Working with objects: 
*Designing Distributed Systems for Reuse*

Trygve Reenskaug, Odd Arild Lehne  
©Taskon 1997

*OOPSLA '97 tutorial # 26*  
*Atlanta, Georgia, 5 October 1997*
1. Working with objects

Designing Distributed Systems for Reuse

OOPSLA Tutorial # 26

Trygve Reenskaug, Odd Arild Lehne
Taskon, Oslo

odda@taskon.no
trygve@taskon.no

1.1 Motto:

There are two ways of constructing a software design:

- One way is to make it so simple that there are obviously no deficiencies
- and the other way is to make it so complicated that there are no obvious deficiencies.

-- C. A. R. Hoare
1.2
Three dimensions of software development

1.3
Incidental and planned reuse

Incidental reuse (ad hoc):
- search existing solutions for applicable ideas, models, code, ...

Planned reuse (product development):
- Understand the business
- Understand the developers
- Understand applicable solutions
- Create reusable designs and code
1.4
Reusable component lifecycle

Build experience
Application Programming, Isolated Applications:
- analyze
- design
- implement

Capture experience
Production Engineering:
Create Pool of Reusable Assets

Reuse experience
Application Programming, Family of Applications:
- analyze
- design
- implement

Role Modeling technology
2. Role Modeling
   Focus On Object Collaboration

2.1 Four aspects of OO modeling

- **Object Oriented Modeling**
  - **Role modeling**
    - The role is the why abstraction
    - *Why do we have this object?*
  - **Class modeling**
    - The class is the how abstraction
    - *How is this object implemented?*
  - **Interface modeling**
    - The interface is a what abstraction
    - *What does this object look like?*
  - **Data modeling**
    - The entity is another what abstraction
    - *What does it mean?*

---

Build experience

Application Programming, Isolated Applications:
- analyze
- design
- implement

Capture experience

Production Engineering: Create Pool of Reusable Assets

Reuse experience

Application Programming, Family of Applications:
- analyze
- design
- implement

Role Modeling technology
2.1.1 Simple examples

```java
interface Number {
    Number add (Number num);
    Number subtract (Number num);
    Number multiply (Number num);
    Number divide (Number num);
}
```

2.2 Main features of role modeling

- We only show relevant
  -- objects
  -- aspects
  -- details

- We abstract object identity to its position in the pattern
2.2.1
Model inheritance: *Specialization*

Base model: Fraction

- Fraction
- Numerator
- Denominator

Derived model: Ohms law

- Voltage
- Current
- Resistance

2.2.2
Separation of concern: *Three uses of synthesis*

1. Specialization -- generalization

2. Composition on same level of abstraction

3. Aggregation
2.2.3 Composition on same level of abstraction

Example
\[
\frac{4}{\frac{7}{8}}
\]

2.2.4 Aggregation

Example
\[
\frac{4}{\frac{7}{8}}
\]
2.2.5
Important observations:

- A model is created for a purpose.
- A model is never complete.
- We think in multiple models, always trying to choose the best model for our purpose.
- We tend to think in hierarchical models, even though the world is rarely hierarchical.

2.3
Role Model Advantages

- Role models describe object patterns
- Modeling distribution
- Separation of Concern
- Reuse through model inheritance
- Seamless bridge to implementation
2.4
Role models describe object patterns

2.4.1
Role Model Collaboration view notation

Working with objects:
Designing Distributed Systems for Reuse

©Taskon 1997. Page 9
2.4.2
Compare with class hierarchy
(VisualWorks class library)

