JERUSALEM COLLEGE OF TECHNOLOGY
LEV ACADEMIC CENTER

Thinking Fast:

Patterns Of Cognitive Error In Software Engineering Education

Reuven Gallant
Dept. of Computer Science and Engineering
JCT Lev Academic Center, Jerusalem, IL
rgallant@alum.mit.edu

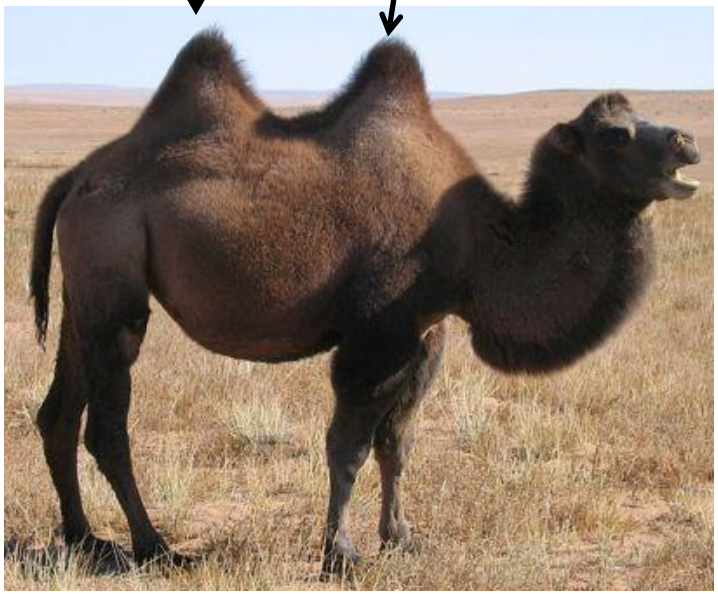
My (undergraduate 3rd year) courses

- Mandatory for Computer Science and Software Engineering students
- (1)Advanced Object Oriented Programming
- (2) Software Design for Engineers
- Content
  - Classical OO Principles and Design Patterns
  - UML
  - Executable modeling with OO Statecharts
  - System construction leveraging all of the above
    - Which is what makes the courses difficult for many students

Student Body

- ~300 Students
- 9 virtual campuses (culturally and gender segregated)
  - Distributed among 4 physical campuses

System 1 System 2



Thinking, Fast and Slow (Kahneman, 2011)

System 1

- Intuitive thinking
- operates automatically and quickly, with little or no effort and no sense of voluntary control.
- effortlessly originates impressions and feelings that are the main sources of the explicit beliefs and deliberate choices of System 2

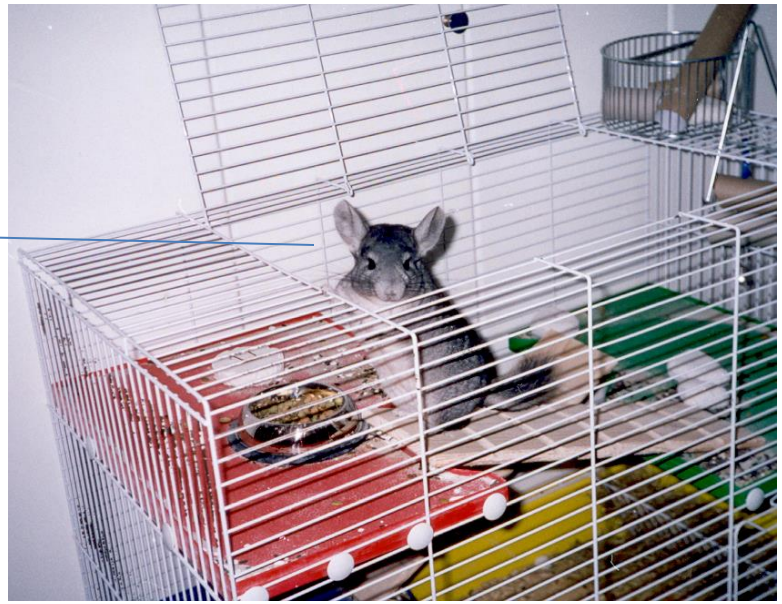
System 2

- Rational thinking
- allocates effortful mental activities that demand it, including complex computations.
- believes itself to be where the action is, but system 1…
- there are vital tasks that only System 2 can perform because they require effort and acts of self-control in which the intuitions and impulses of System 1 are overcome.

