Unified Modelling Language

Phil Robinson

Lonsdale Systems

www.iinet.net.au/~lonsdale

What is the UML?

- A language that unifies the industry's best engineering practices for modelling software systems
- Goals
  - Simple and extensible
  - Broad application
  - Implementation independent
  - Process independent

The Evolution of the UML

Introducing the ‘Three Amigos’

Mid 70's - Mid 90's

- Competing object oriented methods
  - Booch
  - OMT (Rumbaugh)
  - OOSE (Jacobson)
  - Others

1995

- **Unified Method**
  - **Booch**
  - **Rumbaugh**

1996

- Booch and Rumbaugh are joined by Jacobson
- They become the ‘Three Amigos’
- ‘Unified Method’ becomes ‘Unified Modelling Language’
- The ‘UML Partners’ begin working with the ‘Amigos’

1997

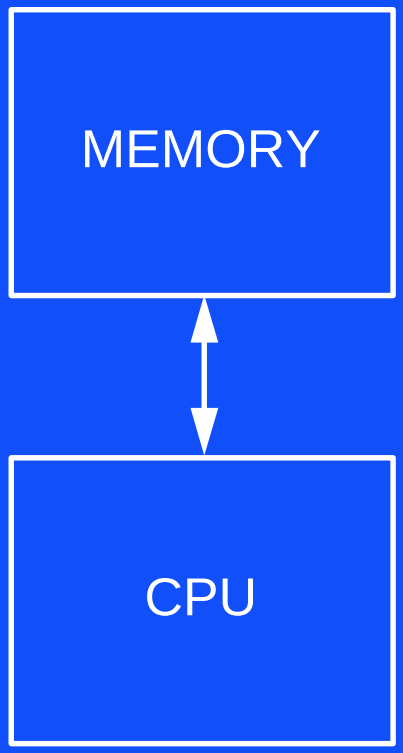
- **UML 1.0 proposal submitted to the Object Management Group (OMG)**
- **UML 1.1 Adopted as an OMG standard**

UML Since Adoption

- 1998 **UML 1.2** (editorial clean up)
- 1999 **UML 1.3** (technical revision)
- 2000? **UML 1.4** (planned minor revision)
  International standard (ISO)
- 2001? **UML 2.0** (planned major revision)

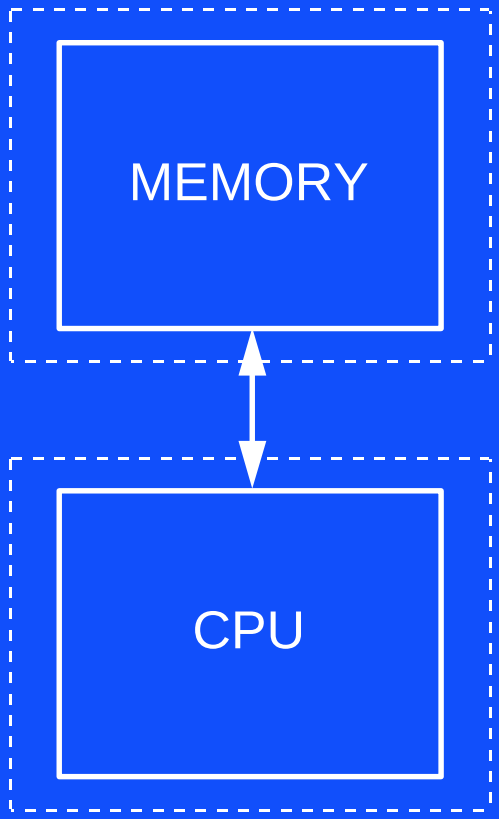
Object Oriented Software

Hardware Architecture



- **Unchanged since 1950's**
- **Separation of data and logic**
- **Strong influence on software development**