<table>
<thead>
<tr>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>- ControllerWithMenu</td>
</tr>
<tr>
<td>- ListController</td>
</tr>
<tr>
<td>- StandardSystemController</td>
</tr>
<tr>
<td>. . .</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Browser</td>
</tr>
<tr>
<td>- Debuger</td>
</tr>
<tr>
<td>- Inspector</td>
</tr>
<tr>
<td>- ContextInspector</td>
</tr>
<tr>
<td>- DictionaryInspector</td>
</tr>
<tr>
<td>- PopUpMenu</td>
</tr>
<tr>
<td>. . .</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>- AutoScrollingView</td>
</tr>
<tr>
<td>- ListView</td>
</tr>
<tr>
<td>- FractionalWidgetView</td>
</tr>
<tr>
<td>- LauncherView</td>
</tr>
<tr>
<td>- Scrollbar</td>
</tr>
<tr>
<td>. . .</td>
</tr>
</tbody>
</table>

2.4.3
Environment  Collaboration view

Port (messages sent)

User  System

Environment role

Virtual role
(hides many)
2.4.4
Scenario view

User
Controller
View
Model

modifyModel
modify()
update()
getInformation()
present()

Interaction sequence

2.4.5
Role modeling
focuses on object patterns

Object A

Port
IN
OUT-B
OUT-C
Methods
Variables

Object B

Port
IN
OUT-C
Methods
Variables

Object C

Port
IN
Methods
Variables

Identity
Encapsulation
Polymorphism

Message interaction
2.5
Modeling distribution

(a) Two-tier, Fat Server

(b) Two-tier, Fat Client

(b) Four-tier, Tool and Service

2.5.1
Determine distribution architecture

Select possible communication architecture
- Evaluate communication requirements
- Choose communication method
  - e.g., on demand
  - e.g., caching
- Estimate communication costs (time, money)

Implement solution
- Subclass frameworks
- Specialize code
- Optimize code

Iterate
2.6 Separation of Concern
A TextView object plays many roles

- **VisualPart role**
- **View role**
- **TextView role**

Windowing role model
MVC role model
TextEditor role model

class TextView instance

2.7 Reuse through model inheritance

Inherit static properties:
- Semantics
- Responsibility
- Attributes
- Base classes

Inherit dynamic properties:
- Use cases
- Message flows
- Data flows
- Finite State Machines
2.8
Seamless bridge to implementation

interface ViewController {
    text select ();  }

interface ViewModel {
    void update ();  
    // Model has changed, refresh
}

abstract class View implements ViewController, ViewModel {
    ...

2.8.1
Method views

Inside object perspective

MVC

update()

/* Retrieve contents from model. */
/* Redisplay contents. */
/* abstract method */

TextEditor

update()

/* Retrieve text from model. */
/* Redisplay text. */

text.displayOn ()

getText()
presentToUser

User

Text Store

Text View
2.8.2
Mapping concepts from OOram to implementation

Roles
--> Interfaces
  --> Classes

Ports
--> Member variables & other references
  --> Interfaces

Messages
--> Methods

Model inheritance
--> Class ensemble inheritance

2.8.3
Program code
Class perspective

abstract class View implements ViewController, ViewModel {
    Model model;
    ControllerView controller;
    abstract void select (···) ;
    abstract void update (···) ;
    // Model has changed, refresh
    ...
}

class TextView extends View implements TextViewController TextViewText {
    Text textCache;
    void update () {
        //Retrieve text from model.
        textCache = model.getText();
        //Redisplay text.
        this.display();
    }
}

Working with objects:
Designing Distributed Systems for Reuse
©Taskon 1997. Page 15
2.8.4
Semantic model - Object System

Object System

Object

Role Model

Role (Why)

consists of

responds to and uses

implemented by

Program

implements

consists of

Interface (What)

uses

implemented by

Object System

composed_of

composed_of

©Taskon 1997. Page 16
Working with objects:
Designing Distributed Systems for Reuse
3.
The first isolated application
Card access control system (CAC)

Role Modeling technology

3.1
An Access Control problem
Area of concern

An office complex has a number of office areas separated by automatic doors.

