UNIVERSITY OF Southampton
School of Electronics and Computer Science

# 12. Extending Event-B Models

Michael Butler and Thai Son Hoang

ECS, University of Southampton, U.K.

COMP1216
18th March 2019

# Objectives

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Understanding abstraction and refinement
- Extension refinement by an example

## Readings

- Hoang [2013] (in particular section on superposition refinement)

# Outline

UNIVERSITY OF Southampton
School of Electronics and Computer Science

UNIVERSITY OF Southampton
School of Electronics and Computer Science

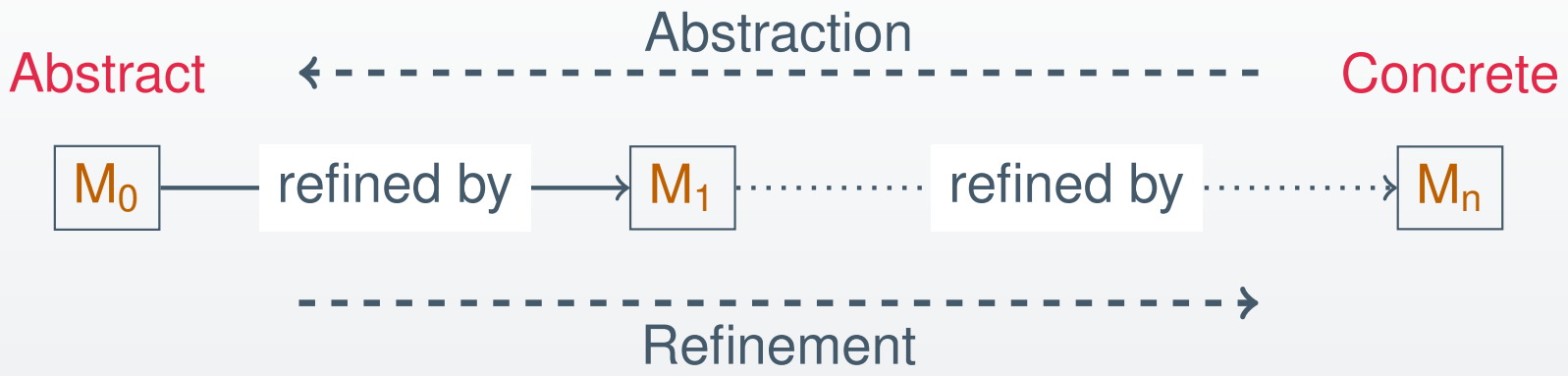
## Abstraction

Abstraction can be viewed as a process of simplifying our understanding of a system.

- The simplification should
  - focus on the intended purpose of the system
  - ignore details of how that purpose is achieved.
- The modeller needs to make judgements about what they believe to be the key features of the system.

# On Refinement (1/3)

UNIVERSITY OF Southampton
School of Electronics and Computer Science




- Refinement is a process of enriching or modifying a model to
  - augment the functionality being modelled, or
  - explain how some purpose is achieved

# On Refinement (2/3)

UNIVERSITY OF Southampton
School of Electronics and Computer Science


- $M_1$ is a refinement of $M_0$
- $M_0$ is an abstraction of $M_1$

## Facilitates abstraction

We can postpone treatment of some system features to later refinement steps

## Coping with System Complexity

Abstraction and refinement together should allow us to manage system complexity in the design process

