UNIVERSITY OF Southampton
School of Electronics and Computer Science

10. Modelling Classes and Associations

Michael Butler and Thai Son Hoang

ECS, University of Southampton, U.K.

COMP1216 - Software Modelling and Design
11th March 2019

Objectives

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Relations/Functions in Event-B
- Modelling classes and associations

Properties of Relations

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Totality: Every element of S has *at least one relationship*
- Surjectivity: Every element of T has *at least one relationship*.
- Functional: Every element of S has *at most one relationship*
- Injectivity: Every element of T has *at most one relationship*

Different Types of Relations and Functions

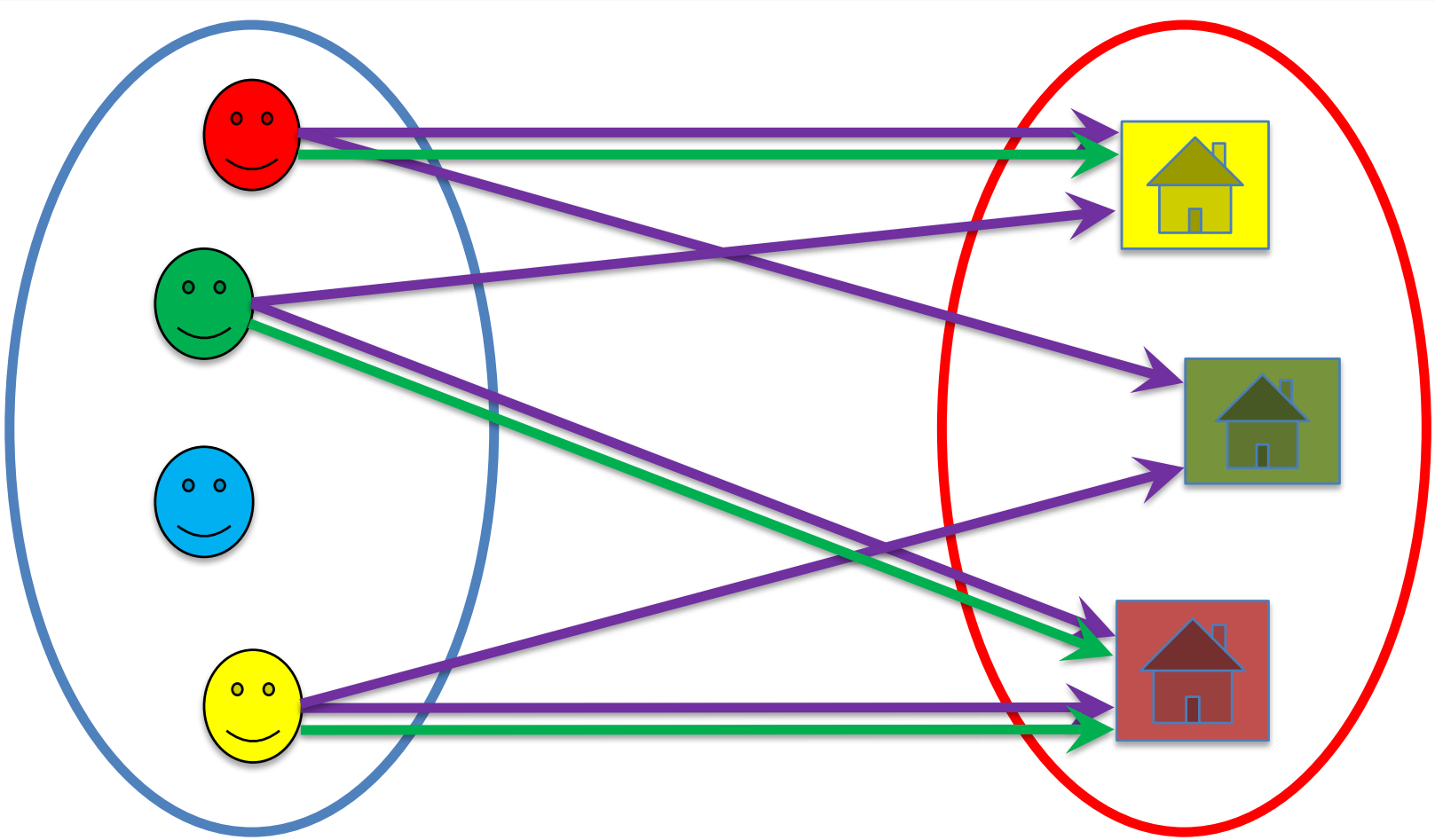
UNIVERSITY OF Southampton
School of Electronics and Computer Science

