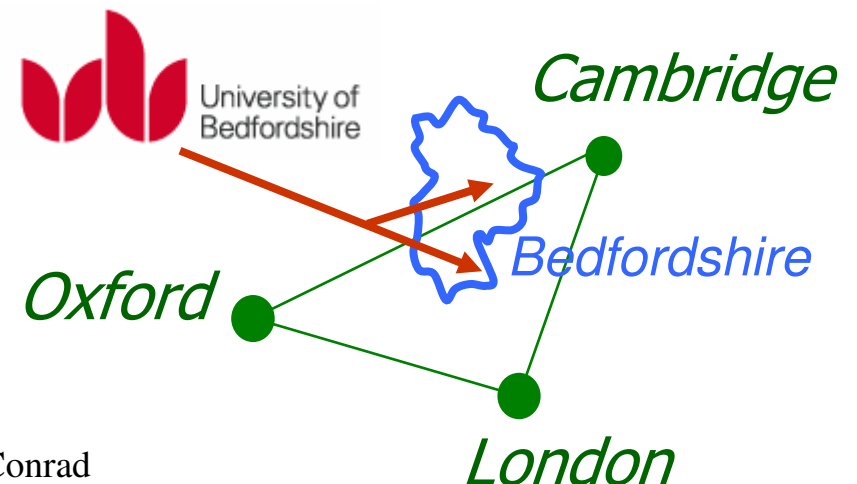
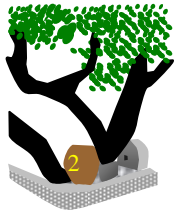


Enhancing the UML with Shadows for Agile Development

1. Who?
2. What?
3. Where?
4. Why?
5. How?

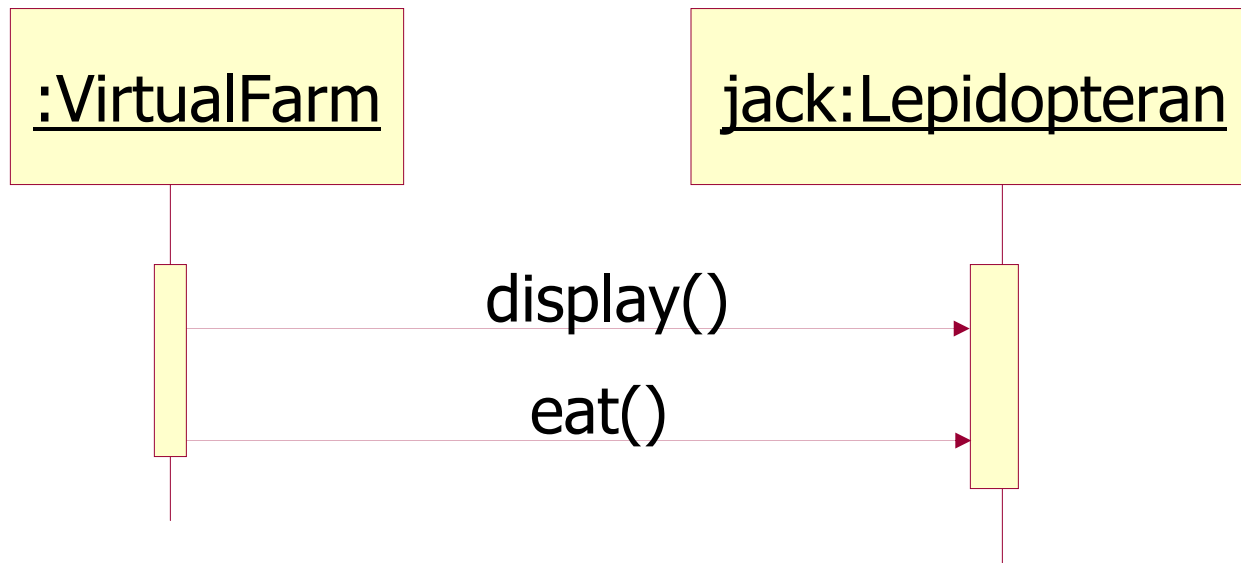
- *Marc Conrad*
 - *University of Bedfordshire*
 - <http://perisic.com/marc>
- *Marianne Huchard*
 - *Université Montpellier II*
 - <http://www.lirmm.fr/~huchard>