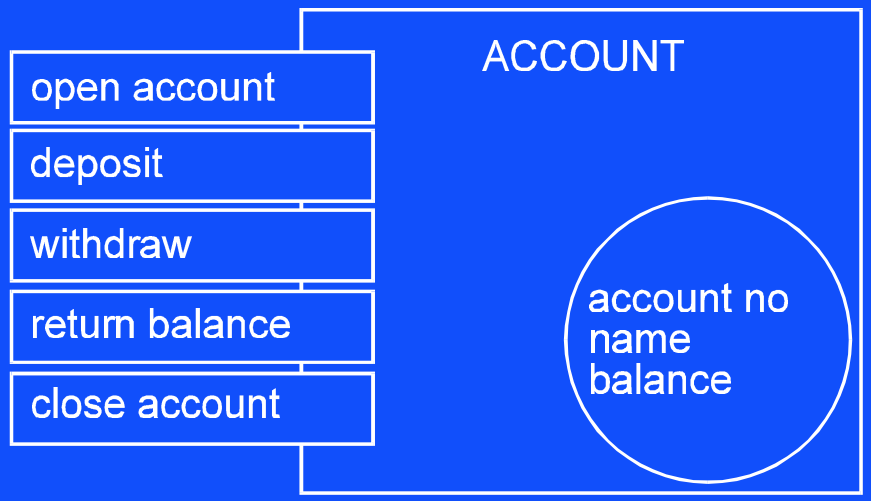
Conventional Software




```
char name, address;
float balance, amount;

...
balance = balance + amount;
...
```

Object Oriented Software

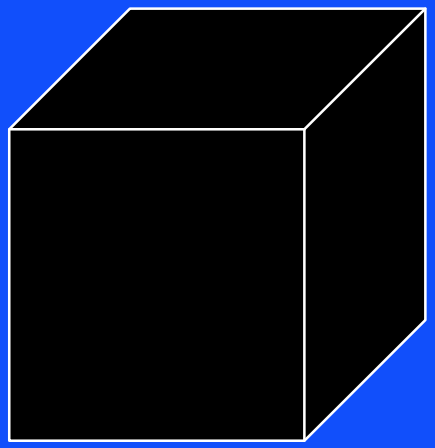


- **'Object Classes' reflect the problem domain (rather than the hardware architecture)**
- **Information hiding**
- **'Normalisation' of logic**
- **Easier to reuse**

Development is Different

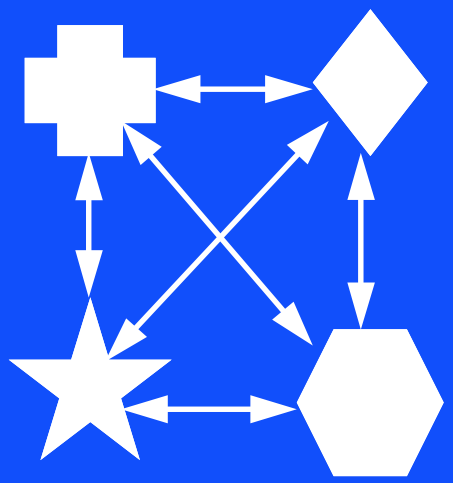
- **Conventional**
  - **Separate data and function models**
  - **Decomposition guides modelling**
  - **Paradigm changes throughout the development life-cycle**
  - **Danger of ‘logical spaghetti’**
- **Object Oriented**
  - **Single object model with many ‘views’**
  - **Abstraction guides modelling**
  - **Consistent paradigm throughout the development life-cycle**
  - **Danger of ‘logical ravioli’**

External System Behaviour



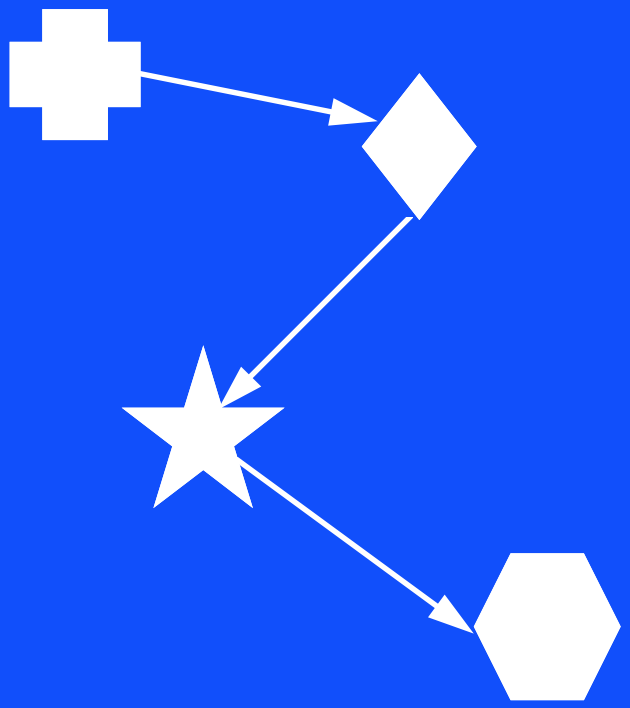
- **Externally visible behaviour**
- **User's perspective**
- **'Black box' view**