	None	Total	Surjective	Total $\wedge$ Surjective
None	$S \leftrightarrow T$	$S \nleftrightarrow T$	$S \twoheadleftarrow T$	$S \nleftrightarrow\!\!\!\!\twoheadleftarrow T$
Functional	$S \nrightarrow T$	$S \rightarrow T$	$S \nrightarrow\!\!\!\!\twoheadrightarrow T$	$S \twoheadrightarrow T$
Injective	–	–	–	–
Functional $\wedge$ Injective	$S \rightarrowtail T$	$S \rightarrowtail T$	–	$S \rightarrowtail\!\!\!\!\twoheadrightarrow T$

Table: Event-B relations

Diagram of building access

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Location $\subseteq$ Permission

Class diagram abstraction

UNIVERSITY OF Southampton
School of Electronics and Computer Science


```
1 invariants
2   @typeof−permission: permission ∈ USER ↔ BUILDING // many−to−many
3   @typeof−location: location ∈ USER ⇸ BUILDING
```

Classes and Associations

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Consider our model of a birthday book:

1 **variables**
2 person
3 birthday
4 **invariants**
5 @typeof−person: person $\subseteq$ PERSON
6 @typeof−birthday: birthday $\in$ person $\rightarrow$ DATE

Representing birthday as a simple class diagram:

Classes and Associations (2/2)

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Consider our model of a birthday book:

```
1 variables
2   person
3   birthday
4 invariants
5   @typeof−person: person ⊆ PERSON
6   @typeof−birthday: birthday ∈ person → DATE
```

Making the variable set explicit

Multiple Associations

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Suppose we want to model a person's address as well.
- Multiple attributes of an entity (e.g., person) are modelled as separate total functions on the same domain:

