TECHNISCHE UNIVERSITÄT DRESDEN

ROSI

Department of Computer Science Institute of Systems Architecture, Computer Networks Group

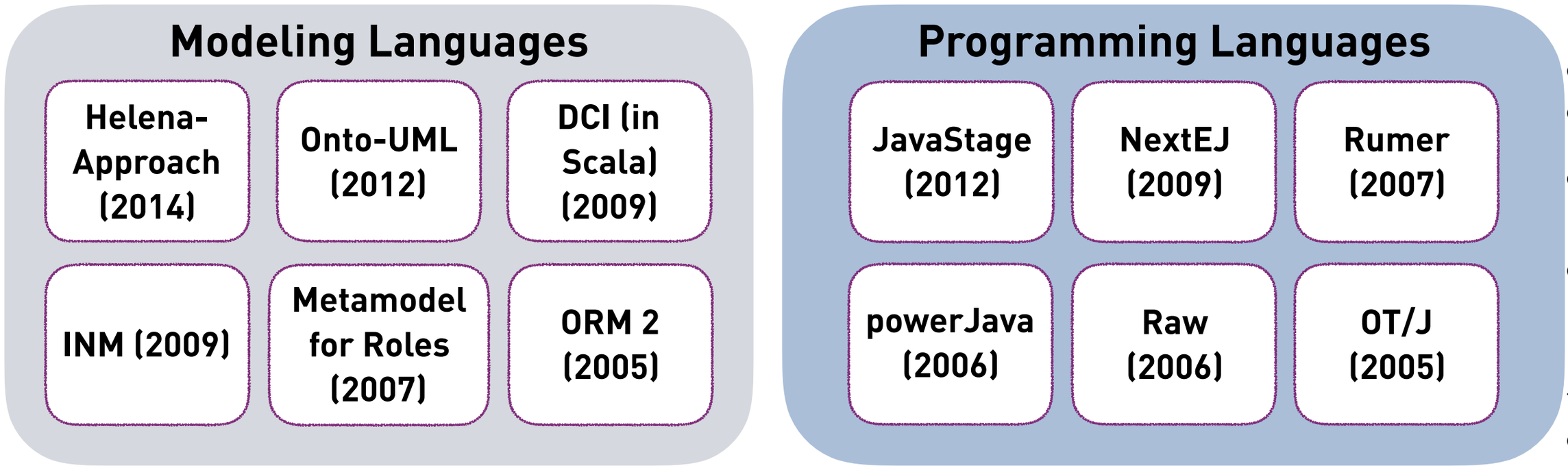
Position Paper: Runtime Model for Role-based Software Systems

Tobias Jäckel[*], Martin Weißbach[†], Kai Herrmann[*], Hannes Voigt[*], Max Leuthäuser[‡]

[*] Database Systems Group
[†] Computer Networks Group
[‡] Software Technology Group

DFG

Motivation



- roles allows for an intuitive separation of static and dynamic, context-dependent behavior
- investigation of roles throughout all steps of the software development process and across all layers of developed software applications

Modeling & Programming Languages taken from: T. Kühn, et al., "A Metamodel Family for Role-Based Modeling and Programming Languages," in Software Language Engineering, Cham: Springer International Publishing, 2014, pp. 141–160.