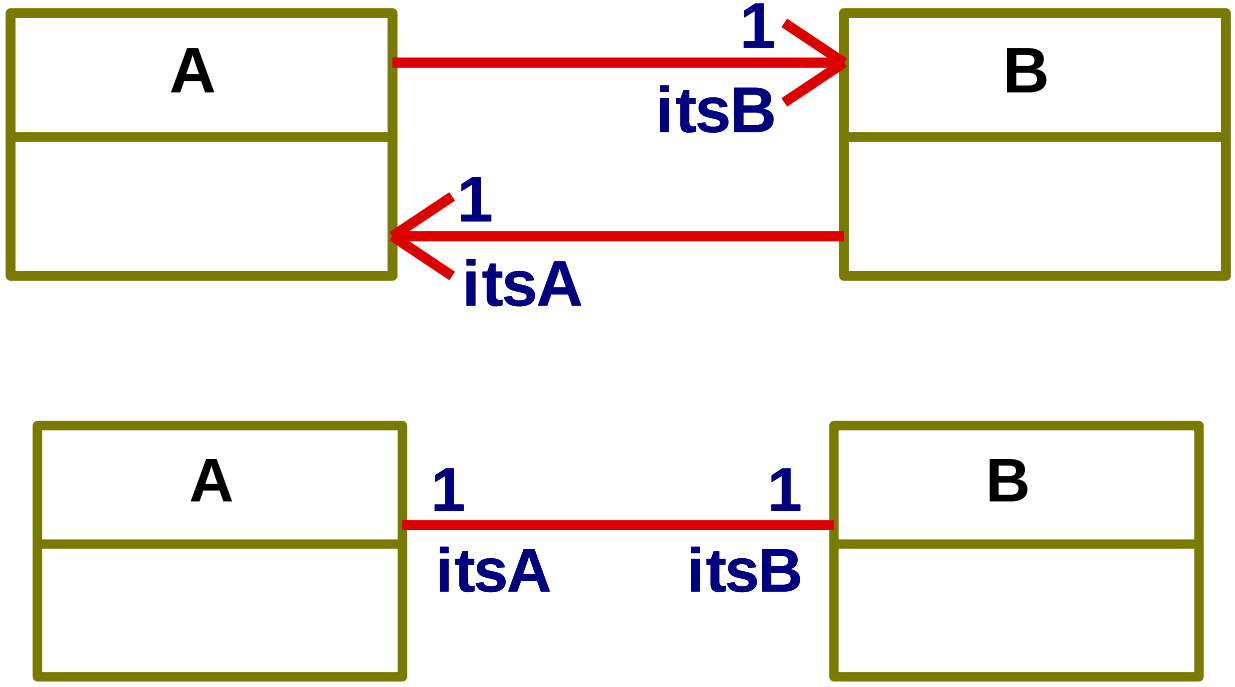
Law of least effort

- if there are several ways of achieving the same goal, people will eventually gravitate to the least demanding course of action.
- In the economy of action, effort is a cost and the acquisition of skill is driven by the balance of benefits and costs.
- Laziness is built deep into our nature.
  - priming
    - EAT->SO_P->SOUP and not SOAP
    - Emotions, unconscious thoughts, visual cues also prime
  - difficult question-> easier “heuristic” question
    - “how much would you spend to save an endangered species?”
    - ->”how much emotion do I feel when I think of dying dolphins?”

System 1->System 2



Reciprocity: Bi-directional association (UML)



Reciprocity (2)

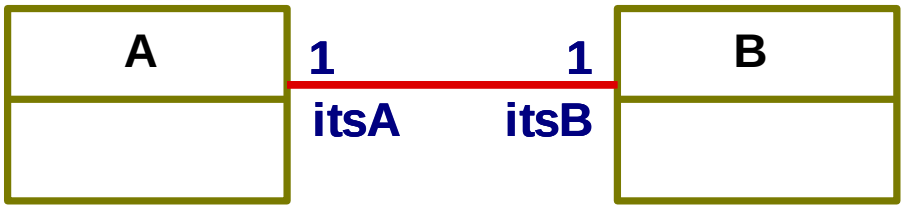
// file A.h

```
#ifndef A_H
#define A_H
☒ #include “B.h”
☑class B;
class A {
public :
A();
~A();
void setItsB (B* p_B);
B* getItsB ();
protected :
B *itsB;
};
#endif
```

// file B.h

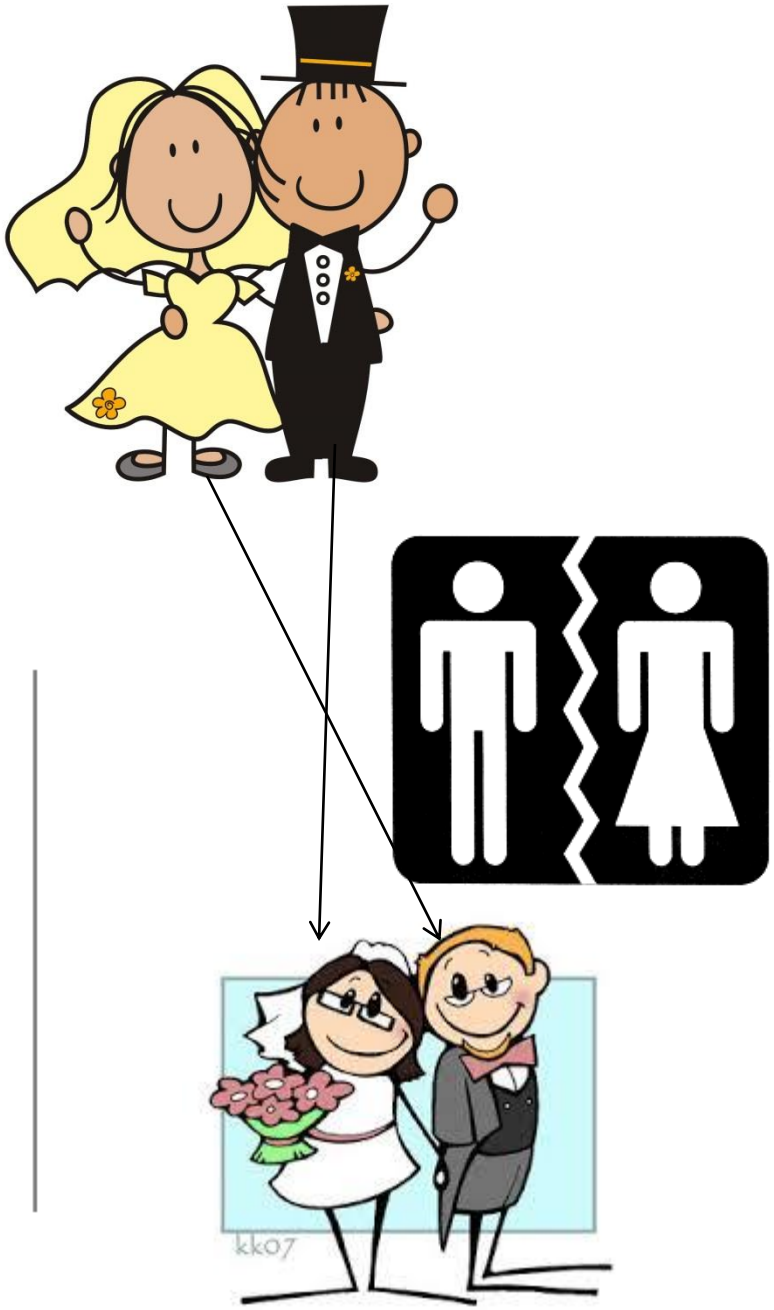
```
#ifndef B_H
#define B_H
☒#include “A.h”
☑class A;
class B {
public :
B();
~B();
void setItsA (A* p_A);
A* getItsA ();
protected :
A *itsA;
};
#endif
```