System Structure



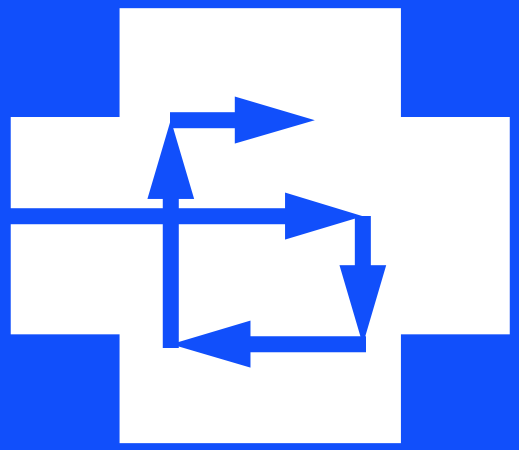
- Classes
- Instances
- Relationships

Internal System Behaviour



- Collaborations
- Object instances
- Messages
- Developer’s perspective
- ‘Glass box’ view

Internal Object Behaviour



- **Behaviour**
  - **Operations**
  - **Methods**
- **State**
  - **Attributes**

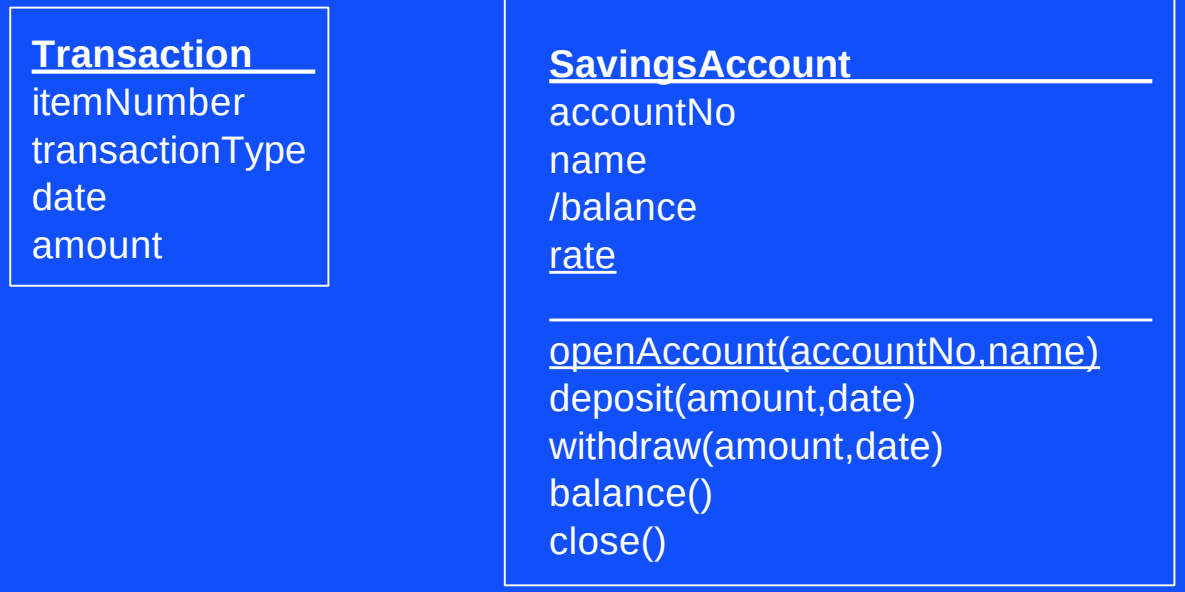
UML Diagrams

- **Class/Object Diagrams** (System structure)
- **Use Case Diagrams** (External system behaviour)
- **Sequence Diagrams** (Internal system behaviour)
- **Statechart Diagrams** (Internal object behaviour)