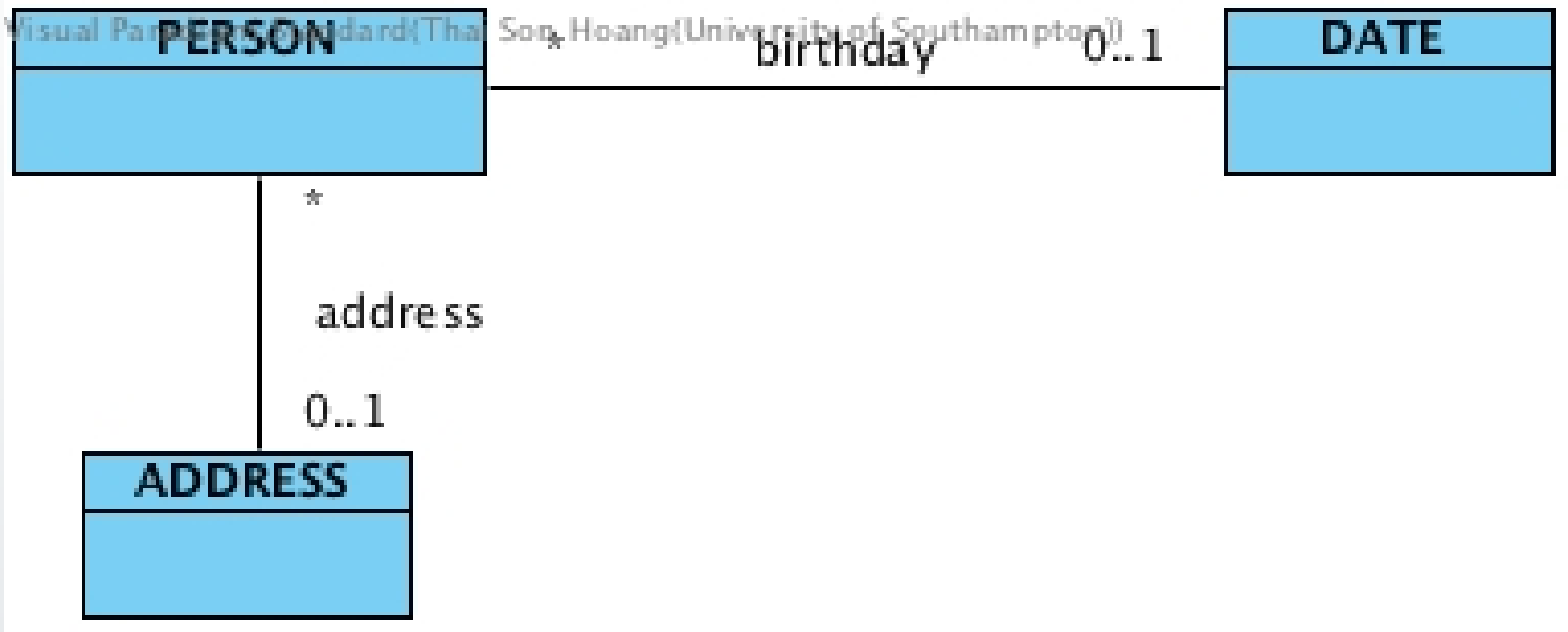
---

1 **variables**
2 person
3 birthday
4 address
5 **invariants**
6 @typeof−person: $person \subseteq PERSON$
7 @typeof−birthday: $birthday \in person \rightarrow DATE$
8 @typeof−address: $address \in person \rightarrow ADDRESS$

- The common domain for both functions means every element of the set person, has both a birthday and an address.

Class diagram for the birthday/address book

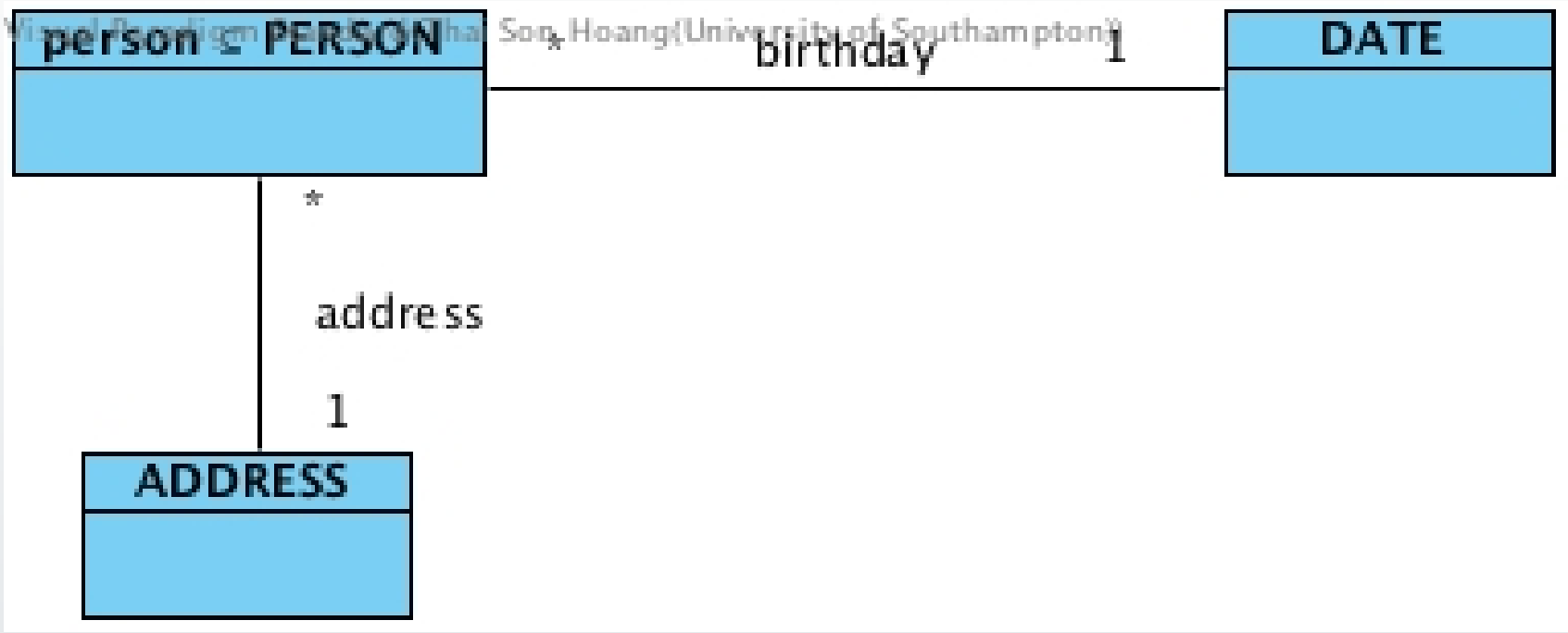
UNIVERSITY OF Southampton
School of Electronics and Computer Science



Class diagram for the birthday/address book

Making the variable set explicit

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Secure database example

We consider a secure database. Each object in the database has a data component.