# On Refinement (3/3)

UNIVERSITY OF Southampton
School of Electronics and Computer Science


## Properties Preservation

Properties are preserved during refinement

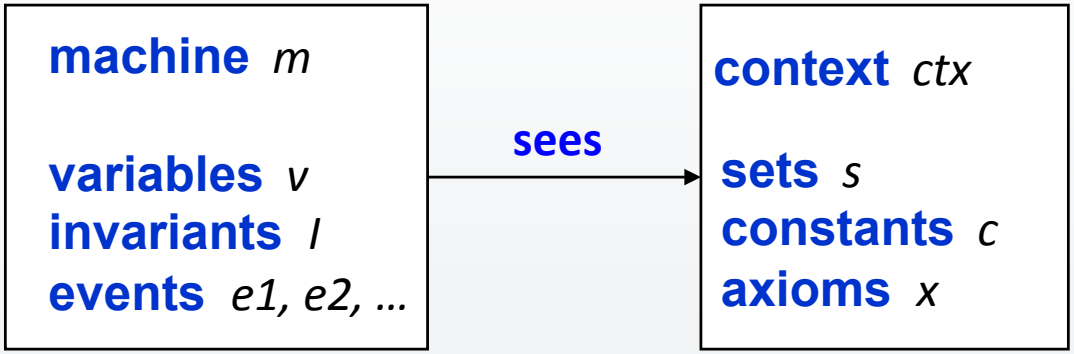
# Event-B Refinement

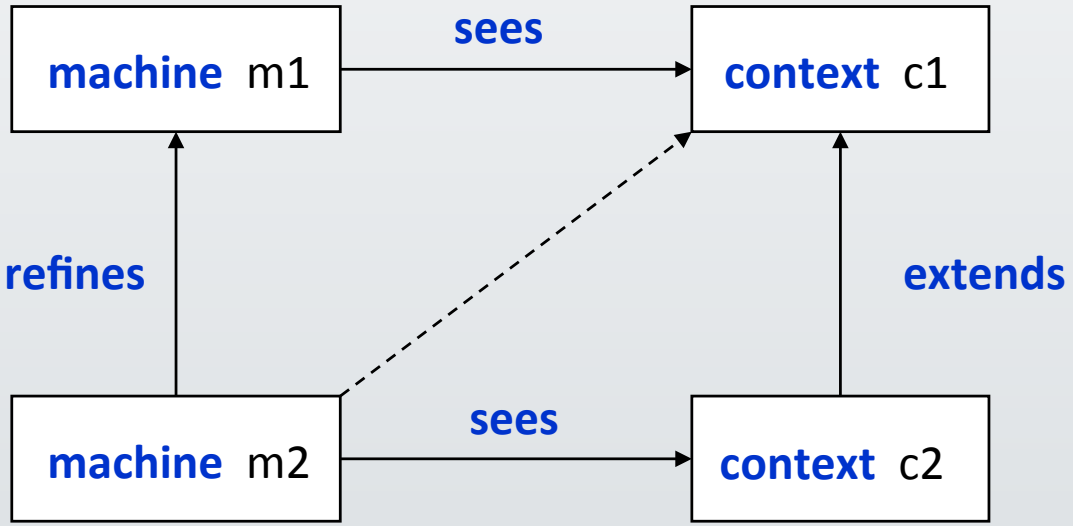
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Preserve safety (e.g., invariants) properties.
- We use proofs to verify the consistency of a refinement step
- Failing proofs help identify inconsistencies in a refinement step

# Modelling Components and Refinement

UNIVERSITY OF Southampton
School of Electronics and Computer Science



# Example. A Secure Database

UNIVERSITY OF Southampton
School of Electronics and Computer Science

## Requirements

REQ 1 A new user can be added with a clearance level (1 .. 10)

REQ 2 An existing user can add a new object (hence becomes the owner of that object).

REQ 3 An existing object can be removed by its owner.

REQ 4 An existing user owning no objects can be removed.

REQ 5 Each object is classified by some access level (1 .. 10)

REQ 6 An object's classification is at most its owner's clearance level.

REQ 7 Each object is associated with some data

REQ 8 An object can be accessed (read/written) by any user whose clearance level at least the object's classification

REQ 9 The user clearance level can be adjusted.

REQ 10 The object classification level can be adjusted.