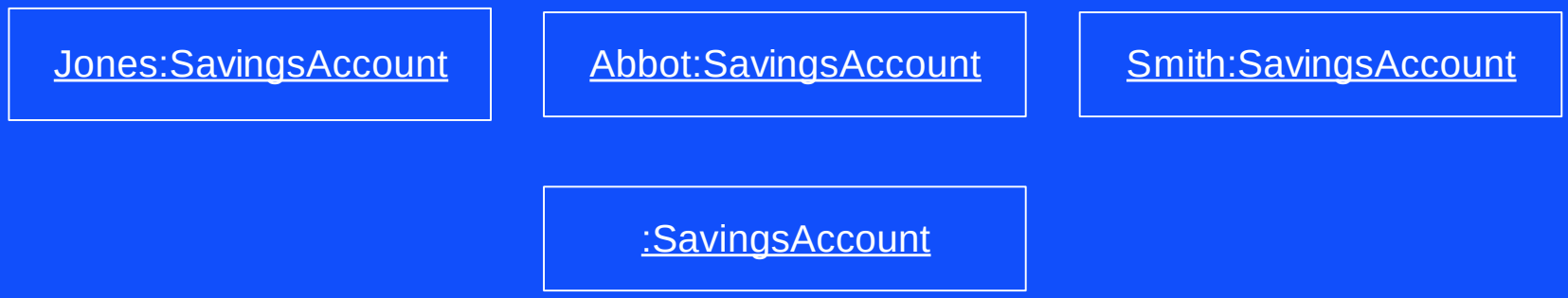
Class/Object Diagrams

System Structure

Classes


Transaction
itemNumber
transactionType
date
amount
SavingsAccount
accountNo
name
/balance
rate
openAccount(accountNo,name)
deposit(amount,date)
withdraw(amount,date)
balance()
close()