Each object has a classification between 1 and 10.

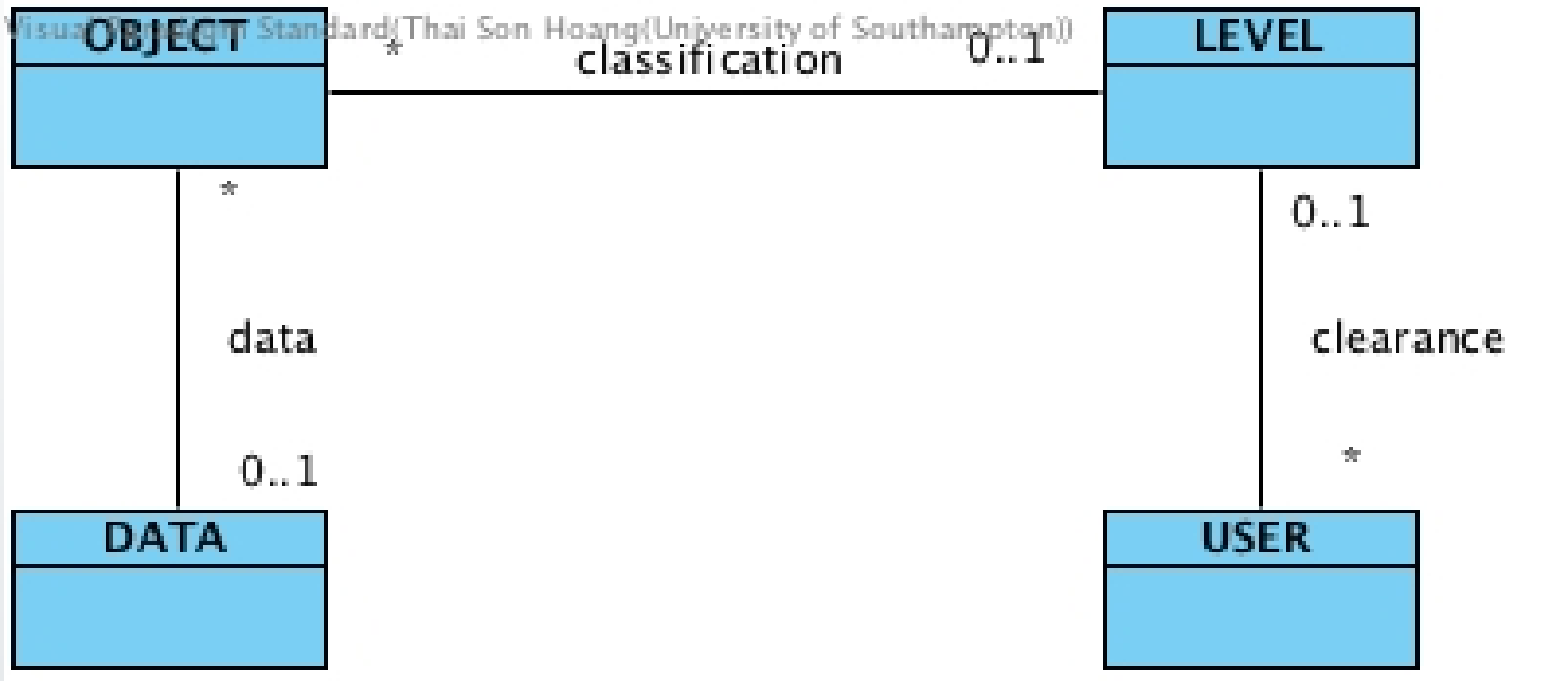
Users of the system have a clearance level between 1 and 10.

Users can only read and write objects whose classification is no greater than the user's clearance level.

What are the *entities*, *associations*, *events*?