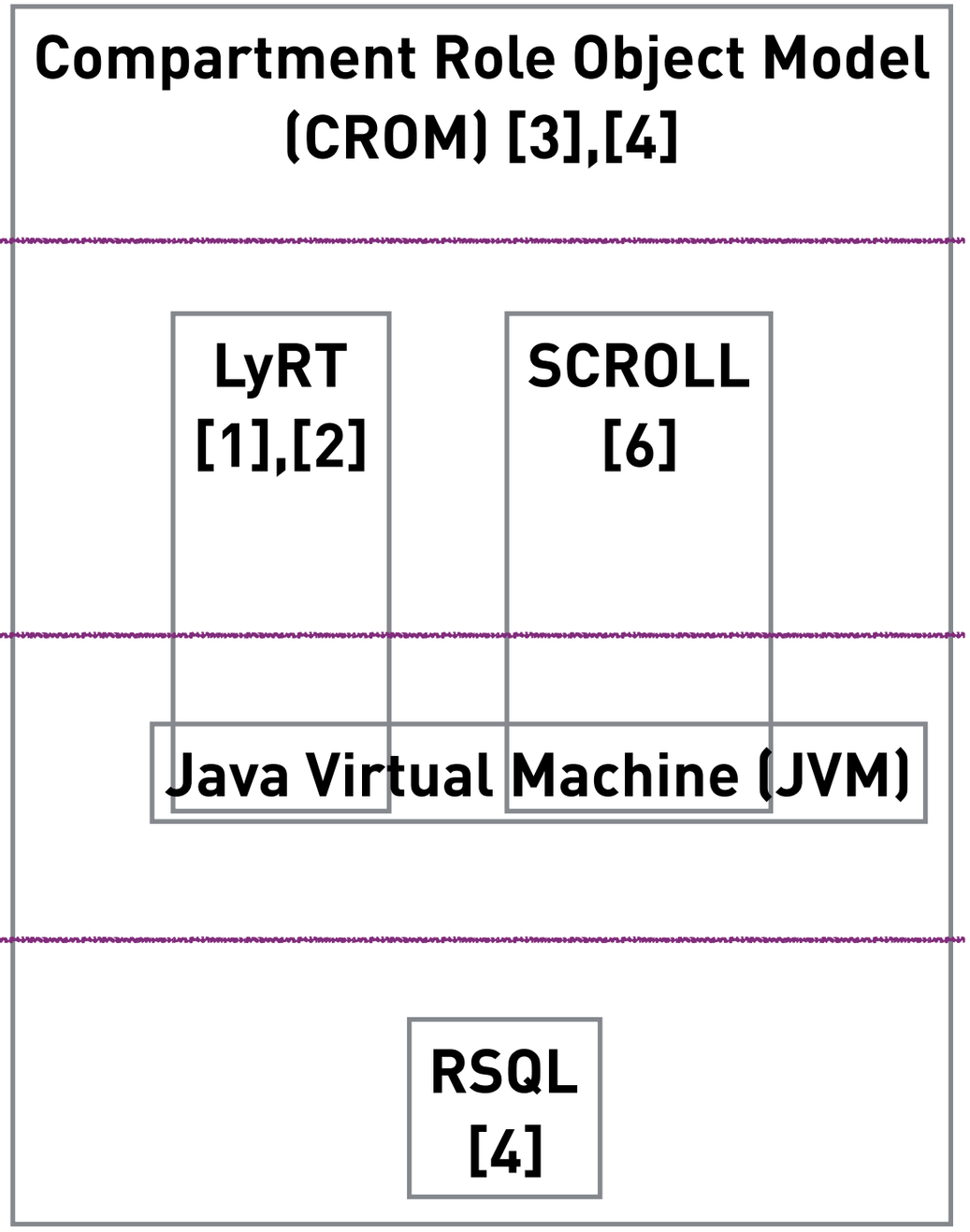
Motivation

Formal Foundation

Programming Language / Framework

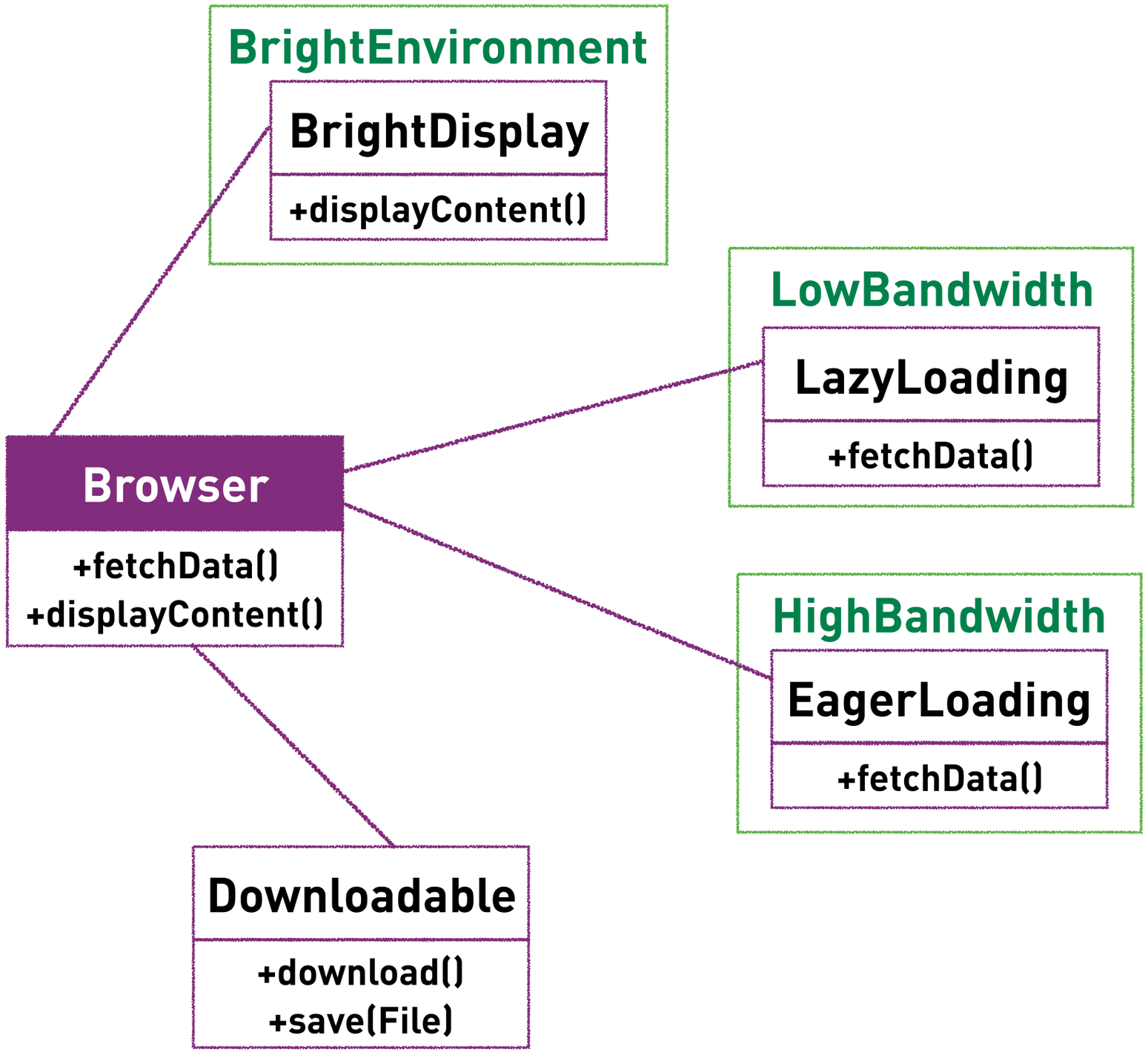
Runtime Environment

Database



Our Assumed Role Model

Our Assumed Role Model



objects may acquire and abandon roles dynamically

objects may play different roles simultaneously