# Refinement Strategy

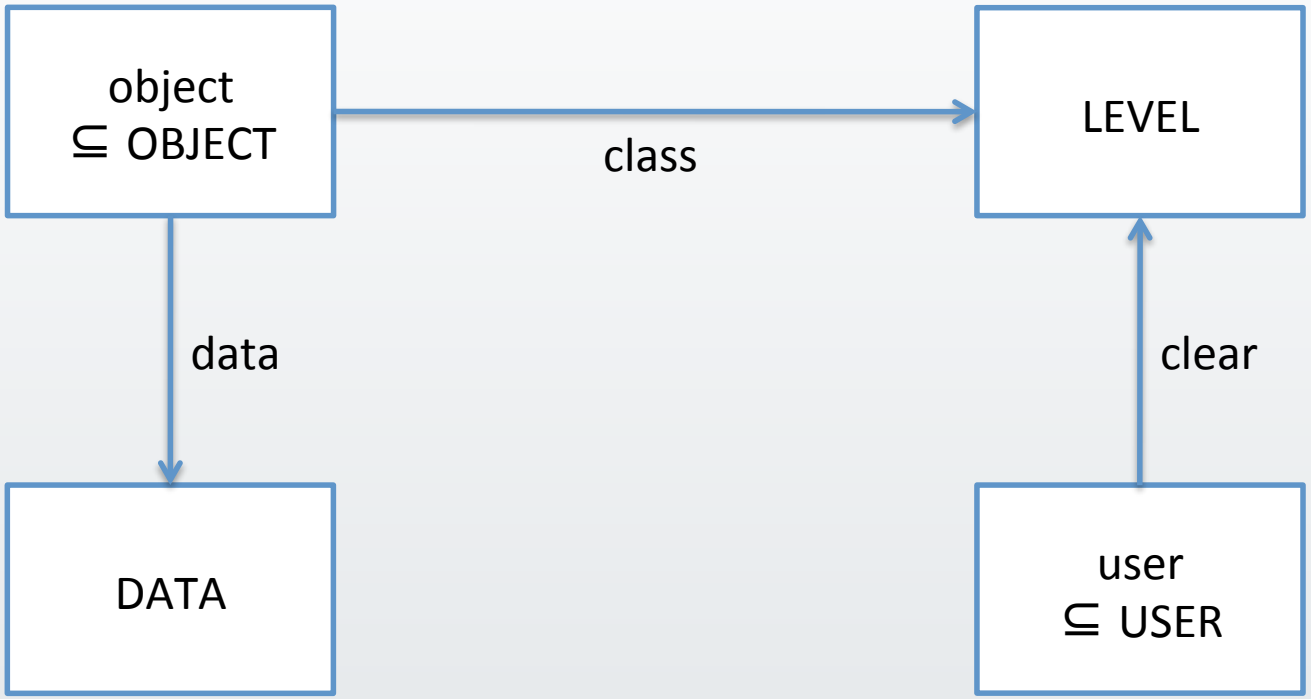
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- SecureDB1: We focus on
  - objects and their classification, and
  - users and their clearance levels.
- SecureDB2: We focus on objects' ownership.

# Abstract Level SecureDB1 (Recall)

## Class diagram for the Secure Database

UNIVERSITY OF Southampton
School of Electronics and Computer Science



# Abstract Level SecureDB1 (Recall)

## Types and Variables for Secure Database

UNIVERSITY OF Southampton
School of Electronics and Computer Science

sets : $OBJECT$ $DATA$ $USER$

constants : $LEVEL$

axioms :
$LEVEL = 1..10$

variables : $object$, $user$, $data$, $class$, $clear$

invariants :
$object \subseteq OBJECT$
$user \subseteq USER$
$data \in object \rightarrow DATA$ // REQ 7
$class \in object \rightarrow LEVEL$ // REQ 5
$clear \in user \rightarrow LEVEL$ // REQ 1

# Abstract Level SecureDB1 (Recall)

## Events for Secure Database

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- AddUser: REQ 1
- RemoveUser: REQ 4
- AddObject: REQ 2
- RemoveObject: REQ 3
- Read: REQ 8
- Write: REQ 8
- ChangeClearance: REQ 9
- ChangeClass: REQ 10

# Adding object ownership

UNIVERSITY OF Southampton
School of Electronics and Computer Science

## Extend the database specification with object ownerships

REQ 3 An existing object can be removed by its owner.

REQ 6 An object's classification is at most its owner's clearance level.