Reciprocity (3)



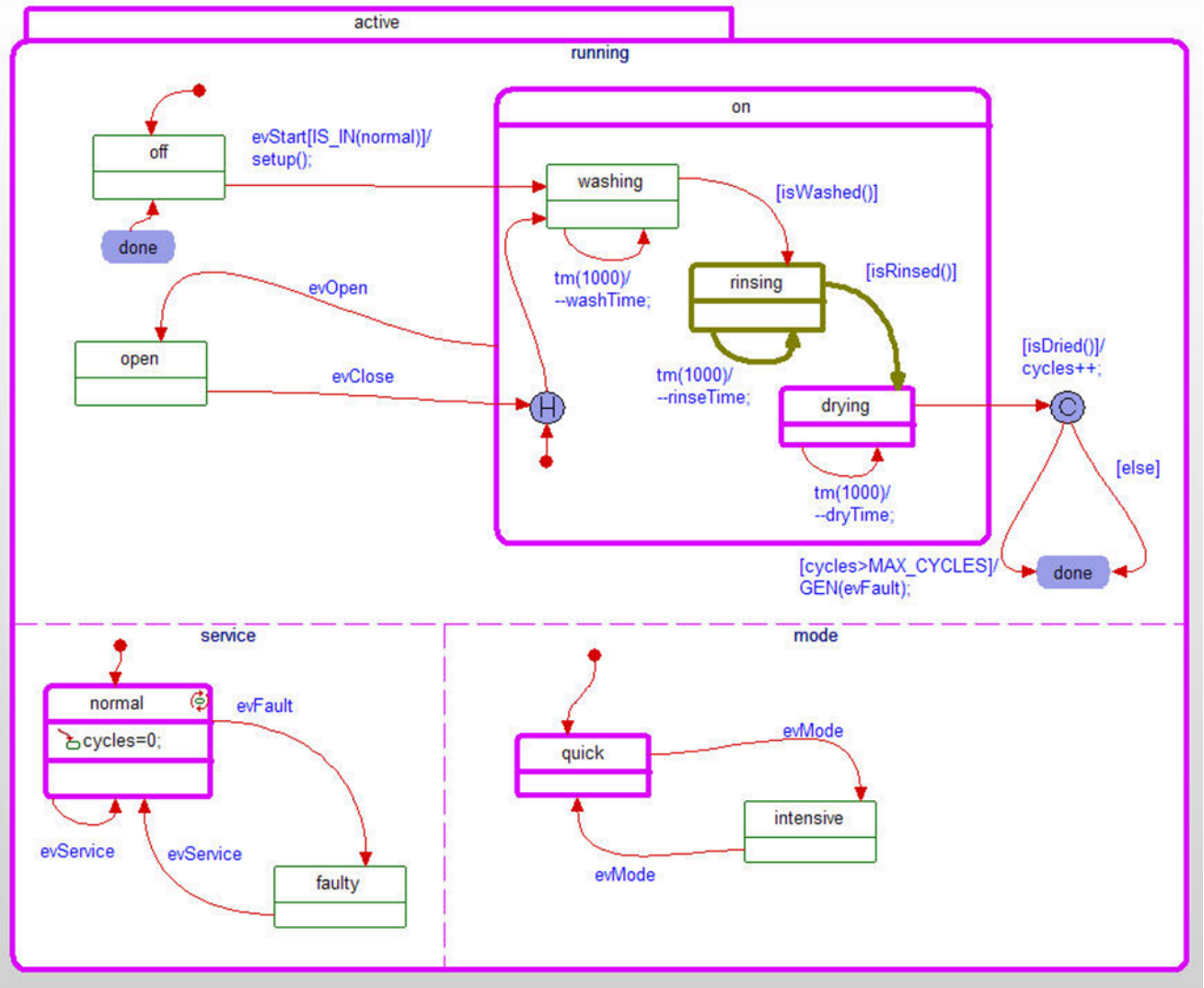

```
void A::__setItsB(B* p_B) {
   itsB = p_B;
}

void A::_setItsB(B* p_B) {
   if(itsB != NULL)
      {
         itsB->__setItsA(NULL);
      }
   __setItsB(p_B);
}

void A::setItsB(B* p_B) {
   if(p_B != NULL)
      {
         p_B->_setItsA(this);
      }
   _setItsB(p_B);
}
```

```
void B::__setItsA(A* p_A) {
   itsA = p_A;
}

void B::_setItsA(A* p_A) {
   if(itsA != NULL)
      {
         itsA->__setItsB(NULL);
      }
   __setItsA(p_A);
}

void B::setItsA(A* p_A) {
   if(p_A != NULL)
      {
         p_A->_setItsB(this);
      }
   _setItsA(p_A);
}
```