We want to limit the access of different persons through a system of machine-readable identity cards.
3.2
User interfaces
Prototype Program

3.3
Process Steps

- Define Area Of Concern
- Describe system seen from environment
- Consider system architecture
- Consider separation of concern
  For each sub-system:
  - Describe system seen from environment
  - Describe system roles and collaborations
- Synthesize system from sub-systems
3.3.1
System seen from environment
Environment collaboration view

3.3.2
System seen from environment
Stimulus-Response view

<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person.identification (from);</td>
<td>Door.open ();</td>
<td>Person accepted</td>
</tr>
<tr>
<td></td>
<td>// time delay</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Door.close ();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Person.reject ();</td>
<td>Person rejected</td>
</tr>
<tr>
<td></td>
<td>AlarmHandler .doorOpenAlarm();</td>
<td>Door left open</td>
</tr>
<tr>
<td>SystemManager .addPersonWithAccess ();</td>
<td>Access data base updated</td>
<td></td>
</tr>
</tbody>
</table>
3.3.3
System Architecture
A Choice

3.3.4
Separation of Concern

<table>
<thead>
<tr>
<th>On activities</th>
<th>Management</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role modeling</td>
<td>Person</td>
<td>Door</td>
</tr>
<tr>
<td></td>
<td>Local Station</td>
<td>Central station</td>
</tr>
<tr>
<td></td>
<td>System Manager</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On objects (Data modeling)</th>
<th>Local Objects</th>
<th>Central Objects</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>On classes (Class modeling)</th>
<th>Object</th>
<th>Person</th>
<th>Card Reader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Device</td>
<td>Input</td>
<td>Keyboard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>Display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input/Output</td>
<td>Door</td>
</tr>
</tbody>
</table>
3.3.5
Consider Separation of concern

Hints for separation of concern:
- Separate on stimulus objects
- Separate on stimulus messages

<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person.identification (from);</td>
<td>Door.open ();</td>
<td>Person accepted</td>
</tr>
<tr>
<td></td>
<td>// time delay</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Door.close();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Person.reject ();</td>
<td>Person rejected</td>
</tr>
<tr>
<td></td>
<td>AlarmHandler .doorOpenAlarm();</td>
<td>Door left open</td>
</tr>
<tr>
<td>SystemManager .addPersonWithAccess (;)</td>
<td>Access data base updated</td>
<td></td>
</tr>
</tbody>
</table>

3.3.6
Revised plan for system development

Card only access control system (CA)
- System seen from environment
- Separation of concern

Validation subsystem (VAL)
- Environment
- Inter object views
- Inside object views

Management subsystem (MS)
- Environment
- Inter object views
- Inside object views

synthesis

Card only access control system (CA)
- System Implementation
3.4
Validation subsystem (VAL)

Card only access control system (CA)
- System seen from environment
- Separation of concern

Validation subsystem (VAL)
- Environment
- Inter object views
- Inside object views

Management subsystem (MS)
- Environment
- Inter object views
- Inside object views

synthesis

Card only access control system (CA)
- System Implementation

3.4.1
Validation Environment
3.4.2
VAL Normal access scenario

3.4.3
VAL Open door alarm scenario
### 3.4.4 Validation sub-system
**Controller details**

![Diagram showing the connection between CardReader, Controller, Delay class, and Validator.]

### 3.4.5 Validation
**Normal access scenario**
**Controller details**

![Diagram showing the sequence of operations including CardReader, Door, Controller, Delay, Delay class, and Validator.]

- CardReader
- Door
- Controller
- Delay
- Delay class
- Validator

Operations:
- check()
- validate()
- openDoor()
- forSeconds()
- wait()
- closeDoor()
### 3.4.6
Inside object perspective
*Controller state diagram*

![Controller state diagram](image)

### 3.4.7
Inside object perspective
*A Controller Method*

![Controller method diagram](image)
3.5
Management subsystem (MS)

Card only access control system (CA)
- System seen from environment
- Separation of concern

Validation subsystem (VAL)
- Environment
- Inter object views
- Inside object views

Management subsystem (MS)
- Environment
- Inter object views
- Inside object views

3.5.1
Area of Concern
MS Design Model

An office complex has a number of office areas separated by automatic doors.