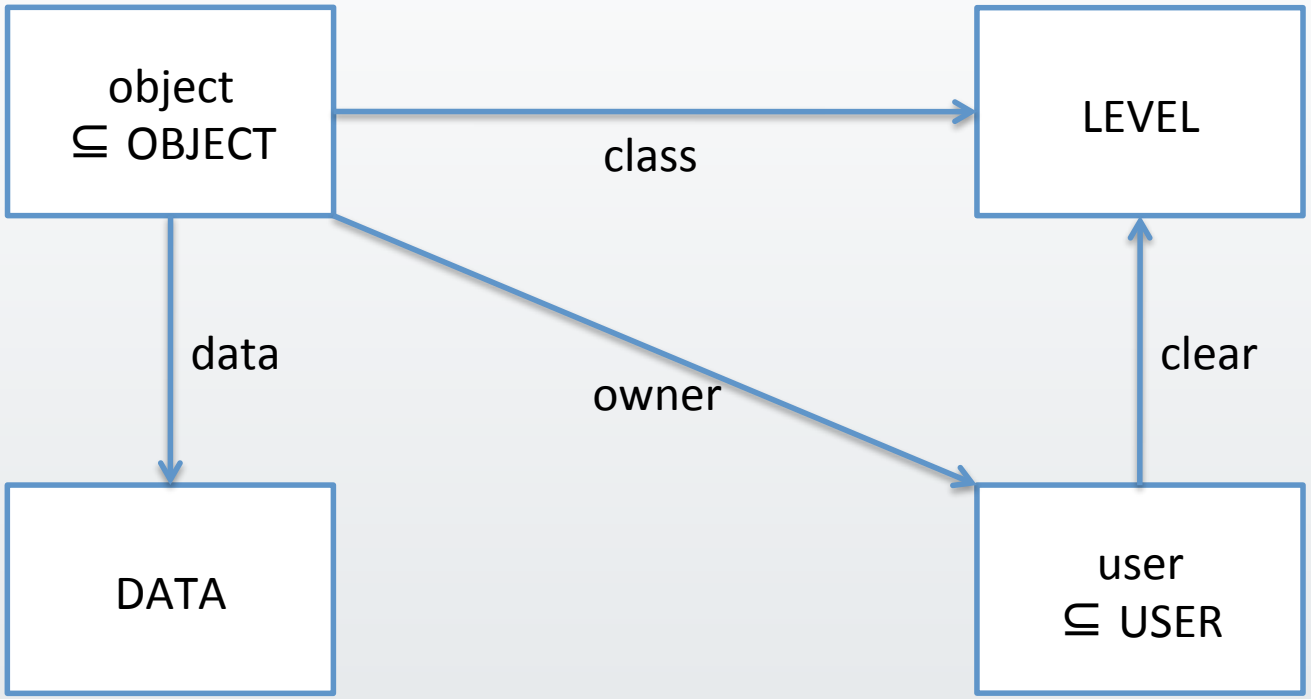
## Design consideration

- What additional variables are required?
- What events are affected?
  - Existing events
  - New events

# Concrete Level SecureDB2

## Class diagram with ownership

UNIVERSITY OF Southampton
School of Electronics and Computer Science



# Variables

UNIVERSITY OF Southampton
School of Electronics and Computer Science

variables : $object$, $user$, $data$, $class$, $clear$, $owner$

invariants :

$owner \in object \rightarrow user$

$\forall o \cdot o \in object \Rightarrow class(o) \leq clear(owner(o))$ // *REQ 6*

## Important

- For extension refinement, we must list all the variables:
  - those from SecureDB1 that we wish to retain,
  - new variables, e.g., *owner*
- We do not repeat invariants of SecureDB1 in SecureDB2

# Event AddUser

UNIVERSITY OF Southampton
School of Electronics and Computer Science

**AddUser**
  **any** $u, c$ **where**
    $u \in USER$
    $u \notin user$
    $c \in LEVEL$
  **then**
    $user := user \cup \{u\}$
    $clear(u) := c$
  **end**

Do we need to modify event AddUser?

# Event AddObject

UNIVERSITY OF Southampton
School of Electronics and Computer Science

**AddObject**
  any $o, d, c$ where
    $o \in OBJECT$
    $o \notin object$
    $d \in DATA$
    $c \in LEVEL$
  then
    $object := object \cup \{o\}$
    $data(o) := d$
    $class(o) := c$
  end

Do we need to modify event AddObject?

# Event AddObject

## Event Extension

UNIVERSITY OF Southampton
School of Electronics and Computer Science

AddObject **extended**
refines AddObject
any $u$ where
$u \in user$
$clear(u) \geq c$
then
$owner(o) := u$
end

AddObject
refines AddObject
any $o, d, c, u$ where
$o \in OBJECT$
$o \notin object$
$d \in DATA$
$c \in LEVEL$
$u \in user$
$clear(u) \geq c$
then
$object := object \cup \{o\}$
$data(o) := d$
$class(o) := c$
$owner(o) := u$
end

# Other events to consider

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Read
- Write
- ChangeClass
- ChangeClearance
- RemoveUser
- RemoveObject

## Another question

Do we need new events?
(Alternatively) What additional functionality the system should have?

# Concluding

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Abstraction vs. Refinement
- Event-B Refinement
  - Extension Refinement

# References I

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Thai Son Hoang. An introduction to the Event-B modelling method. In Alexander Romanovsky and Martyn Thomas, editors, *Industrial Deployment of System Engineering Methods*, pages 211–236. Springer-Verlag, July 2013. http://www.springer.com/computer/swe/book/978-3-642-33169-5.