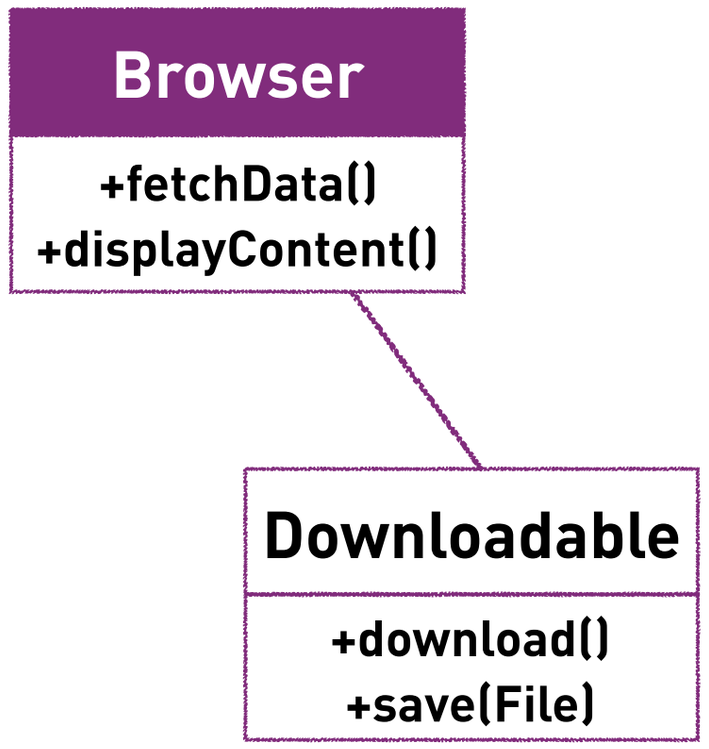
object can be role-specific

objects may play different roles simultaneously

features of an object can be role-specific

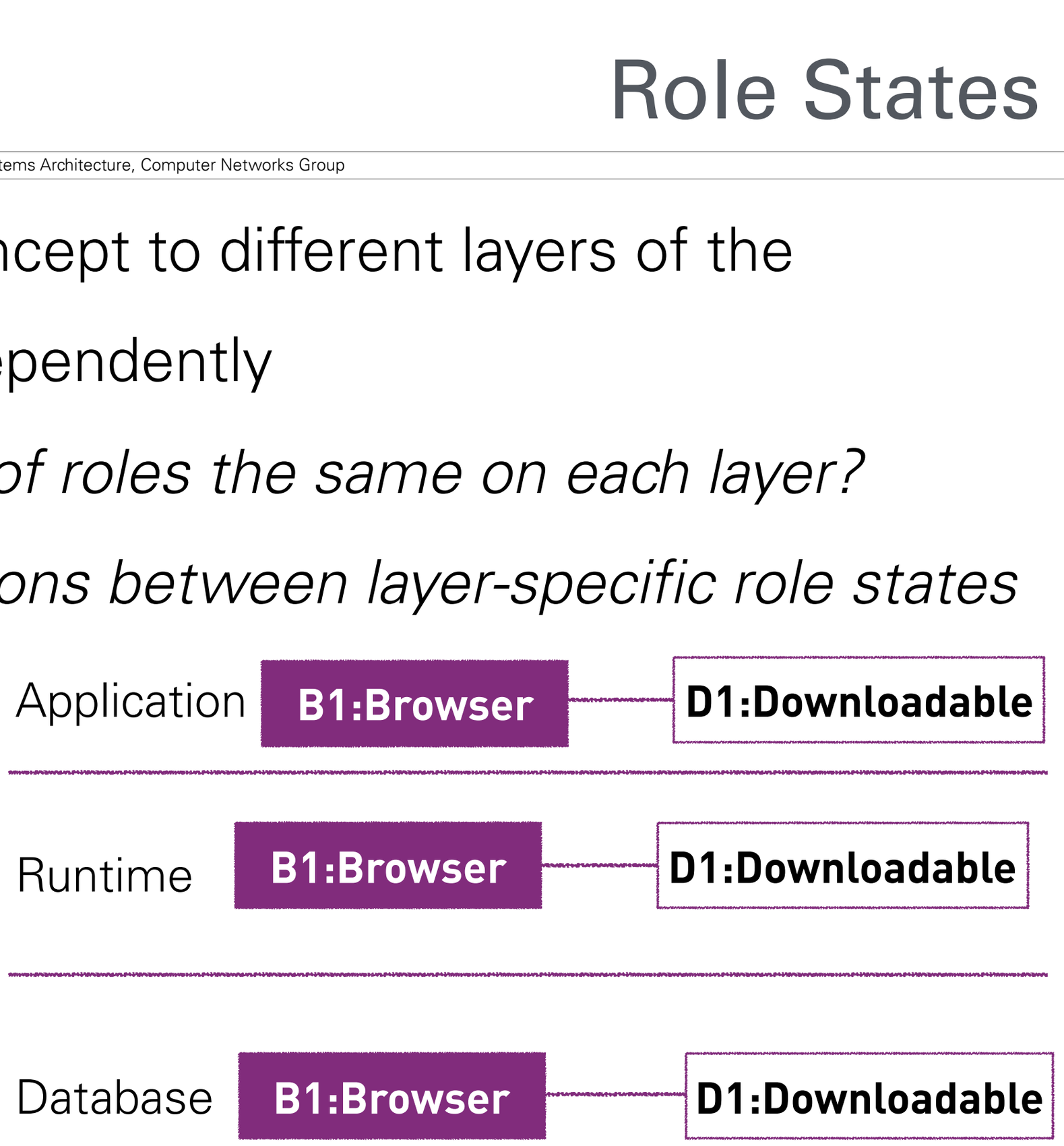
Our Assumed Role Model

- Roles have properties and behaviors
- *objects may play different roles simultaneously*
- *objects may acquire and abandon roles dynamically*
- roles can be transferred between objects
- the state of an object can be role-specific
- *features of an object can be role-specific*
- an object and its roles share identity
- an object and its roles have different identity
- roles depend on compartments
- compartments have their own identity