Objects


Jones:SavingsAccount
Abbot:SavingsAccount
Smith:SavingsAccount
:SavingsAccount

Relationships

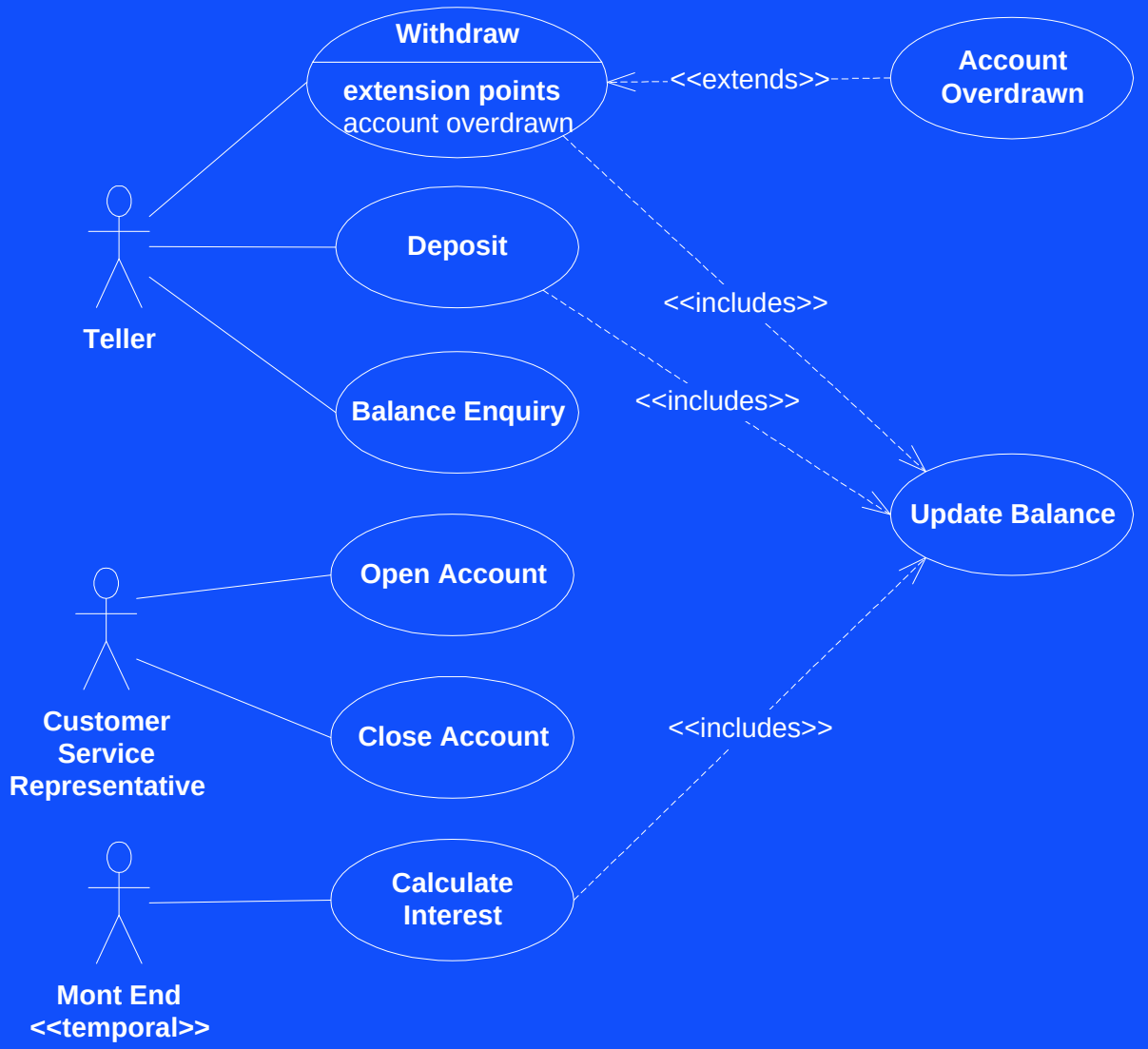
Use Case Diagrams

External System Behaviour

Use Case Components

- Actors
- Use case
  - Basic scenario
  - Alternate scenarios
- Extends
- Includes

