D^2AL

A Design-based Distribution Aspect Language

Ulrich Becker
University of Erlangen-Nuremberg
Department of Computer Science IV

Introduction

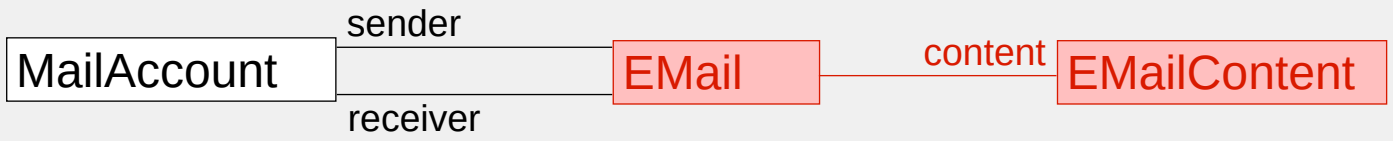
- **Motivation**
  - optimized distribution required for efficient distributed applications
  - software developer should have explicit control
- ***D²AL* is an aspect language for object distribution**
  - improve application performance through...
    - collocation of collaborating objects
    - migration of objects
    - replication of immutable objects
- ***D²AL* is based on the UML design model of the application**
  - design contains distribution-relevant information that is lost in implementation
    - static structure: associations
    - behaviour: state machines

Collaborations (1)