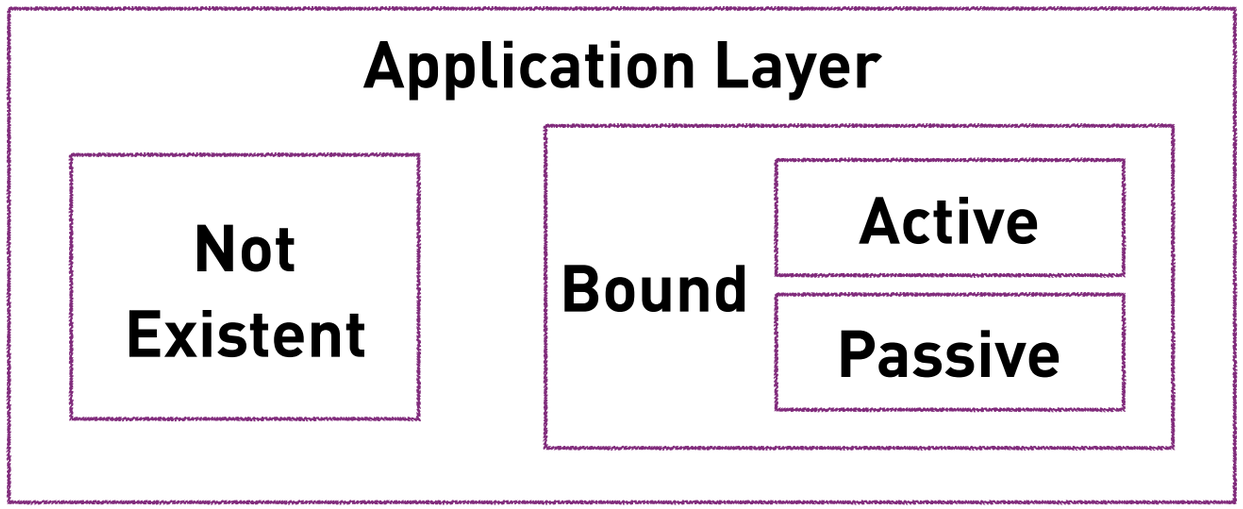
Role States

- applied role concept to different layers of the application independently
- *Are the states of roles the same on each layer?*
- *Are any transitions between layer-specific role states possible?*