Class diagram for secure database

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Making variable set explicit

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Introduce set variables:
- *object* is the variable set of objects managed by the system
- *user* is the variable set of users managed by the system

Primary and secondary carrier sets

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Note that we do **not** introduce set variables for *LEVEL* nor *DATA*

OBJECT and *USER* are *primary* carrier sets: the systems should enable management of these entities (creation, modification of attributes, deletion)

LEVEL and *DATA* are *secondary* carrier sets: serve as attributes of primary carrier sets

Types and variables


```
1  sets OBJECT DATA USER
2  constants LEVEL
3  axioms
4   @def−LEVEL: LEVEL = 1 .. 10
5
6  variables object user data class clear
7  invariants
8   @typeof−object: object ⊆ OBJECT
9   @typeof−user: user ⊆ USER
10  @typeof−data: data ∈ object → DATA
11  @typeof−class: class ∈ object → LEVEL
12  @typeof−clear: clear ∈ user → LEVEL
```

The invariant $data \in object \rightarrow DATA$ means that $data(o)$ is well-defined whenever $o \in object$. Why is this important?

```
1  INITIALISATION
2  then
3   object, user, data, class, clear := ∅, ∅, ∅, ∅, ∅
4  end
```

Adding users


```
1  event AddUser
2  any u c where
3   @grd1: u ∉ user
4   @grd2: c ∈ LEVEL
5  then
6   @act1: user := user ∪ {u}
7   @act2: clear(u) := c
8  end
```

The new user must not already exist.
We need to provide the initial clearance level for the new user.

Adding objects


```
1  event AddObject
2  any o d c where
3   @grd1: o ∉ object
4   @grd2: d ∈ DATA
5   @grd3: c ∈ LEVEL
6  then
7   @act1: object := object ∪ {o}
8   @act2: data(o) := d
9   @act3: class(o) := c
10 end
```

The new object must not already exist.
We need to provide the initial classification level and data value for the new object.

Reading objects

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
1  event Read
2  any u o result where
3   @grd1: u ∈ user // The user must exist
4   @grd2: o ∈ object // The object must exist
5   @grd3: clear(u) ≥ class(o) // The clearance must be OK
6   @grd4: result = data(o) // The data associated with the object
7  end
```

Writing objects


```
1  event Write
2  any u o d where
3   @grd1: u ∈ user
4   @grd2: o ∈ object
5   @grd3: clear(u) ≥ class(o)
6   @grd4: d ∈ DATA
7  then
8   @act1: data(o) := d
9  end
```

The write operation overwrites the data value associate with the object with a new value.

Changing classification and clearance levels

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
1 event ChangeClass
2 any o c where
3   @grd1: o ∈ object
4   @grd2: c ∈ LEVEL
5 then
6   @act1: class(o) := c
7 end
```

```
1 event ChangeClear
2 any u c where
3   @grd1: u ∈ user
4   @grd2: c ∈ LEVEL
5 then
6   @act1: clear(u) := c
7 end
```

Making classification changes more secure

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Include constraints on the user who is changing the object classification.

```
1  event ChangeClass
2  any o c u where
3   @grd1: o ∈ object
4   @grd2: c ∈ LEVEL
5   @grd3: u ∈ user
6   @grd4: clear(u) ≥ class(o)
7   @grd5: clear(u) ≥ c
8  then
9   @act1: class(o) := c
10 end
```

Making clearance changes more secure

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Include constraints on the user who is changing the object classification.

```
1  event ChangeClear
2  any u c a where
3   @grd1: u ∈ user
4   @grd2: c ∈ LEVEL
5   @grd3: a ∈ user
6   @grd4: clear(a) ≥ clear (u)
7   @grd5: clear(a) ≥ c
8  then
9   @act1: clear(u) := c
10 end
```

Removing users and objects

UNIVERSITY OF Southampton
School of Electronics and Computer Science

```
1 event RemoveUser
2 any u where
3  @grd1: u ∈ user
4 then
5  @act1: user := user \ {u}
6  @act2: clear := {u} ⩤ clear
7 end
```

```
1 event RemoveObject
2 any o where
3  @grd1: o ∈ object
4 then
5  @act1: object := object \ {o}
6  @act2: class := {o} ⩤ class
7  @act3: data := {o} ⩤ data
8 end
```

Adding object ownership

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Extend the database specification so that each object has an owner.

The clearance associated with that owner must be at least as high as the classification of the object.

Only the owner of an object is allowed to delete it.