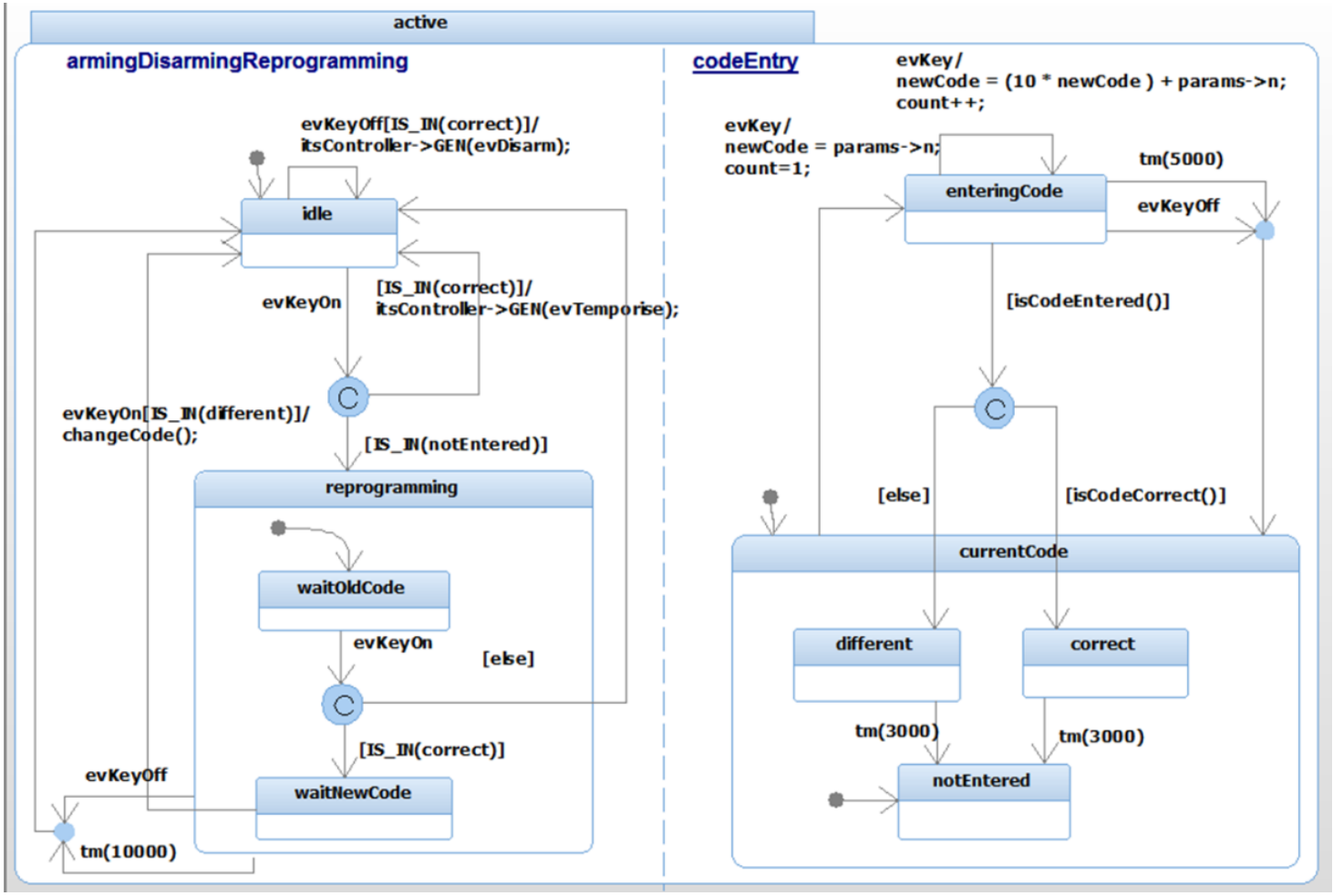


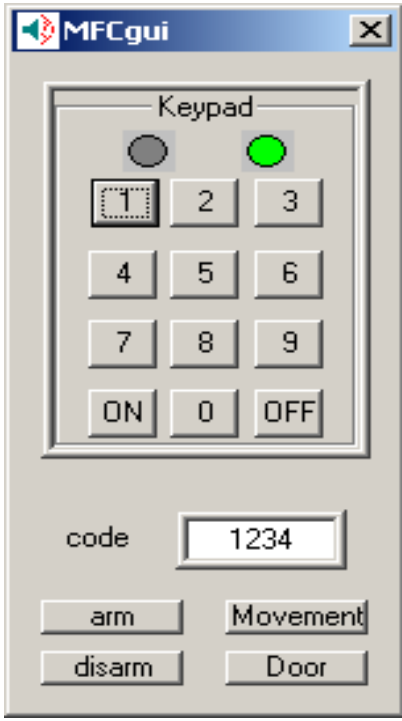
Reciprocity (4)

- What happens in the middle of a run, if we change duration settings {intensive, quick}?
- What happens if there is fault in the middle of a run?
- answer (???) WYSIWTI



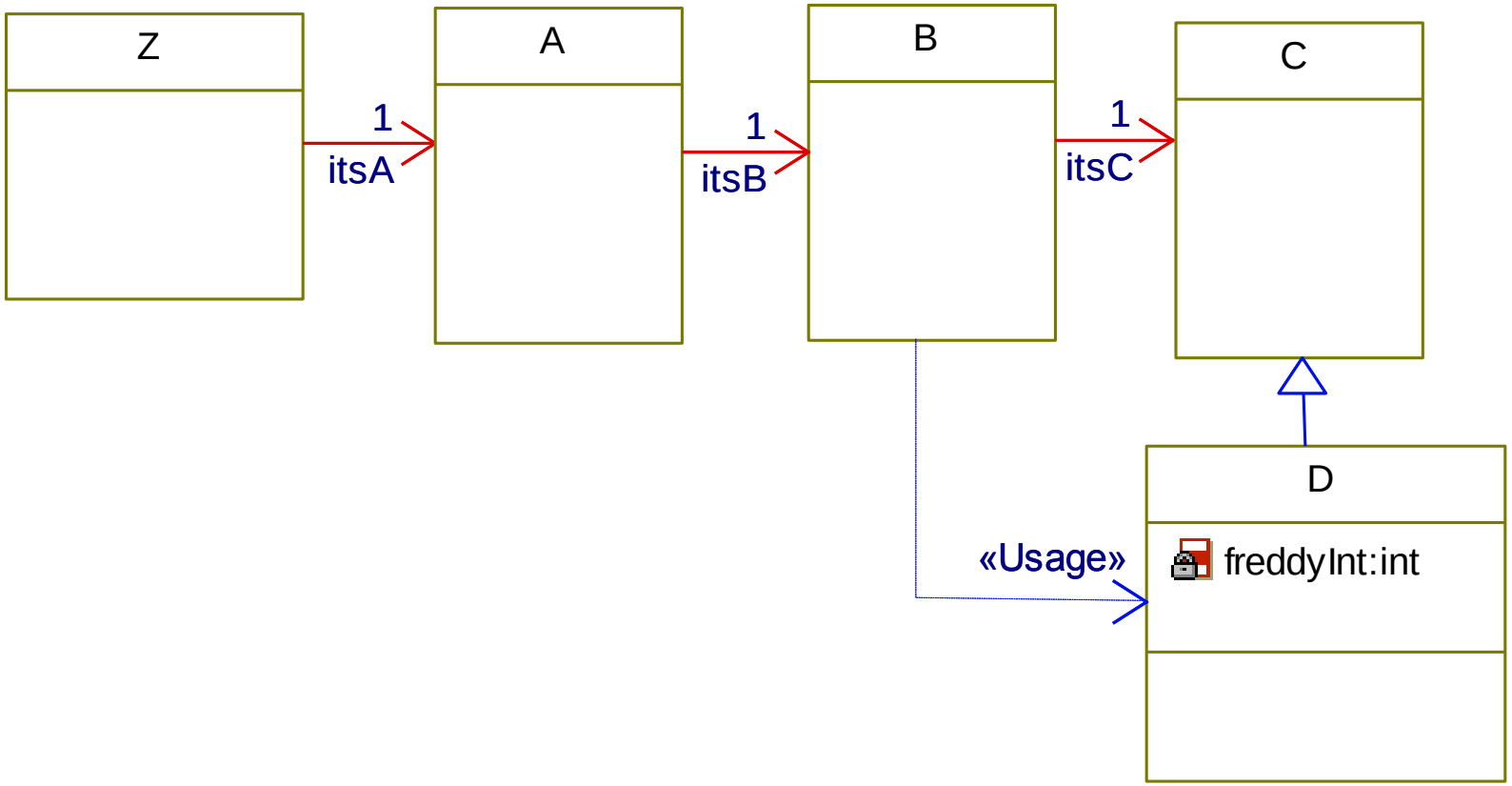
Reciprocity (4)