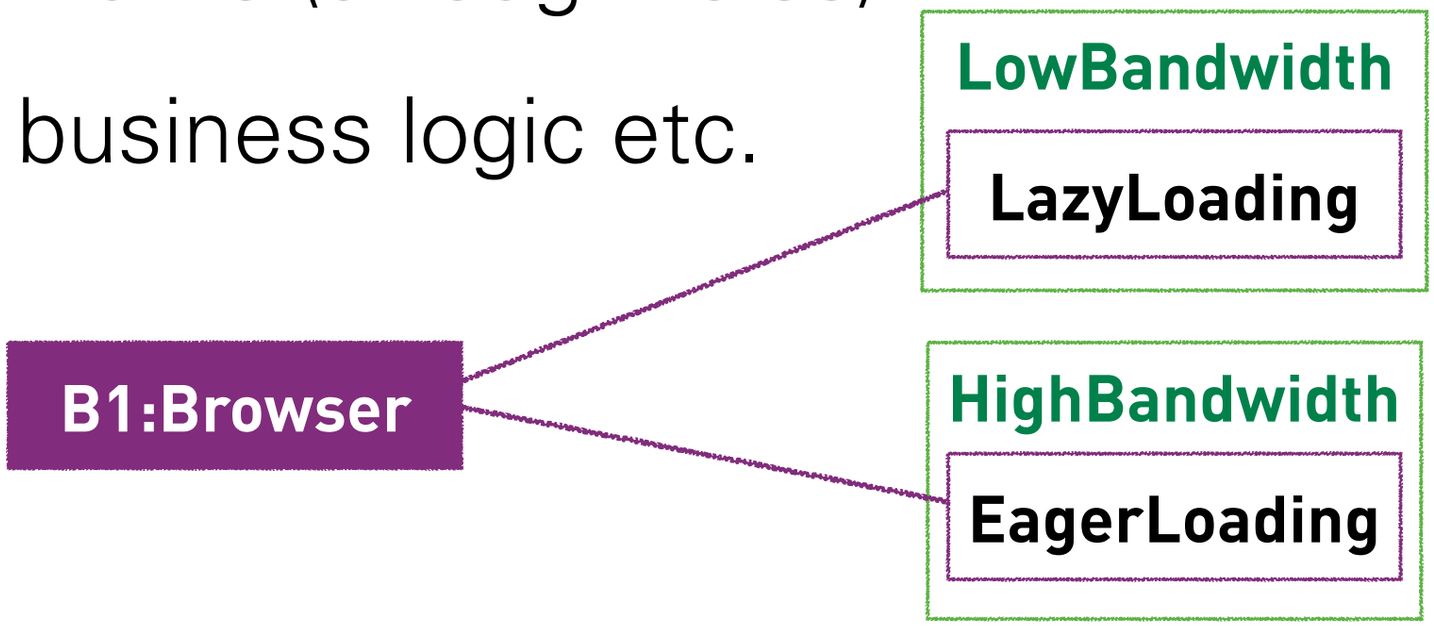


Application Layer (AL)

- Bound Active roles' behavior is 'visible' to the user
- Bound Passive roles' behavior is not perceivable

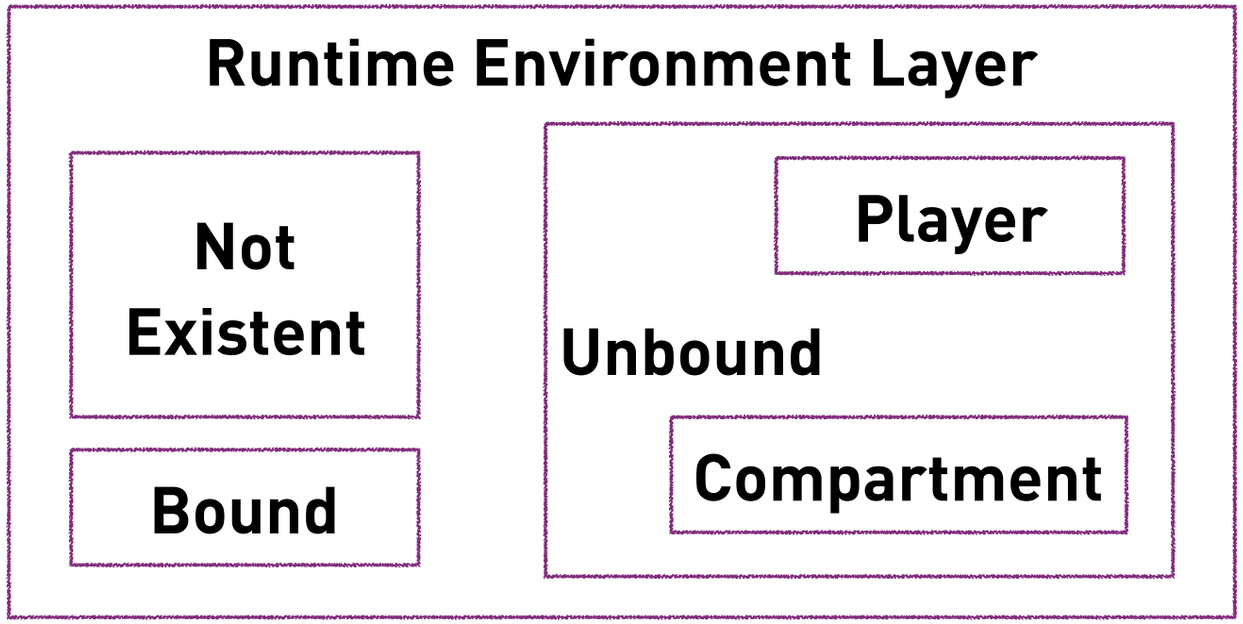


- role-base application thats behavior can be modified dynamically at run time (through roles)
- user-interaction, business logic etc.