Collaborations are the basis for object collocation

- collaborations are pairs of objects
- related by an association

EMail class diagram

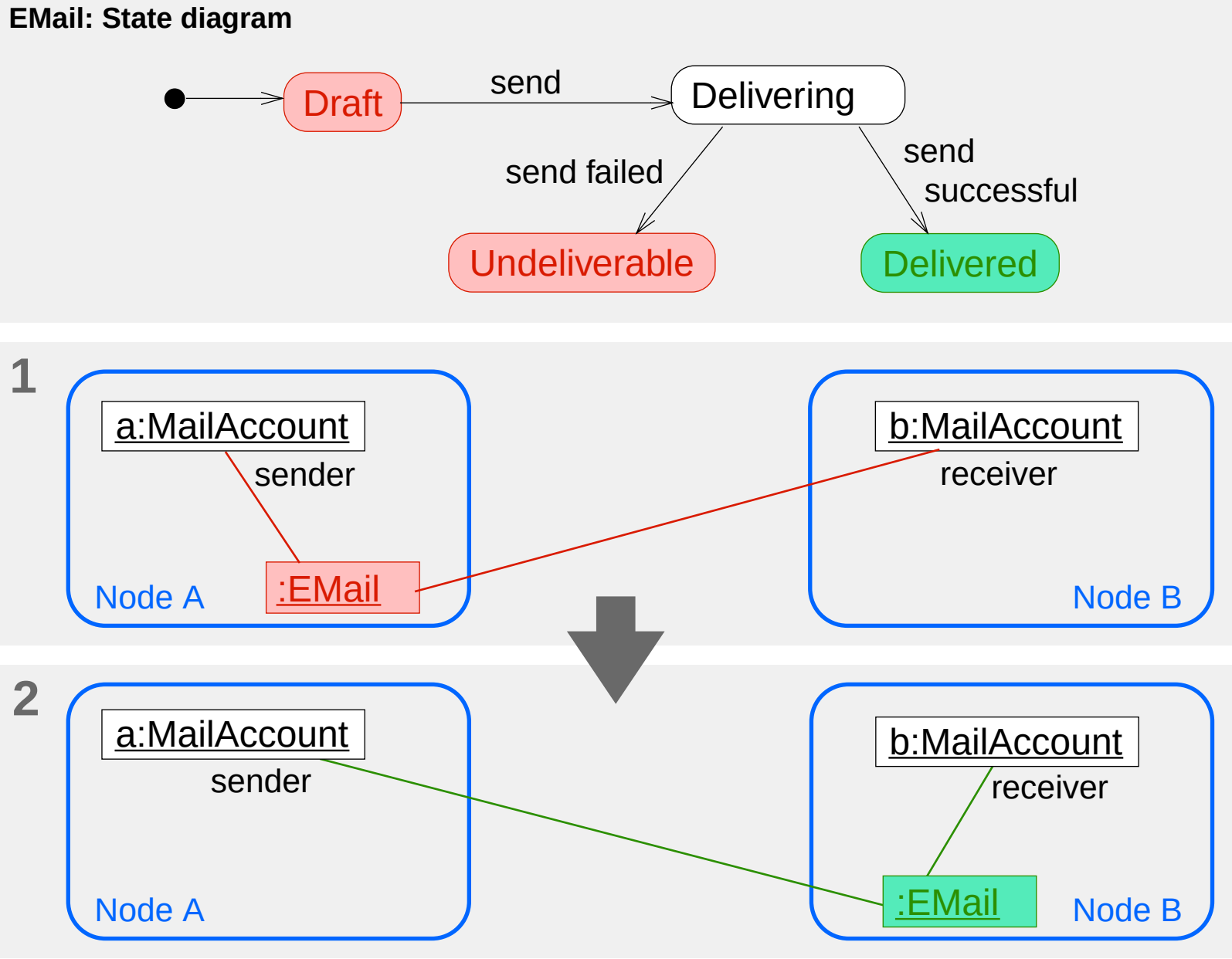


***D²AL* statement for collocation**

```
collaboration EMailCluster {
   link content;
   distribution collocation;
}
```

- collaborations can be limited to abstract states

Collaborations (2)

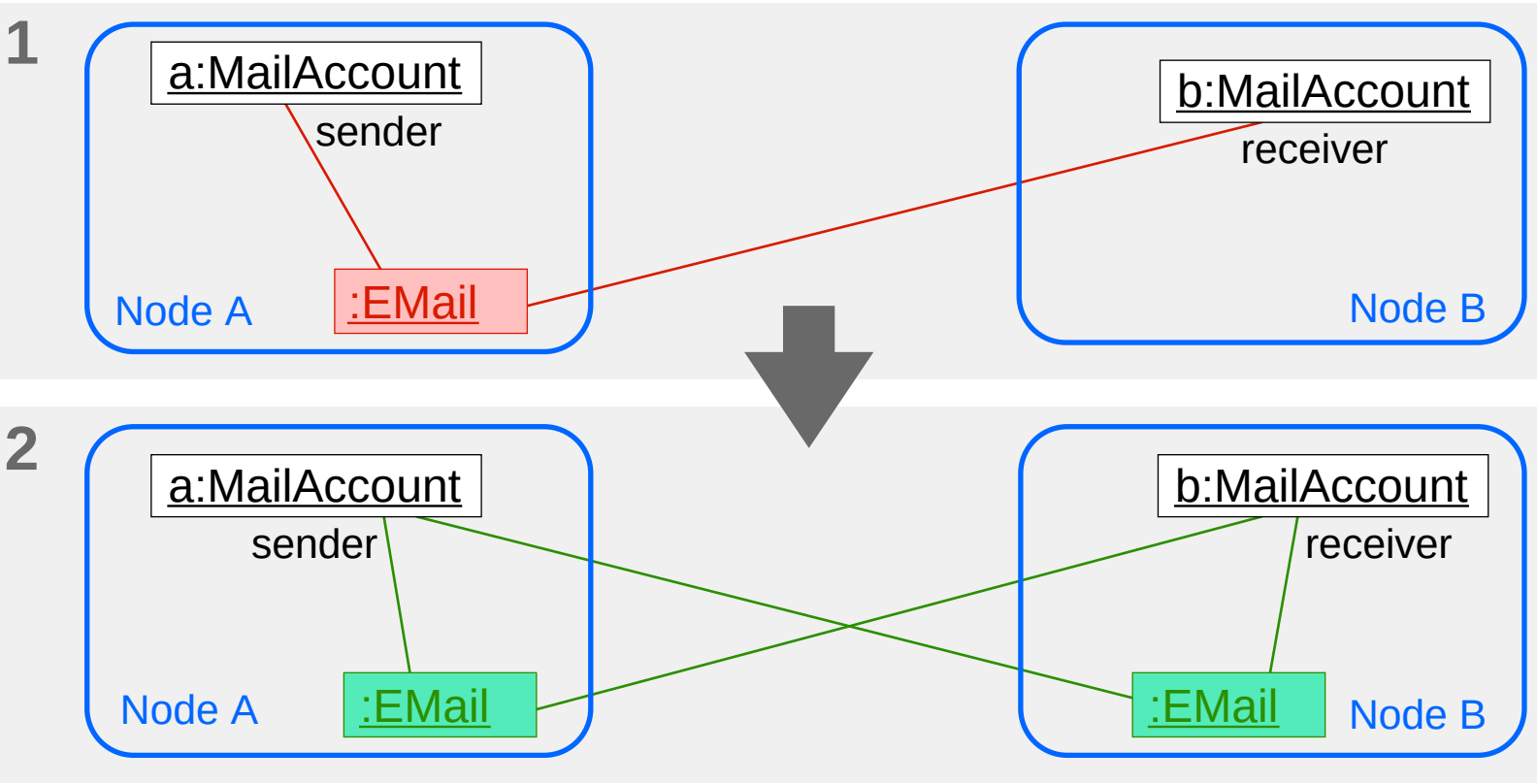


Replication

- **Abstract states can be declared *replicable***
- **Decision whether to replicate is left to runtime system**