Domino Effect (1)

- Which files will require recompilation after adding an attribute int fredrikaInt to class D?
- answer (sic) “What you see is all there is” **WYSIWTI**



Domino Effect (2)

- What happens when freddy opens (and then closes) the door?

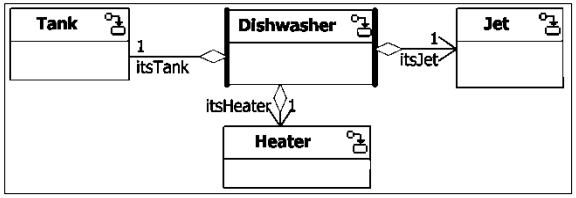


Figure 13: Dishwasher controller and subsystems.

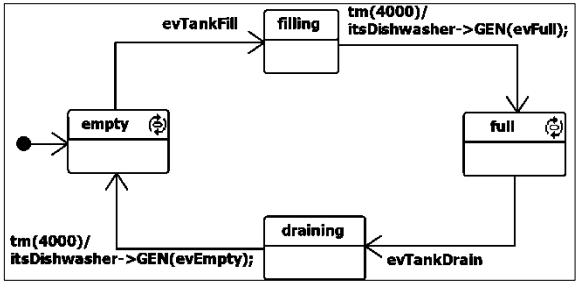


Figure 15: Tank subsystem statechart.

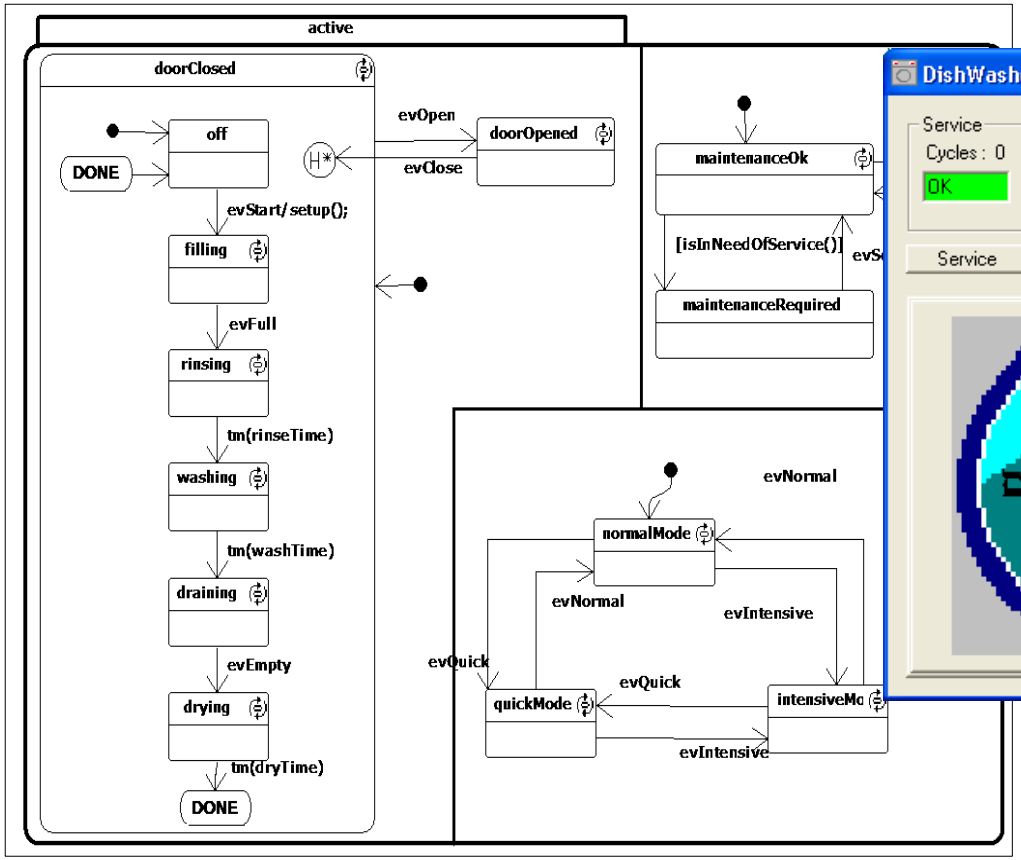


Figure 14:Dishwasher controller statechart.

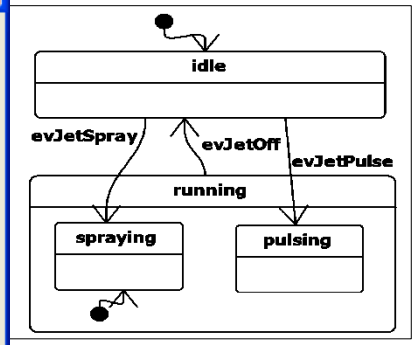


Figure 16:Jet subsystem statechart.

