

Abstraction, Specification and Problem Decomposition

Context

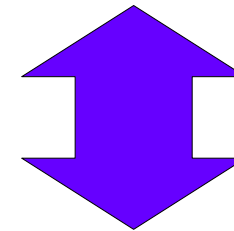
- Virtual machines
 - Word processor
 - Operating system
 - File system
- Simulation
 - Binary representation of entity
 - Operations on data
 - Effects observed
- Virtual machine as computer simulation of intended machine

Issues

- Systems are complex; difficult to build them
 - Little past experience
 - Growing aspirations
 - Software is abstract and requires much intellectual effort
 - Based on discrete logic
- Team effort required
- How can individual effort be coordinated and ultimately integrated?

Conceptual Gap

Applications: word processing, missile control, weather forecasting ..



Hardware & programming languages

Data:
numbers/encodings

Instructions: seq, conditional, loops

Software Engineering Effort

- ❑ Requirements to be understood
- ❑ Systems need to be specified
- ❑ Algorithms need to be designed and properly validated
- ❑ Implementation (coding) needs to be properly tested or validated (via methods of reasoning) for correctness

Techniques to handle Complexity

- ❑ Abstraction
 - Separating relevant from irrelevant attributes
- ❑ Specification
 - Precise statement of intended behaviour without concern of implementation
- ❑ Decomposition
 - Breaking down complex problem into simpler abstract components

Procedural Abstraction

Fix Flat Tire:

```
Check spare tire in boot;  
if Spare tire is not usable  
    Call service station;  
else  
    Replace tire;  
end-if  
Smile;
```

Procedural Abstraction

Replace Tire:

```
Remove spare tire from boot;  
Loosen nuts of flat tire;  
Jack up car;  
Remove flat tire;  
Replace spare tire;  
Release the jack to lower the car;  
Return flat tire to boot;
```

Procedural Abstraction

Loosen nuts of flat tire:

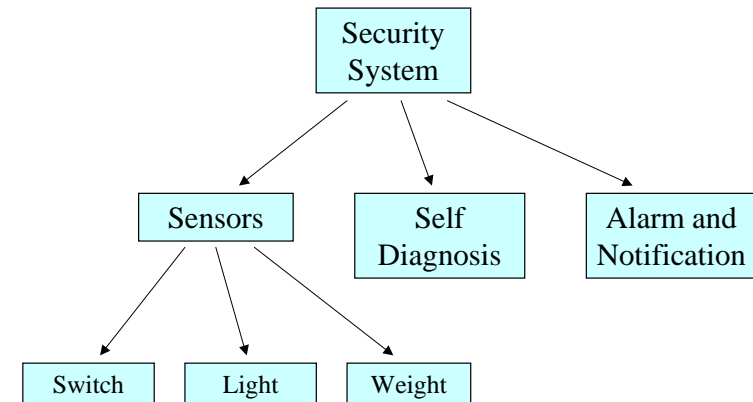
repeat

 Fit the lug wrench over a locking nut;

 Loosen nut;

until All nuts are loose;

Functional Abstraction



Phone Directory Device

□ Functionality

- Add entry
- Delete entry
- Modify entry
- Search for entry

Modify Phone Record

Modify entry:

```
name := ask("Which entry do you want changed?");
entry := Search for entry(name);
if (entry <> NONE)
    entry.number := ask("What is the new number?");
endif
```

Search Phone Record (unsorted)

Search for entry(name):

```
for j :=1 to MAXENTRIES
  if (entry[ j ].name == name)
    return(entry);
  endif
return(NONE);
endfor
```

Search Phone Record (sorted)

Search for entry(name):

```
for j :=1 to MAXENTRIES
  if (entry[ j ].name == name)
    return(entry);
  else if (entry[ j ].name > name)
    return(NONE);
  endif
endfor
```

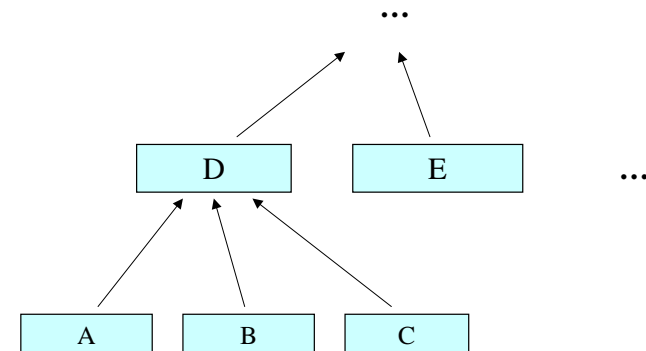
Specifications

Sort(Array)

```
for all i and j between 1 and SIZE
  if i < j then A[ i ] <= A[ j ]
```

- ❑ Agreement for developer and user to work concurrently
- ❑ Contract of services
 - to be provided by developer
 - to be relied upon

Bottom-up development

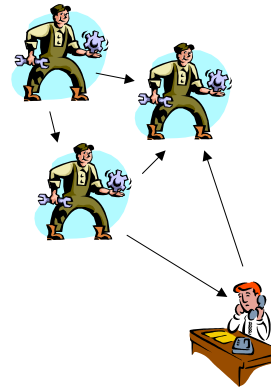


Object-Oriented Technology

- ❑ Software applications incorporate means to manipulate their representation of relevant real world entities.
 - Just as the real world is mostly populated by objects, OOT advocates modeling entities as objects in a computer system.
 - Objects interact via messaging just as entities exchange messages.
 - Objects respond to messages via executing code.

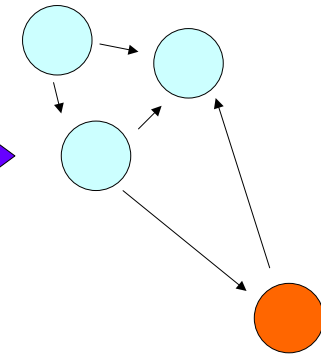
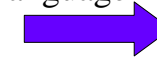
OOLanguage

Real world



Computer world

OO
language



OOT Rationale

- ❑ Relevant attributes of real world entities are abstracted into objects.
- ❑ Set of messages which object can respond to form the object interface and specification.
- ❑ Object behaviour may be implemented independently of other objects.
- ❑ Object definition can be reused in other projects.

Summary

- ❑ Software engineering is difficult due to complexity of applications.
 - Machines are low-level
 - Much VM layers required to bridge conceptual gap.
- ❑ Tools/techniques exist to help control complexity
- ❑ OOT is a proposed methodology for software construction.