Lecture_6

Structured specifications

- an approach to writing requirements where the freedom of the requirements is limited and requirements are written in a standard way

Specification	**Description**
Insulin Pump/Control Software	
Function	Compute insulin dose
Description	Computes dose to be delivered when the current level is in the safe zone between 3 and 7 units
Inputs	Current sugar (r2) the previous two readings (r0 and r1)
Source	
Outputs	
Action	
Requirements	
Pre-condition	
Post-condition	
Side effects	

Use Case

- Kind of scenario - Scenarios are useful in discussing a proposed system with a client - Concept of how the system will be used by different users - Usually, each case represents one functional requirement

Use Case Specification requirements Examples

- The ATM shall dispense cash in 20s - The max withdrawal is 500 dollars...

Actor

- Model for an external entity that interacts with the system - User Role - denoted by a stick figure - External system - Not specific physical entities but merely particular facets (i.e. roles) - Which user groups require help from the system? - Which needs to execute the system's most obvious functions? - Flow of events - Special Requirements - Actors - Alt flow of events

Association

- The participation of an actor in a use case is shown by a solid line in UML

System Boundary

- A rectangular box that indicates the scope of your system

Relationships

- Association between use case and actors - Generalization between actors - Generalization between use case - Include use case and actors - Extend use case and actors

Actor Generalization

- One actor can inherit the role of another - The descendant inherits all the use cases of the ancestor - Solid directed line with triangle

Use Case Generalization

- Relationship between use cases means that a child use case inherits the behavior and meaning of the parent use case - The child may add or override the behavior of the parent - Solid directed line with triangle

<<`Include`>> Relationship

- When a use case is using the functionality of another use case as a part of its event flow this relationship is signified by `<>` - The include modularizes the process - Included with the other use case: checkout at the store, including scan items, purchases, etc

<<`Extend`>> Relationship

- Extend is a directed relationship that specifies how the behavior of a base use case can be extended by a supplementary use case - Extending defines optional behavior: Ex Deposit Bonus if over 55