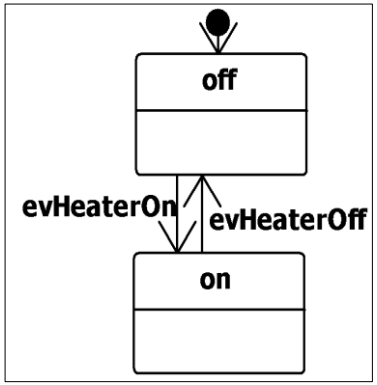
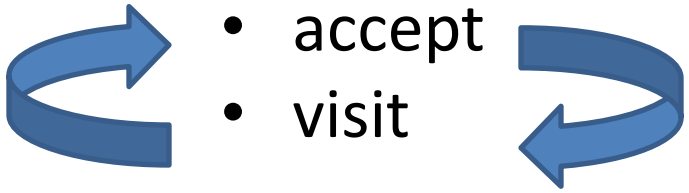


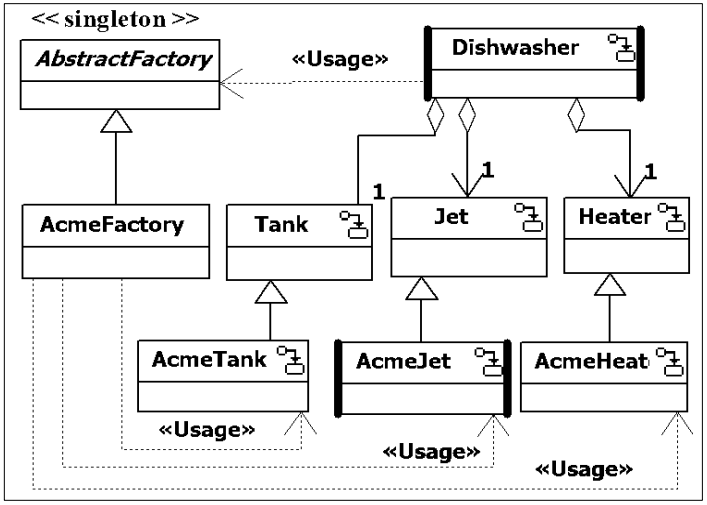
Figure 17: Heater subsystem statechart.

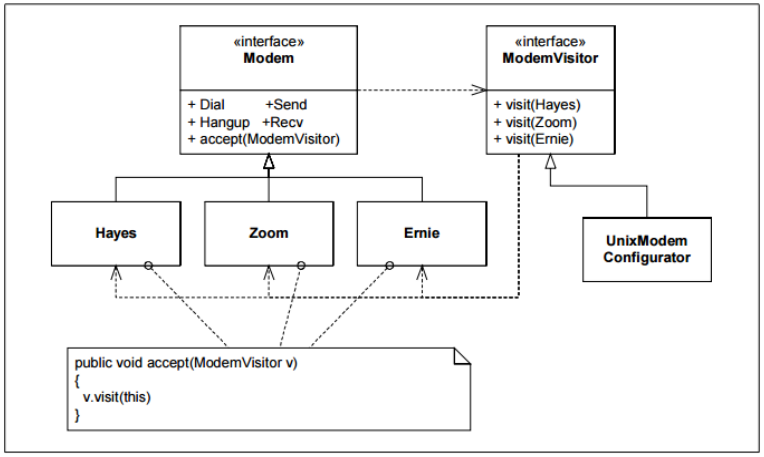
Weak Priming->Mediterranean salad

- Singleton Method for Abstract Factory
- Non-singleton factory method
- Singleton factory method