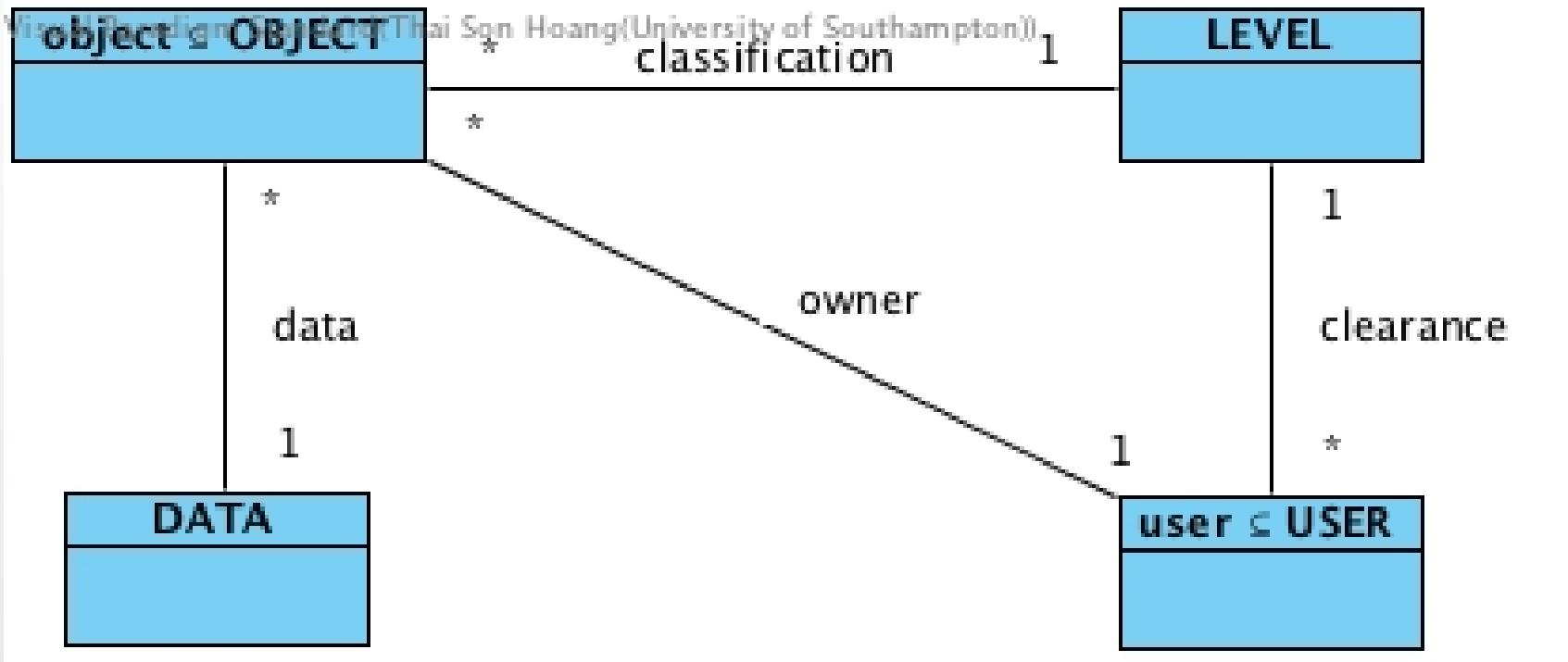
A user's clearance level can only be modified to a new level by another user whose clearance level is at least as high as the new clearance level.

What additional variables are required?

What events are affected?