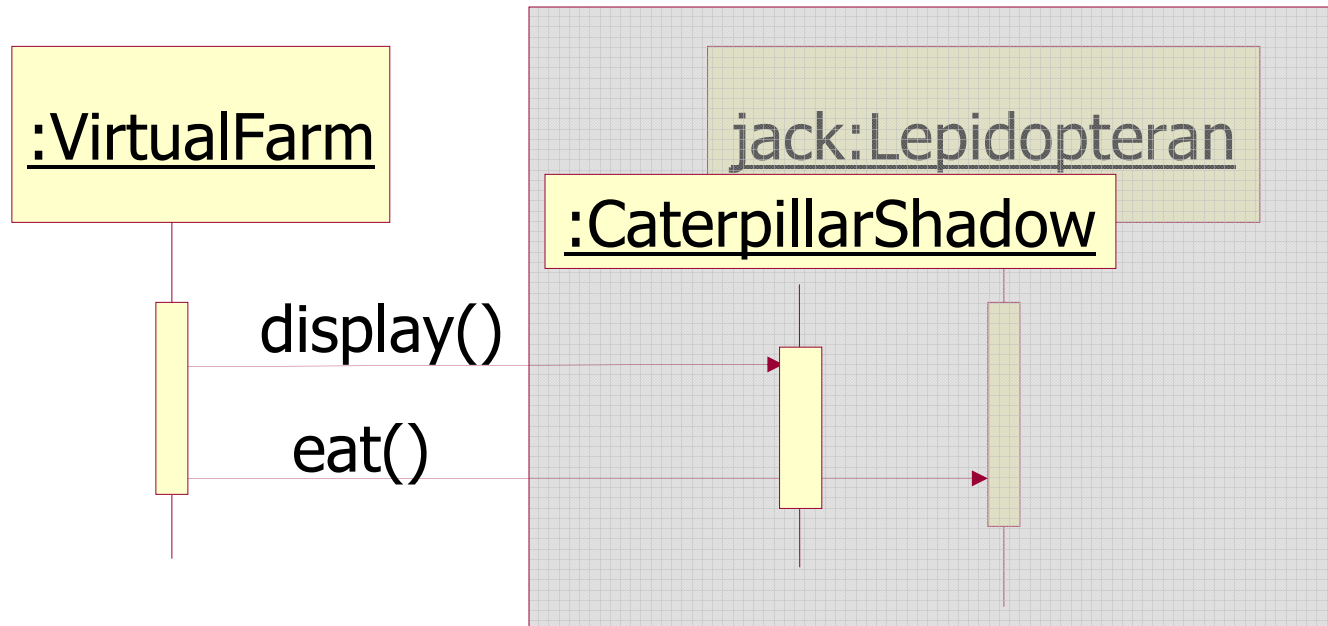
Shadows - what?

- Consider this simple sequence diagram.

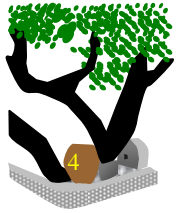




Shadows - what?



- The **CaterpillarShadow** object shadows **jack**, the **Lepidopteran** object. Messages intended for **jack** are intercepted by **:CaterpillarShadow**



Code Example: Lepidopteran

(nothing exciting here)

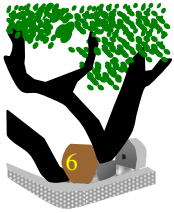
```
class Lepidopteran {  
    boolean full = false;  
    void eat() { full = true; }  
    String display() {  
        if( full ) {  
            return "She is full.";  
        } else {  
            return "She is hungry.";  
        }  
    }  
}
```