- accept
- visit





Strong Priming-> right answer to the wrong question

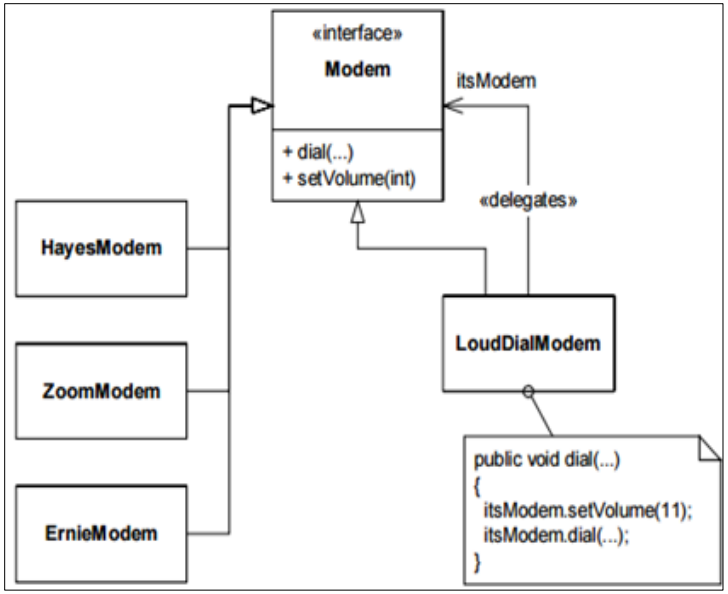


Figure 3: Modem with Decorator Pattern

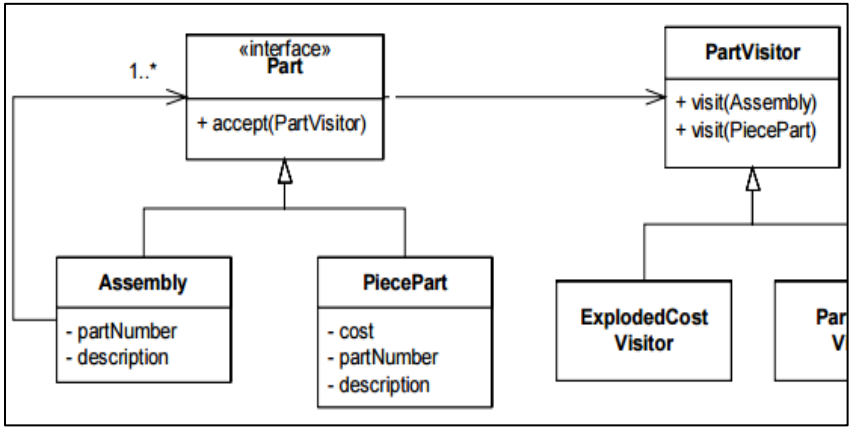
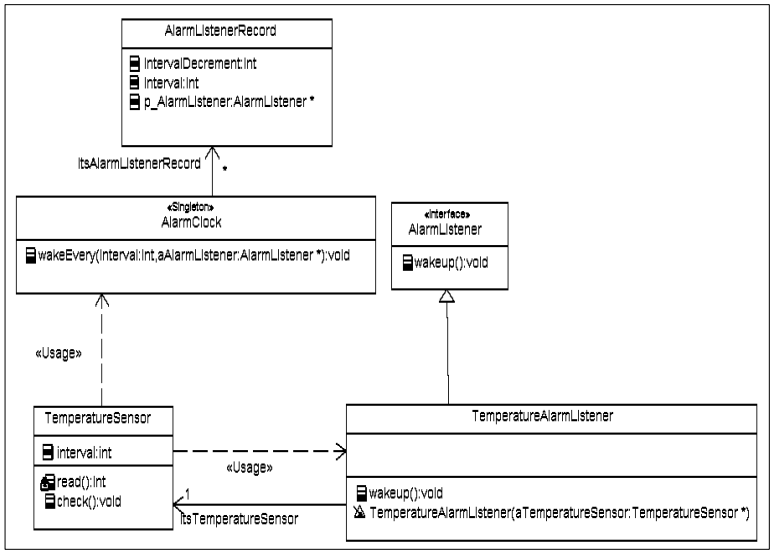
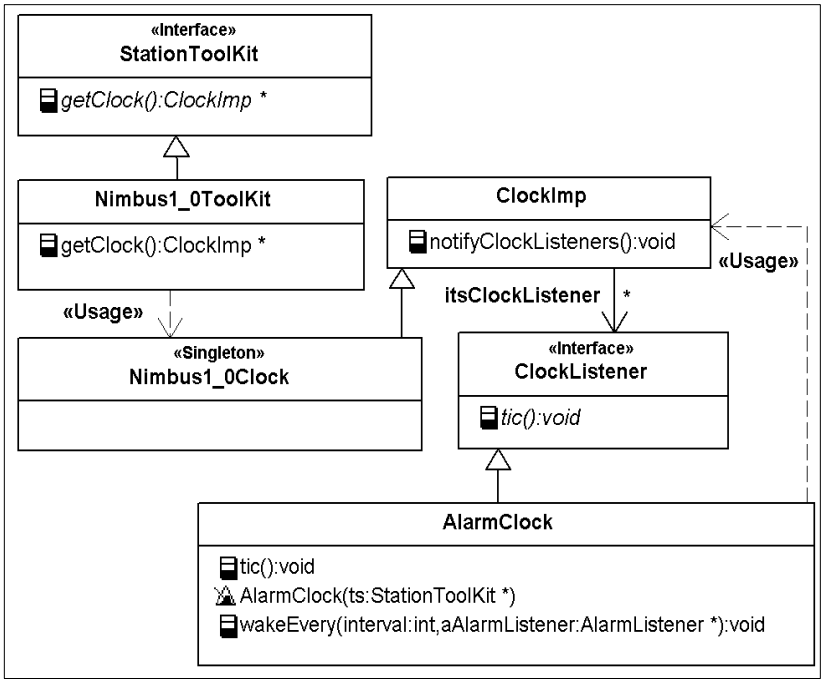
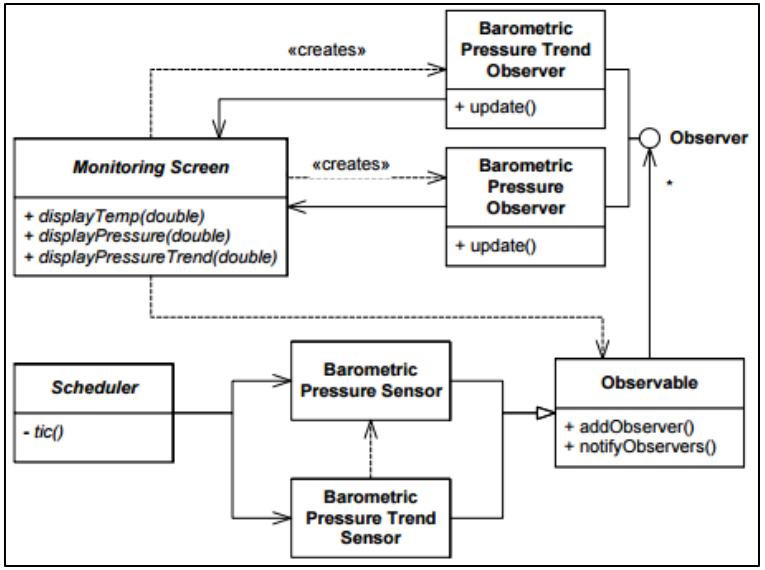