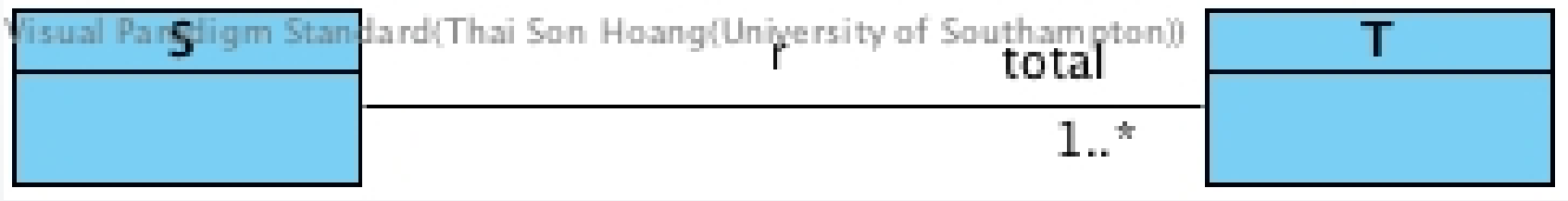
Class diagram with ownership

UNIVERSITY OF Southampton
School of Electronics and Computer Science



Total

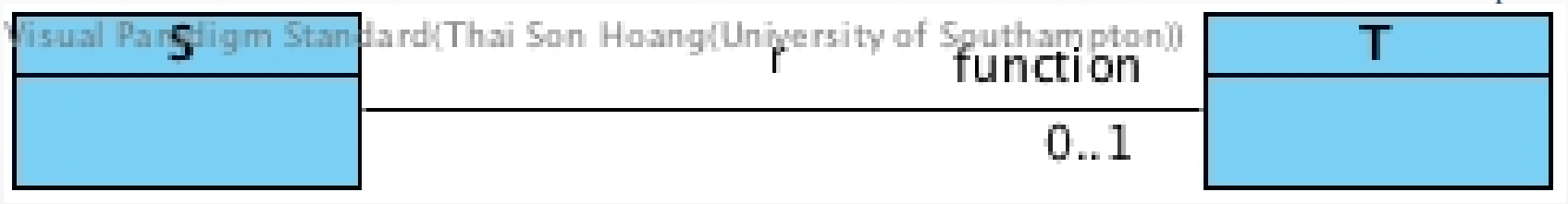
UNIVERSITY OF Southampton
School of Electronics and Computer Science



(We consider the relation from S to T)

Functional

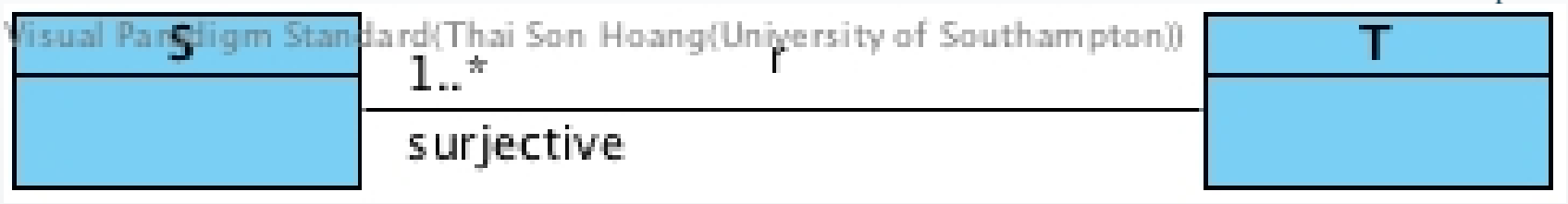
UNIVERSITY OF Southampton
School of Electronics and Computer Science



(We consider the relation from S to T)

Surjective

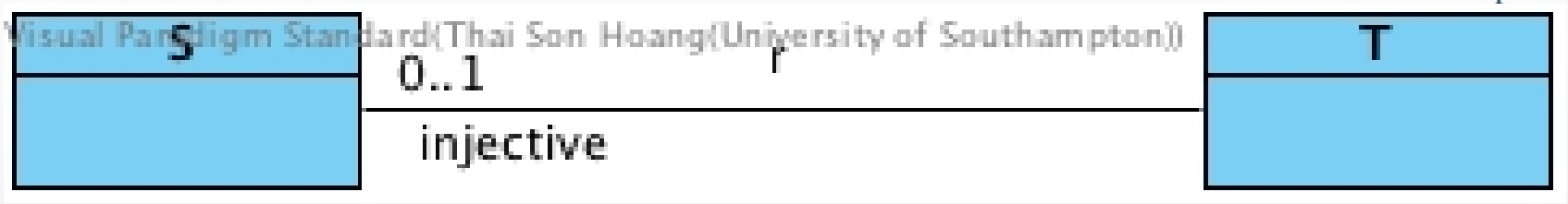
UNIVERSITY OF Southampton
School of Electronics and Computer Science



(We consider the relation from S to T)

Injective

UNIVERSITY OF Southampton
School of Electronics and Computer Science



(We consider the relation from S to T)

Combination

Total functions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

(We consider the relation from S to T)

Combination

Injective functions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

(We consider the relation from S to T)

Combination

UNIVERSITY OF Southampton
School of Electronics and Computer Science

How about

- total surjective relations?
- total injective functions?
- surjective functions?
- bijections?
- ...

Summary

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Properties of relations in Event-B
  - Total, functional, surjective, injective
- Model classes using sets
- Model associations using relations and functions