We shall design a subsystem that permits a SystemManager to manage people's access privileges
3.5.2
Describe object structure
*MS Design Model*

<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemManager</td>
<td>Access data base</td>
</tr>
<tr>
<td>. addPersonWithAccess</td>
<td>updated</td>
</tr>
</tbody>
</table>

3.6
Card access control system (CA)

Card only access control system (CA)
- System seen from environment
- Separation of concern

Validation subsystem (VAL)
- Environment
- Inter object views
- Inside object views

Management subsystem (MS)
- Environment
- Inter object views
- Inside object views

*synthesis*

Card only access control system (CA)
- System Implementation
3.6.1
Synthesize from subsystems
\textit{CA Design model}

\begin{itemize}
  \item Synthesis not required
  \item Implement single class for Validator+Management roles
\end{itemize}

3.6.2
The Validator type satisfies
\textit{COValidator and ManagementSubsystem Roles}

\begin{itemize}
  \item Synthesis not required
  \item Implement single class for Validator+Management roles
\end{itemize}
3.7

Summary
First isolated application

Card only access control system (CA)
- System seen from environment
- Separation of concern

Validation subsystem (VAL)
- Environment
- Inter object views
- Inside object views

Management subsystem (MS)
- Environment
- Inter object views
- Inside object views

Card only access control system (CA)
- System Implementation

synthesis
4. The second isolated application

*Keyboard only access control (KO)*

---

4.1 Area of concern

*Keyboard only access control (KO)*

An office complex has a number of office areas separated by automatic doors.

We want to limit the access of different persons through a **Keyboard Secret Code system**.
4.2
Incidental reuse

search existing solutions for applicable ideas, models, code, ...

4.3
Stimulus-Response view

*Keyboard only access control (KO)*

<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person.identification (from);</td>
<td>Door.open (); // time delay Door.close();</td>
<td>Person accepted</td>
</tr>
<tr>
<td></td>
<td>Person.reject ();</td>
<td>Person rejected</td>
</tr>
<tr>
<td></td>
<td>AlarmHandler .doorOpenAlarm();</td>
<td>Door left open</td>
</tr>
<tr>
<td>SystemManager .addPersonWithAccess ();</td>
<td>Access data base updated</td>
<td></td>
</tr>
</tbody>
</table>
4.4
Collaboration View

Keyboard only access control (KO)

Different Device

Different Data

Different Test

4.5
Normal Access Scenario View

Keyboard only access control (KO)

Working with objects:
Designing Distributed Systems for Reuse
5. Creating reusable components

5.1 Large systems from small projects
5.2
The Taskon Fountain Model for Planned Reuse

HAPPY END USER

Forward engineering

Reverse engineering

Application programming

Production engineering

Pool of Reusable assets

5.3
Some reuse terminology - 1

Alexander:
A pattern is a description of a problem with hints for its solution ensuring coherence and humanity

Some OO methodologists:
An object pattern is a structure of interacting objects
5.4
Some reuse terminology - 2

**OOram Pattern:**
An OOram pattern is a description of a problem with a Role Model describing its solution

**OOram Framework:**
A canned solution with role model + base classes for subclassing

5.5
Creating an OOram framework

- Identify consumers and consumer needs
- Perform cost-benefit analysis
- Perform reverse engineering on existing applications
- Create new framework(s)
- Inform user community
5.6 Identify reusable activities: CO Scenario view

![Diagram showing CO Scenario view]

5.7 Isolate reusable parts Inverse synthesis

![Diagram showing reusable parts and inverse synthesis]

What's left

![Diagram showing remaining components]

