducts().contains(p);
```

University of Leicester

Non-deterministic Matching



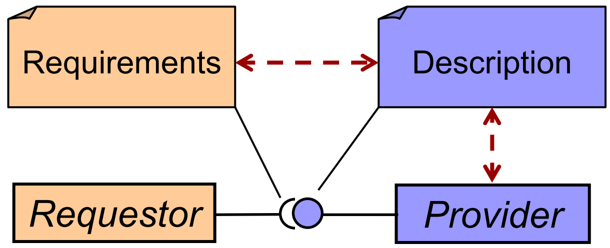
alternative class diagram



Solution: store all possible bindings and check that at least on satisfies post-condition

University of Leicester

Consistency of Service Composition



Visual representation of contracts based on GT with loose semantics
- External: syntactic characterization of service compatibility
- Internal: mapping of contracts to JML

Open questions
- relation between business-level and analysis-level contracts
- verification of mapping GT → JML
- implementation and evaluation

University of Leicester

Papers

With A. Cherchago, M. Lohmann: *A Formal Approach to Service Specification and Matching based on Conditional Graph Transformation*, ICGT 2004 in Rome

With M. Lohmann: *Model-Driven Development of Reactive Information Systems: From Graph Transformation Rules to JML Contracts, to appear in STTT*