The claim

“A component is specified in terms of the interfaces it provides and the interfaces it requires.”

This seems fine, but it raises a question:

Should information about what a component requires form part of its contract with its clients?

No!

And this means we need to think about component specification in a completely new way...
The claim in pictures

“A component is specified in terms of the interfaces it provides and the interfaces it requires.”

The details

- A complete specification must contain sufficient information for the component to be built and used.
- What shall we add? How about a sequence diagram?
Interface specification

We now have these interfaces:

<table>
<thead>
<tr>
<th>IOrderMgt</th>
</tr>
</thead>
<tbody>
<tr>
<td>placeOrder(custNum, prodNum, quan)</td>
</tr>
<tr>
<td>numOfOrders(custNum): Integer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IProductMgt</th>
</tr>
</thead>
<tbody>
<tr>
<td>reserveStock(prodNum, quan)</td>
</tr>
<tr>
<td>availableStock(prodNum): Integer</td>
</tr>
</tbody>
</table>

We could specify `placeOrder()` like this:

“The number of orders for the customer is increased by one and a `reserveStock` message is sent to the component supporting the `IProductMgt` interface”

But...

- The client of a component is not supposed to depend on the nature of its implementation
- An interface can be supported by many different components, and each component can:
  - make its own implementation choices and
  - be subject to different implementation constraints
- The specification of `placeOrder()` contains information about the required implementation of the operation
- Therefore: We need to separate the needs of the client from the needs of the implementer
Two distinct contracts

Usage contract: a contract between a component object’s interface and a client

Realization contract: a contract between a component specification and a component implementation

Contracts and roles

Specifier (Architect):
A person who produces the technical specification for a system or components within a system

Realizer:
A person who builds a component that meets a component specification

Client:
A person who writes software that uses a component
Separation of specification concerns

"The number of orders for the customer is increased by one and a reserveStock message is sent to the component supporting the IProductMgt interface."

The client cares about this - it affects the subsequent result of numOfOrders(). Therefore it is part of the usage contract.

The client does not care about this - but the implementer does. Therefore it is part of the realization contract.

Component Concepts

Component Interface

* supportedInterface

Component Specification

1 realization

Component Implementation

1 installation

Installed Component

1 instance

Component Object

John Daniels - Component Contracts

5
Interfaces vs Component Specs

- Represents the usage contract
- Provides a list of operations
- Defines an underlying logical information model specific to the interface
- Specifies how operations affect or rely on the information model
- Describes local effects only

- Represents the realization contract
- Provides a list of supported interfaces
- Defines the run-time unit
- Defines the relationships between the information models of different interfaces
- Specifies how operations should be implemented in terms of usage of other interfaces

Specifying the usage contract

<table>
<thead>
<tr>
<th>placeOrder(custNum, prodNum, quan)</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: Integer</td>
<td></td>
</tr>
<tr>
<td>orderCount: Integer</td>
<td></td>
</tr>
</tbody>
</table>

context IOrderMgt::placeOrder(custNum, prodNum, quan)
pre:   -- custNum and prodNum are valid ids
post:   -- the orderCount of Customer with id=custNum has increased by 1

context IOrderMgt::numOfOrders(custNum): Integer
pre:   -- custNum is a valid customer id
post:   -- result = the orderCount of the Customer with id=custNum
Specifying the realization contract

- We can use interaction diagrams, one or more per operation, focused on the component being specified
- These specify how the operations must be implemented

A more complex example

Architectural decision: Only the ProductManager component holds stock levels
Specifying the usage contract

<table>
<thead>
<tr>
<th>IOrderMgt</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Customer</td>
</tr>
<tr>
<td>id: Integer</td>
</tr>
<tr>
<td>orderCount: Integer</td>
</tr>
<tr>
<td>* Product</td>
</tr>
<tr>
<td>id: Integer</td>
</tr>
<tr>
<td>stockQuan: Integer</td>
</tr>
</tbody>
</table>

context IOrderMgt::availableStock(prodNum): Integer
pre: -- prodNum is a valid product id
post: -- result = the stockQuan of the Product with id=prodNum

We extend the information model to describe products, even though the OrderManager component is not responsible for holding stock levels.

Specifying the realization contract

- We want to specify that the OrderManager gets stock information from the ProductManager
- We could provide an interaction diagram for availableStock(), but that might be overspecification
- Instead, we can constrain the relationship between the information models of the two interfaces. This allows a variety of implementations
Want to know more?

- Forthcoming book by John Cheesman and John Daniels, Addison-Wesley, October 2000
- Tutorial tomorrow!
- Tutorial and workshop at OOPSLA 2000, Minneapolis, October 2000