Use Case Diagram



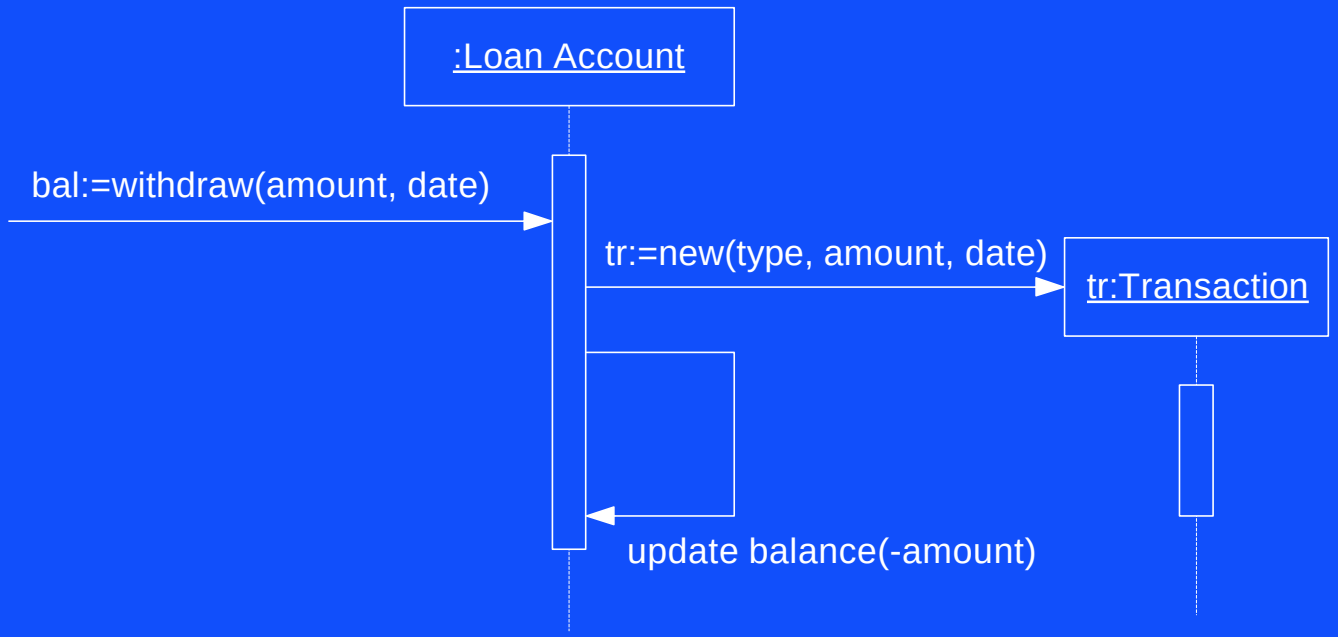
Sequence Diagrams

Internal System Behaviour

Sequence Components

- Objects
- Life lines
- Activations
- Messages

Sequence Diagram



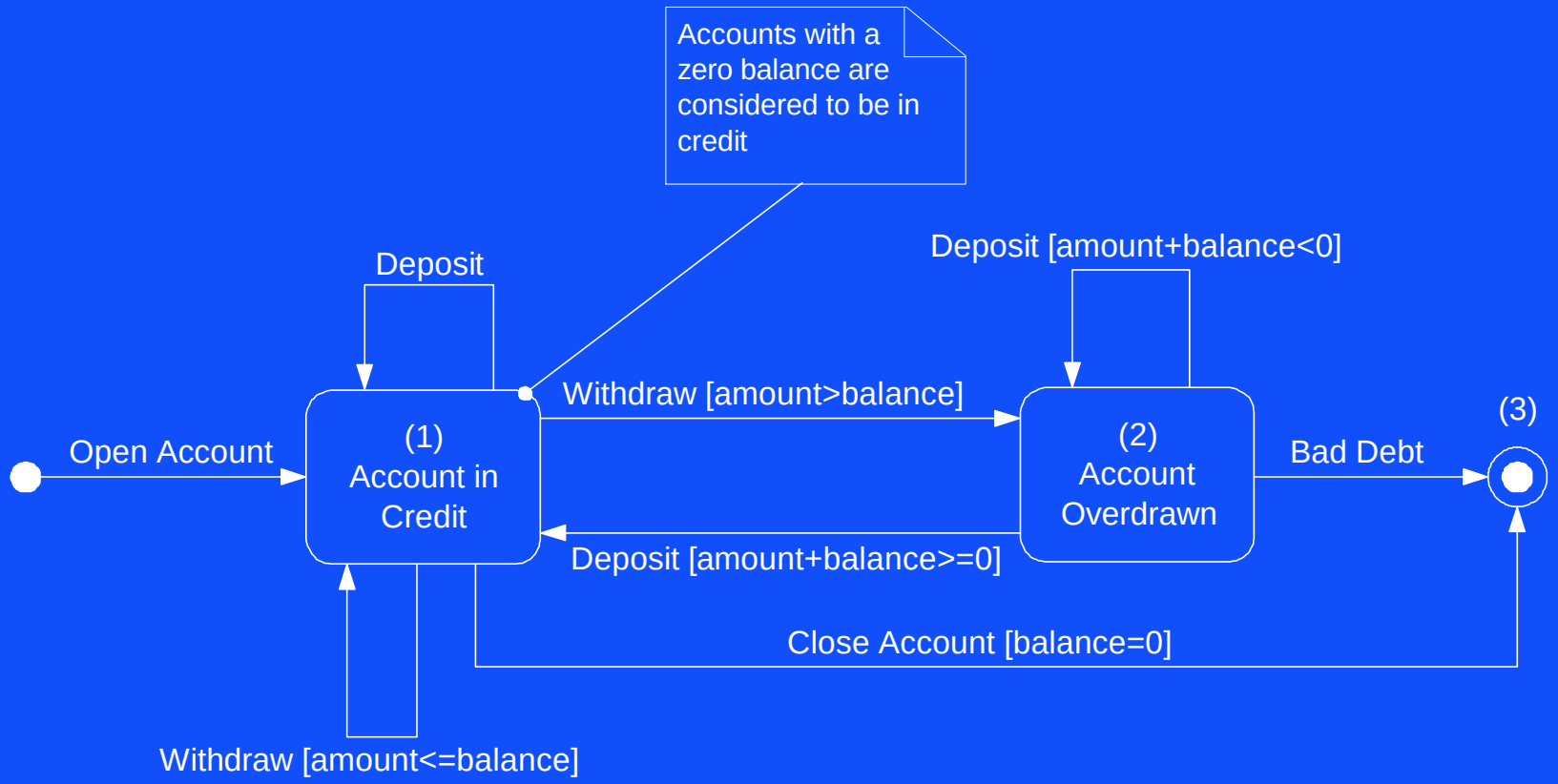
Statechart Diagrams

Internal Object Behaviour

Statechart Components

- States
- Transitions
- Events

Statechart Diagram



But there’s more…

Modelling Elements

- Collaboration diagrams
- Activity diagrams
- Interfaces
- Packages
- Realise relationship
- Dependency relationship
- Object Constraint Language (OCL)
- Extension mechanisms

Implementation

- Deployment
  - Nodes
  - Connections
  - Components
- Repository based development
- ‘Round trip’ development tools

Process

- UML describes software development ‘artefacts’
- The ‘Unified Software Development Process’
  - Use case driven
  - Architecture centric
  - Iterative and incremental
- UML can be used with other processes

Unified Modelling Language

Phil Robinson

Lonsdale Systems

www.iinet.net.au/~lonsdale