Code Example: CaterpillarShadow

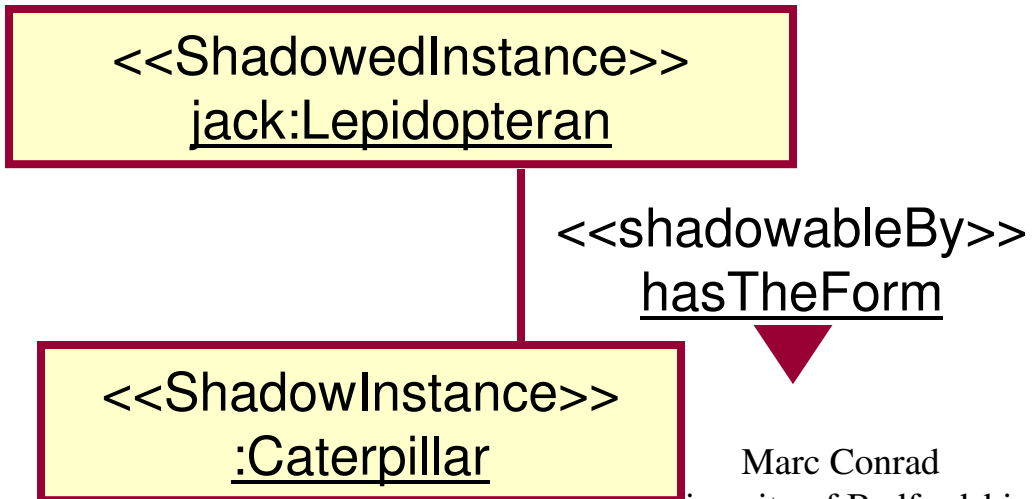
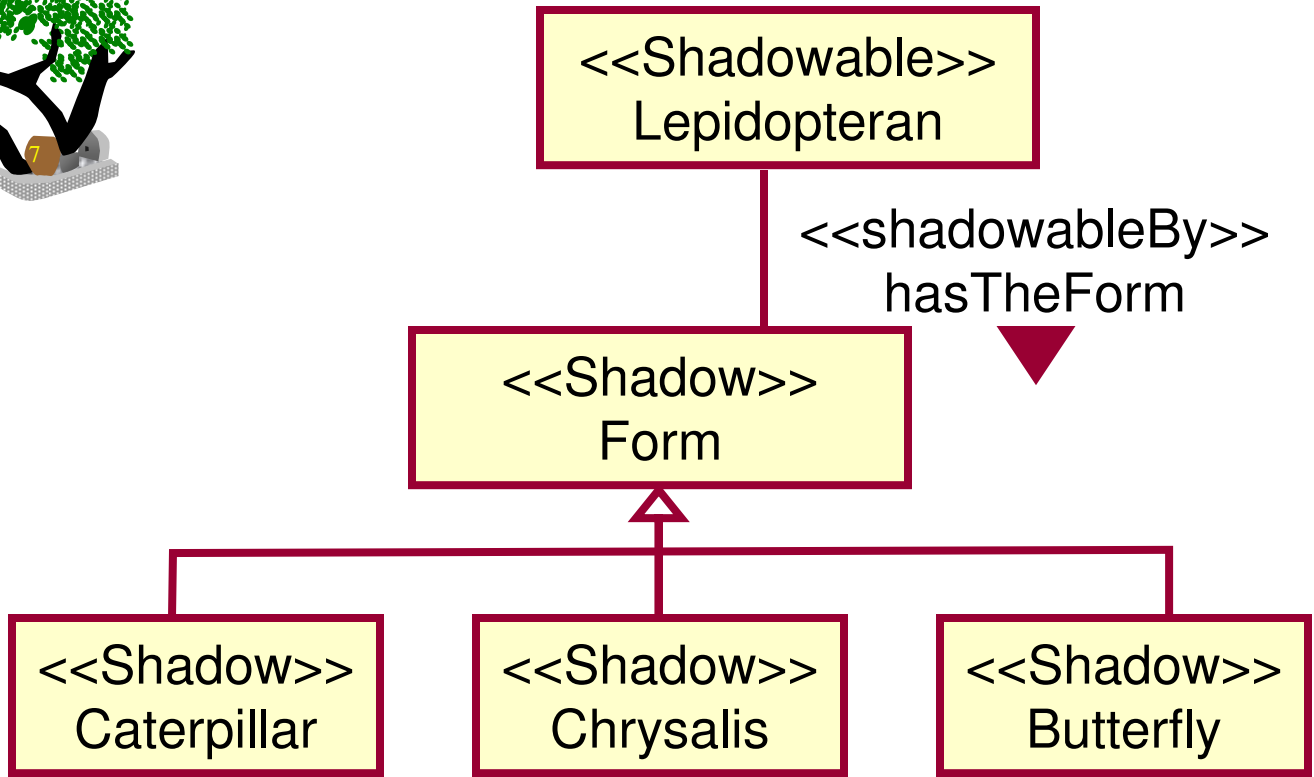
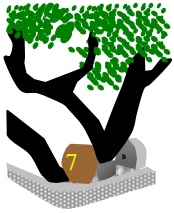
```
class CaterpillarShadow extends Shadow {
    String display() {
        return "She is very hairy; "+
            shadowOwner().display();
    }
}

main() {
    Lepidopteran jack = new Lepidopteran();
    CaterpillarShadow cat = new CaterpillarShadow();
    jack.addShadow(cat);
    jack.display(); // Jack looks like a caterpillar.
    jack.eat();     // Jack eats.
}
```