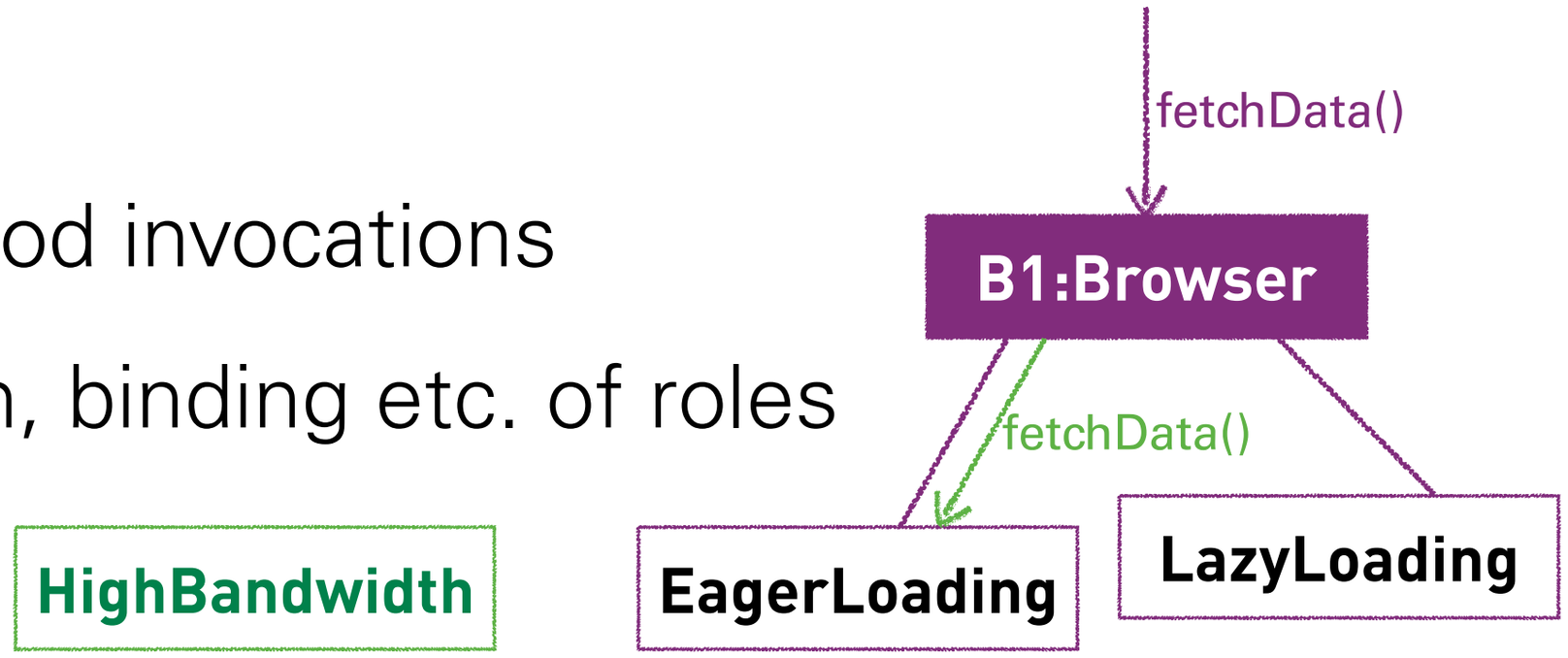


Runtime Environment Layer (REL)

- Bound roles are considered for the method dispatch
- Unbound states are transparent for the application and transient