5.8
Object Management Group (OMG):
Event Service (simplified)

**CORBA Event Channel**

5.8.1 Establish connections

---

Working with objects:
Designing Distributed Systems for Reuse

©Taskon 1997. Page 37
5.8.2
Supplier Changed

5.9
Access Control Frameworks

Compose specific product
5.10
Key LocalStation control method

void identification (String identString, TypeSymbol typeSymbol) {
    /* This is a generic method that collects all identifications
        in Dictionary, passing it on for validation when complete. */
    Boolean accessOK;
    identDict.add (typeSymbol, identString);
    if ((identDict.size = this.noOfInputs()) == true) {
        accessOK = validator.validate(identDict);
        identDict = new IdentityDictionary(); // Clear
        if (accessOK) this.openAndCloseDoor();
    }
}
6.1 Area of Concern

**CK Requirements**

An office complex has a number of office areas separated by automatic doors.

We shall design a System that controls access to the protected Zones
- identified by card during working hours
- identified by card and key code at other times
6.2  
User interfaces
CK Prototype Program

6.3  
Synthesize application
CK Design

Card reader device

Keyboard device

Voice Recognition

etc. ...

Comm. frameworks

Small System Management

Large System Management

High Security Management

Access Control Logic with Door Control

Compose specific product

Specialize

Work with objects:
Designing Distributed Systems for Reuse

©Taskon 1997. Page 41
6.3.1
Synthesized Card/Key model

6.3.2
Determine distribution architecture

Select possible communication architecture
- Evaluate communication requirements
- Choose communication method
  - e.g., on demand
  - e.g., caching
- Estimate communication costs (time, money)

Implement solution
- Subclass frameworks
- Specialize code
- Optimize code
6.3.3 Some possible distributions

Extreme centralization

Decentralized

Replicated database

6.3.4 Example: Replicated database
Controller, initialization and checking

class Manager {
    /* Initialize structure. */
    manager (Manager aManager) {
        super.manager (aManager);
        acr = new ACCardReader();
        acr.openInterface("windowSpec", this);
        akr = new ACKeyboardReader();
        akr.openInterface("windowSpec", this);
        inputDevices = new OrderedCollection (acr, akr);
    }
}

void identification (IdentSpec identString, TypeSymbol typeSymbol) {
    /* Collect identifications in Dictionary, validate when complete. */
    Boolean accessOK;
    identDict (typeSymbol, identString);
    if (((Time.hours > 8) and (Time.hours <= 16)
        and (typeSymbol == "CardReader"))
        or (identDict size() == 2)) {
        accessOK = manager.validate(identDict);
        if (accessOK) {
            this.openAndCloseDoor();
        }
    }
}
}
7. Conclusion

- Role models describe object patterns
- Modeling distribution
- Separation of Concern
- Reuse through model inheritance
- Seamless bridge to implementation

7.1 Role models describe object patterns
7.2 Separation of Concern

Object pattern composition

- VisualPart role
- View role
- TextView role

Windowing role model
MVC role model
TextEditor role model

class TextView instance

7.3 Modeling distribution

(a) Two-tier, Fat Server

(b) Two-tier, Fat Client

(b) Four-tier, Tool and Service
7.4
Reuse through model inheritance

7.5
Seamless bridge to implementation

Roles
--> Interfaces
--> Classes

Ports
--> Member variables & other references
--> Interfaces

Messages
--> Methods

Model inheritance
--> Class ensemble inheritance
7.6
More information

Management processes:
Goldberg and Rubin: "Succeeding with objects".

Modeling method:
T. Reenskaug, P. Wold, O.A. Lehne: "Working With Objects"
Manning/Prentice Hall 1996.
ISBN 0-13-452930-8

Patterns:
Gamma, Helm, Johnson, Vlissides: "Design Patterns".