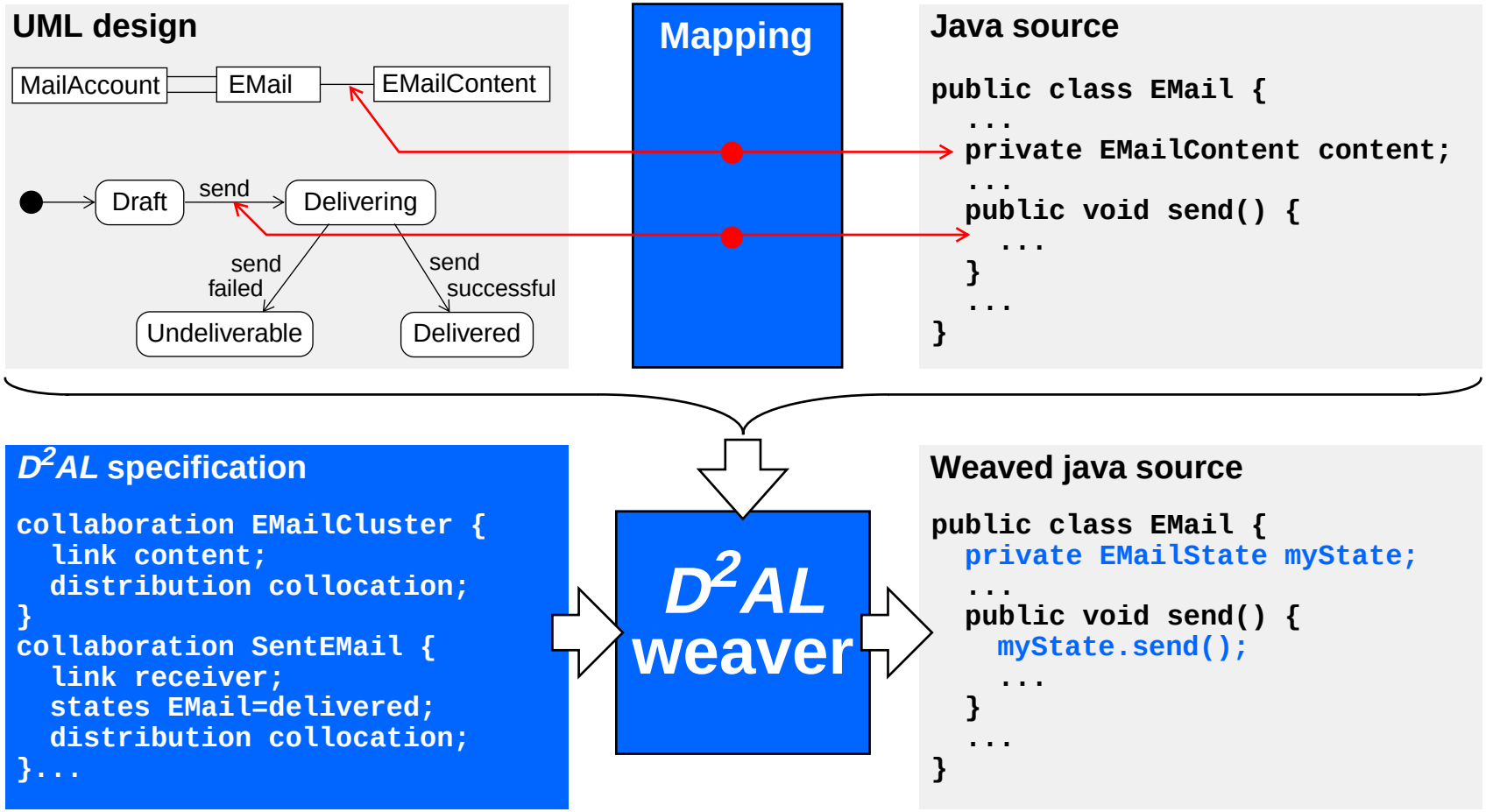
***D²AL* statement for replication**

```
states EMail {
  replicable {Undeliverable, Delivered};
}
```



Weaver

- **Weaved code contains classes and statements to…**
  - maintain design information at runtime (associations, abstract states)
  - make distribution decisions based on this information



Conclusion

- **Advantages of *D²AL***
  - relationships and abstract states from the model can be used
    - increased expressiveness
    - no need to manually maintain abstract state in implementation
  - *D²AL* specification is less prone to implementation changes
- **Current work**
  - prototype implementation
- **Future work**
  - validate suitability of design-based distribution specification
  - consider other model-elements, e.g. interaction diagrams