Figure 4: Visitor Pattern to Generate Report on Composite

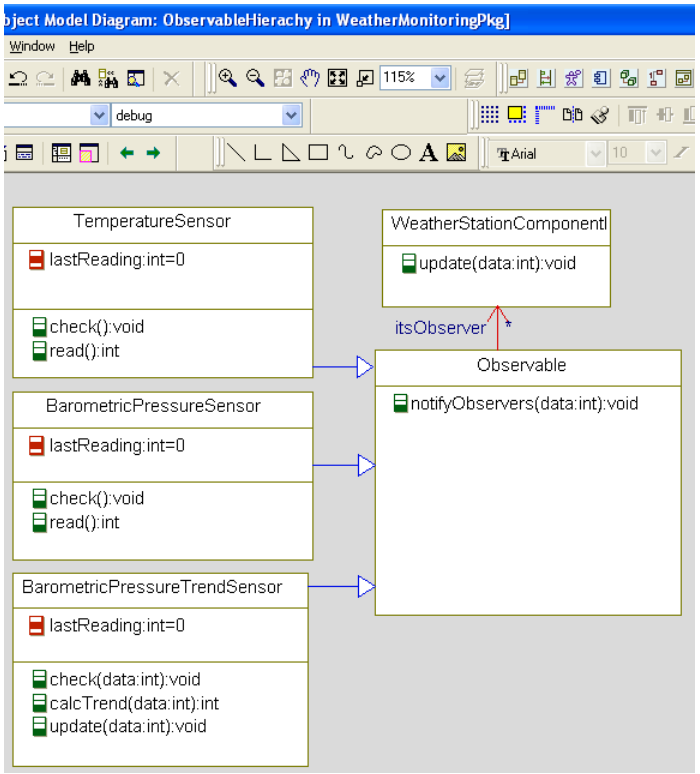
Opposite Roles

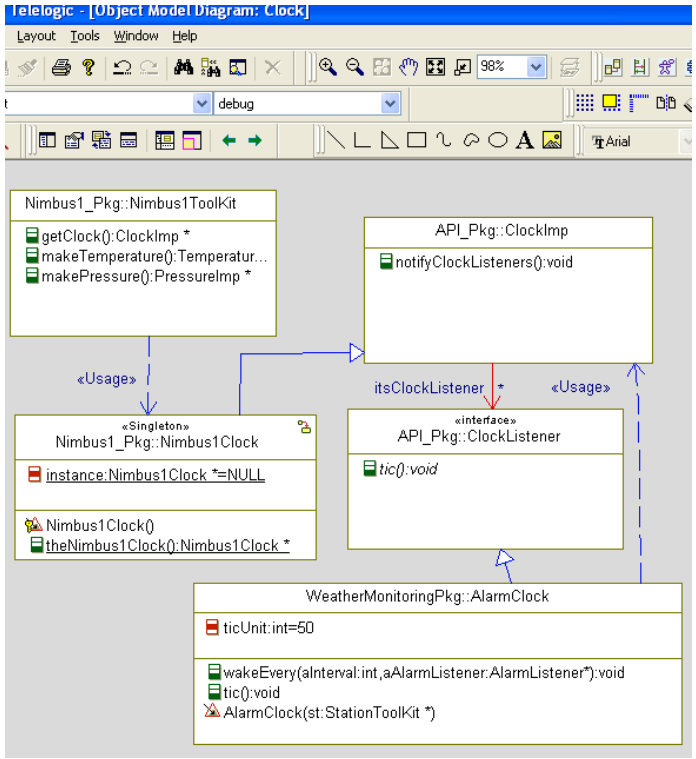


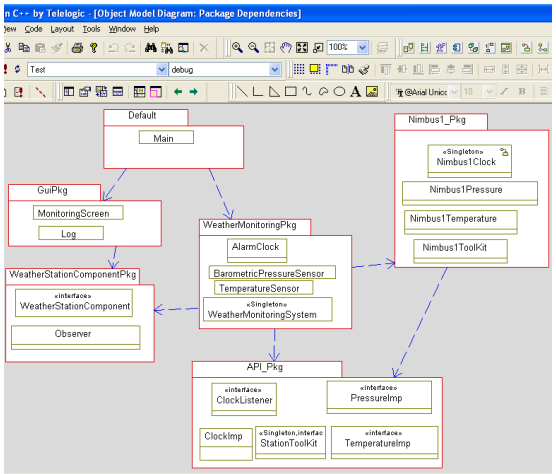


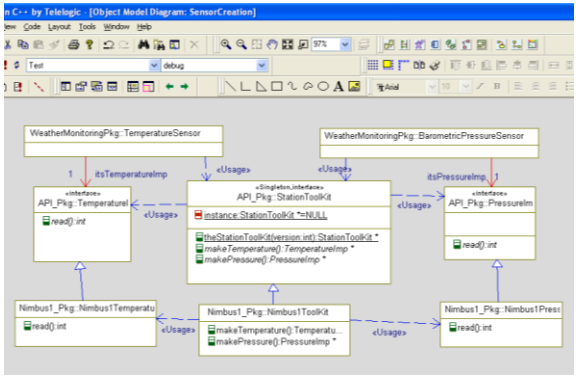


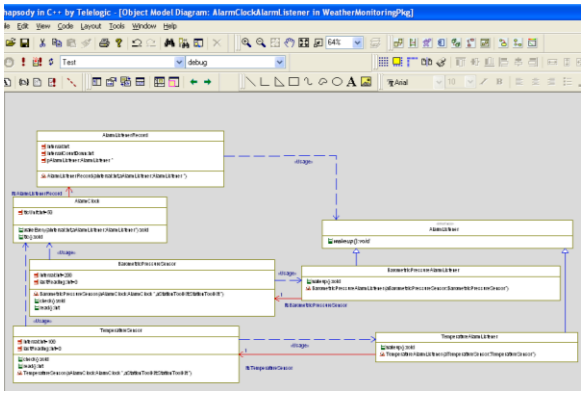
System Construction











Discussion

- student population endowed with highly developed **rational** faculties which must often defer to **intuition** due to constraints on **effort** and **attention** that can be devoted to studies provides a fertile laboratory in which to examine the role of **System 1**, **System 2**, **priming** and **WYSIATI** in the learning process
- Data mining from scanned exam booklets
  - collected from many nations, disciplines and cultures.
  - develop a rich taxanomy of error types;
  - correlate different error types (e.g.., does a student who has trouble differentiating between similar problem types, also have problems with priming?);
  - Data must be adjusted for cultural bias, educational methods, discipline types, etc.
- Course modules tailored per cognitive abilities?
- Whereas in academia failure to account for cognitive variation affects only grade point average, in mission and/or safety critical systems the costs could and have been much higher. We envision tailoring employee training and work assignments according to cognitive characteristics. This however could be a sensitive issue. Just as there is opposition to genetic testing to determine health insurance premiums, there would be considerable, and in our mind justifiable opposition to cognitive testing as a criterion for hiring and advancement.