Distribution:
ODP: Open Distributed Processing

OOram tools can be downloaded from:
http://www.sn.no/taskon/
TABLE OF CONTENTS

1. Working with objects ........................................................................................................... 1
  1.1 Motto: .................................................................................................................................. 1
  1.2 Three dimensions ................................................................................................................. 2
    of software development
  1.3 Incidental and planned reuse .............................................................................................. 2
  1.4 Reusable component lifecycle ............................................................................................ 3

2. Role Modeling .......................................................................................................................... 4
   Focus On Object Collaboration
   2.1 Four aspects of OO modeling ............................................................................................. 4
     2.1.1 Simple examples .......................................................................................................... 5
   2.2 Main features of role modeling ............................................................................................ 5
     2.2.1 Model inheritance: Specialization ................................................................................ 6
     2.2.2 Separation of concern: ................................................................................................ 6
       Three uses of synthesis
     2.2.3 Composition on same level of abstraction .................................................................. 7
     2.2.4 Aggregation ................................................................................................................ 7
     2.2.5 Important observations: .............................................................................................. 8
   2.3 Role Model Advantages ....................................................................................................... 8
   2.4 Role models describe object patterns ................................................................................ 9
     2.4.1 Role Model Collaboration view notation ......................................................................... 9
     2.4.2 Compare with class hierarchy (VisualWorks class library) ........................................ 10
   2.4.3 Environment Collaboration view ................................................................................... 10
     2.4.4 Scenario view .............................................................................................................. 11
     2.4.5 Role modeling focuses on object patterns .................................................................. 11
   2.5 Modeling distribution ......................................................................................................... 12
     2.5.1 Determine distribution architecture .............................................................................. 12
   2.6 Separation of Concern ....................................................................................................... 13
     A TextView object plays many roles
   2.7 Reuse through model inheritance ....................................................................................... 13
   2.8 Seamless bridge to implementation ................................................................................... 14
     2.8.1 Method views .............................................................................................................. 14
     2.8.2 Mapping concepts from OOram to implementation .................................................... 15
     2.8.3 Program code Class perspective .................................................................................. 15
     2.8.4 Semantic model - Object System ................................................................................ 16

3. The first isolated application ................................................................................................... 17
   Card access control system (CAC)
   3.1 An Access Control problem Area of concern .................................................................... 17
   3.2 User interfaces Prototype Program .................................................................................... 18
   3.3 Process Steps Environment collaboration view Stimulus-Response view A Choice .......... 18
     3.3.1 System seen from environment .................................................................................. 19
     3.3.2 System seen from environment Stimulus-Response view A Choice .......................... 19
     3.3.3 System Architecture Revised plan for system development ..................................... 20
     3.3.4 Separation of Concern Consider Separation of concern ............................................ 20
     3.3.5 Consider Separation of concern .................................................................................. 21
   3.4 Validation subsystem (VAL) ............................................................................................... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Synthesize application</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td><em>CK Design</em></td>
<td></td>
</tr>
<tr>
<td>6.3.1</td>
<td>Synthesized Card/Key model</td>
<td>42</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Determine distribution architecture</td>
<td>42</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Some possible distributions</td>
<td>43</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Example: Replicated database</td>
<td>43</td>
</tr>
<tr>
<td>6.4</td>
<td>Controller, initialization and checking</td>
<td>44</td>
</tr>
<tr>
<td>7.</td>
<td>Conclusion</td>
<td>45</td>
</tr>
<tr>
<td>7.1</td>
<td>Role models describe object patterns</td>
<td>45</td>
</tr>
<tr>
<td>7.2</td>
<td>Separation of Concern</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td><em>Object pattern composition</em></td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>Modeling distribution</td>
<td>46</td>
</tr>
<tr>
<td>7.4</td>
<td>Reuse through model inheritance</td>
<td>47</td>
</tr>
<tr>
<td>7.5</td>
<td>Seamless bridge to implementation</td>
<td>47</td>
</tr>
<tr>
<td>7.6</td>
<td>More information</td>
<td>48</td>
</tr>
</tbody>
</table>