Code Example: Shadows in Action (dynamic re-classification)

```
main() {  
    Lepidopteran jack = new Lepidopteran();  
    CaterpillarShadow cat = new CaterpillarShadow();  
    jack.addShadow(cat);  
  
    jack.display(); // Jack looks like a caterpillar.  
  
    ChrysalisShadow chr = new ChrysalisShadow();  
    jack.removeShadow(cat);  
    jack.addShadow(chr);  
  
    jack.display(); // Jack looks like a chrysalis.  
}
```



Example:
Dynamic classification
(notation with explicit
stereotypes)



Shadows - where?

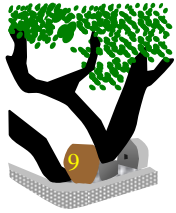
- **LPC**

- For LPMuds (text-based online multi-user games)
- E.g. Unitopia: <http://unitopia.rus.uni-stuttgart.de>; Since 1993. Lots of examples. Standard technique to implement new features.

- **Java**

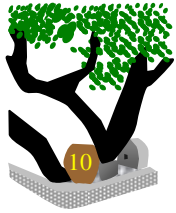
- Prototype implementation using the jikes Java compiler; <http://www.perisic.com/shadow>.
- Presented & discussed at ECOOP 2004 workshop.

- Similar to “posing” in **Objective-C**.



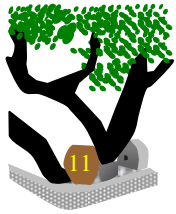
Shadows – why?

- Shadows enhance the unanticipated evolution of written programs.
 - Prototyping & Deprecation, temporary features for objects and classes.
- Help to structure software.
 - Distinct features of an object can be separated into different packages.
- Also: Implementation of non-standard features.
 - Interclassing, re-classification, multiple inheritance, roles, ...



Shadows – why?

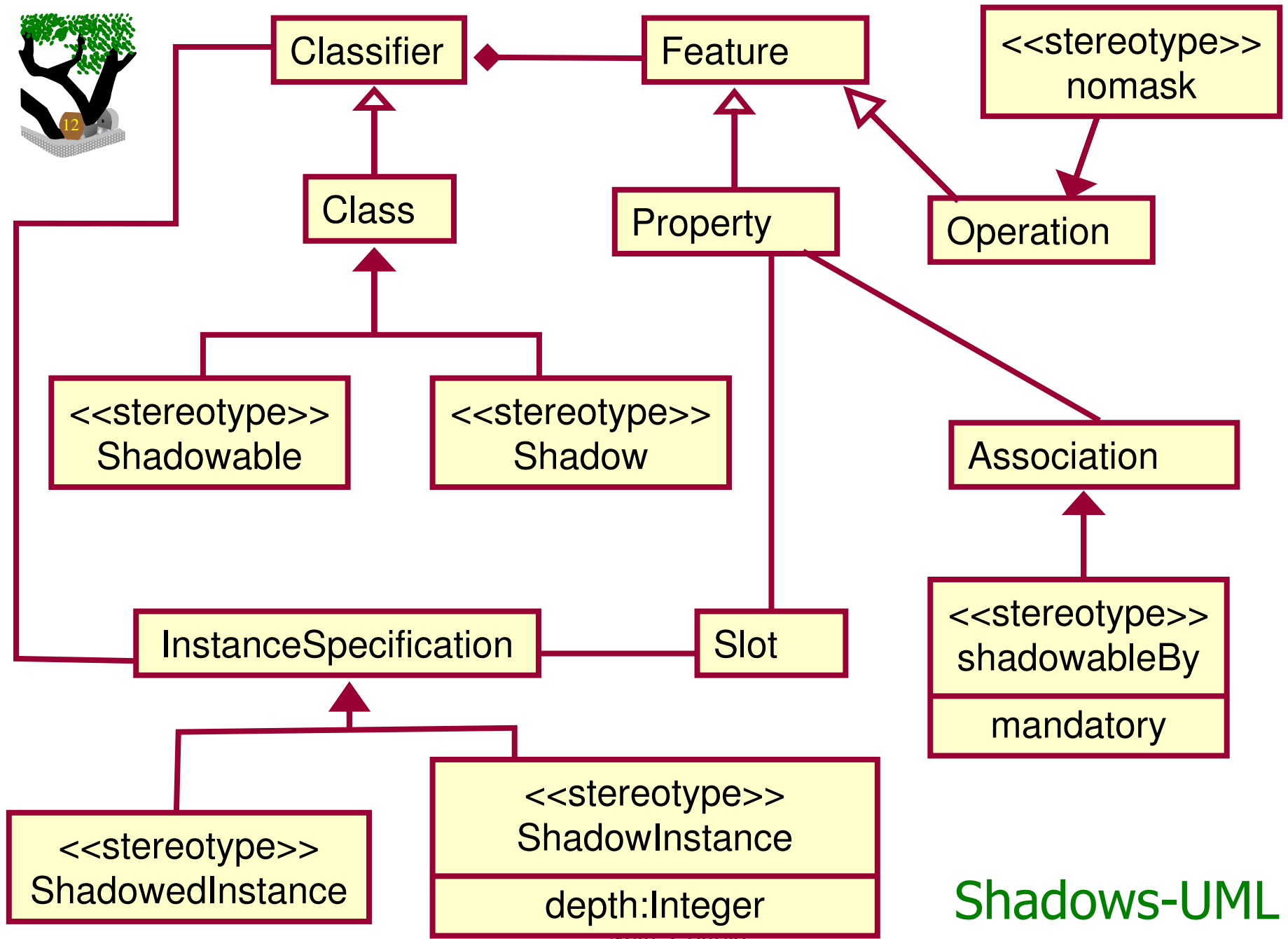
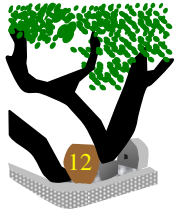
- Impact on Agile Methods
 - Speculation phase in **Adaptive Software Development**.
 - Prototyping in exploration phase of **XP**.
 - **Feature Driven Development**.
 - ...



Shadows – How?

- Model-driven software development
- Agile methodologies using the UML
- (Automatic) Design-to-Code translation
- PIM to PSM translation

Integration of shadows in the UML (via the UML 2.0 Superstructure Specification)



Shadows-UML Profile

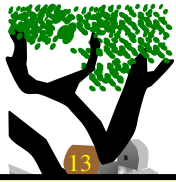


Table 1: Stereotype Definition

Stereotype	BaseClass	Parent	Tags	Constraints	Description
Shadowable	Class	N/A			Instances of shadowable classes can be associated with an ordered set of shadows.
Shadow	Class	N/A		Shadow and shadowable stereotypes cannot be applied on the same class.	Instances are used to enhance and shadow instances of Shadowable classes.
Shadowed-instance	Instance-Specification	N/A			Instance of a Shadowable class that admits shadows.
Shadow-instance	Instance-Specification	N/A	depth		Instance of a Shadow class
shadowableBy	Association	N/A	mandatory		Associates a shadowable class and a shadow class.
nomask	Operation	N/A			An operation that cannot be shadowed

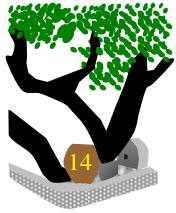
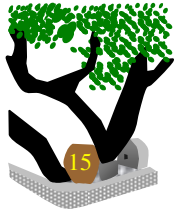


Table 2: Tag Definition

Tag	Stereotype	Type	Multiplicity	Description
mandatory	shadowableBy	Boolean	1	True iff every instance of the shadowable object has a shadow of the target class.
depth	ShadowInstance	Integer	1	Indicates the depth of the shadow (in relation to other shadows applied to the object).



Summary

- Who: MC & MH
- What: Shadows – a useful feature
- Where: LPC & Java
- Why: Agile Development
- How: UML

Thank you / Merci beaucoup / Vielen Dank