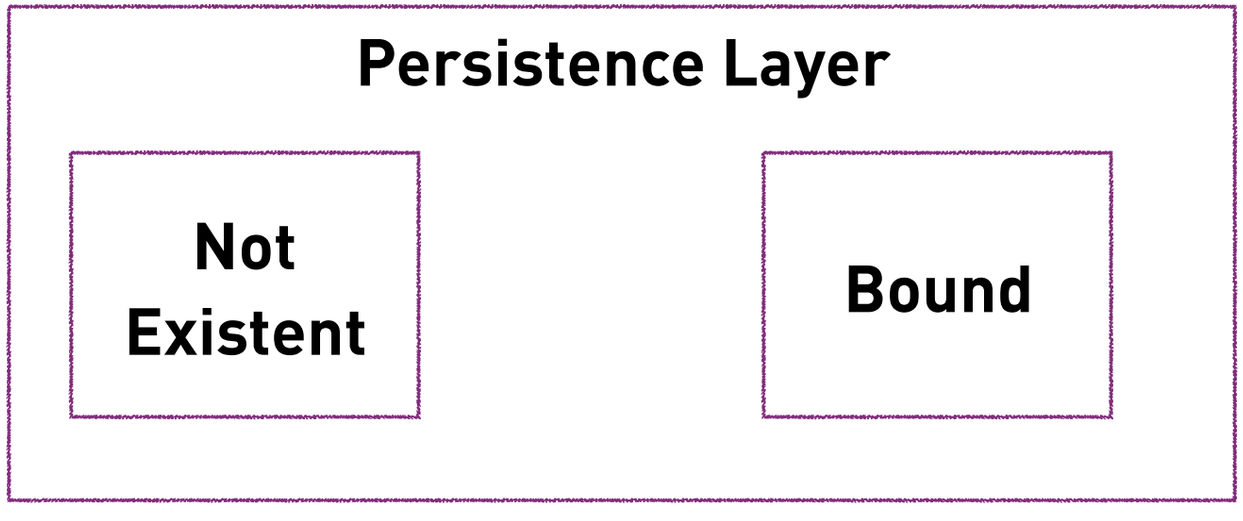


- book-keeping of currently played roles (binding information)
- dispatch of method invocations
- creation, deletion, binding etc. of roles



Persistence Layer (PL)

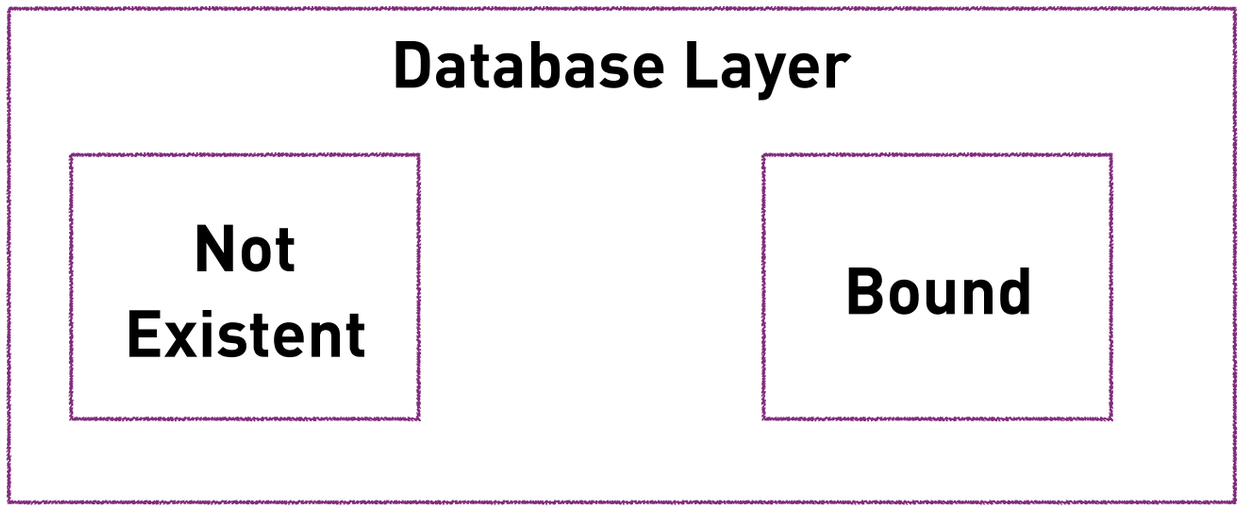
- Bound roles are stored to the database



- mapping of the programming-language specific object instances of roles and players to their respective representation on the database layer
- autonomous storage to the database on changes made to the objects

Database Layer (DBL)

- Only information about roles being bound to players are required being persistent



- stores roles, players and bind information persistently

Global Role State

- combinatorics allow invalid states → might also occur if the system is left unattended
- motivation: roles instances might exist on a single layer only, but the “entity” of the role cuts all layers in the layer stack
- limit space of possible transitions of a role, globally in the system

Create and Store a Role

C1:BrightEnvironment

P1:Browser

R1:BrightDisplay

Application Layer

bound Active

bound Passive

not existent

Runtime Environment Layer

bound

Unbound Player

Unbound Compartment

not existent

Persistence Layer

bound

not existent

Database Layer

bound

not existent

Reload a Role from the Database

The Global State Manager

C1:BrightEnvironment

P1:Browser

R1:Downloadable

Application Layer

bound Active

Runtime Environment Layer

bound

Persistence Layer

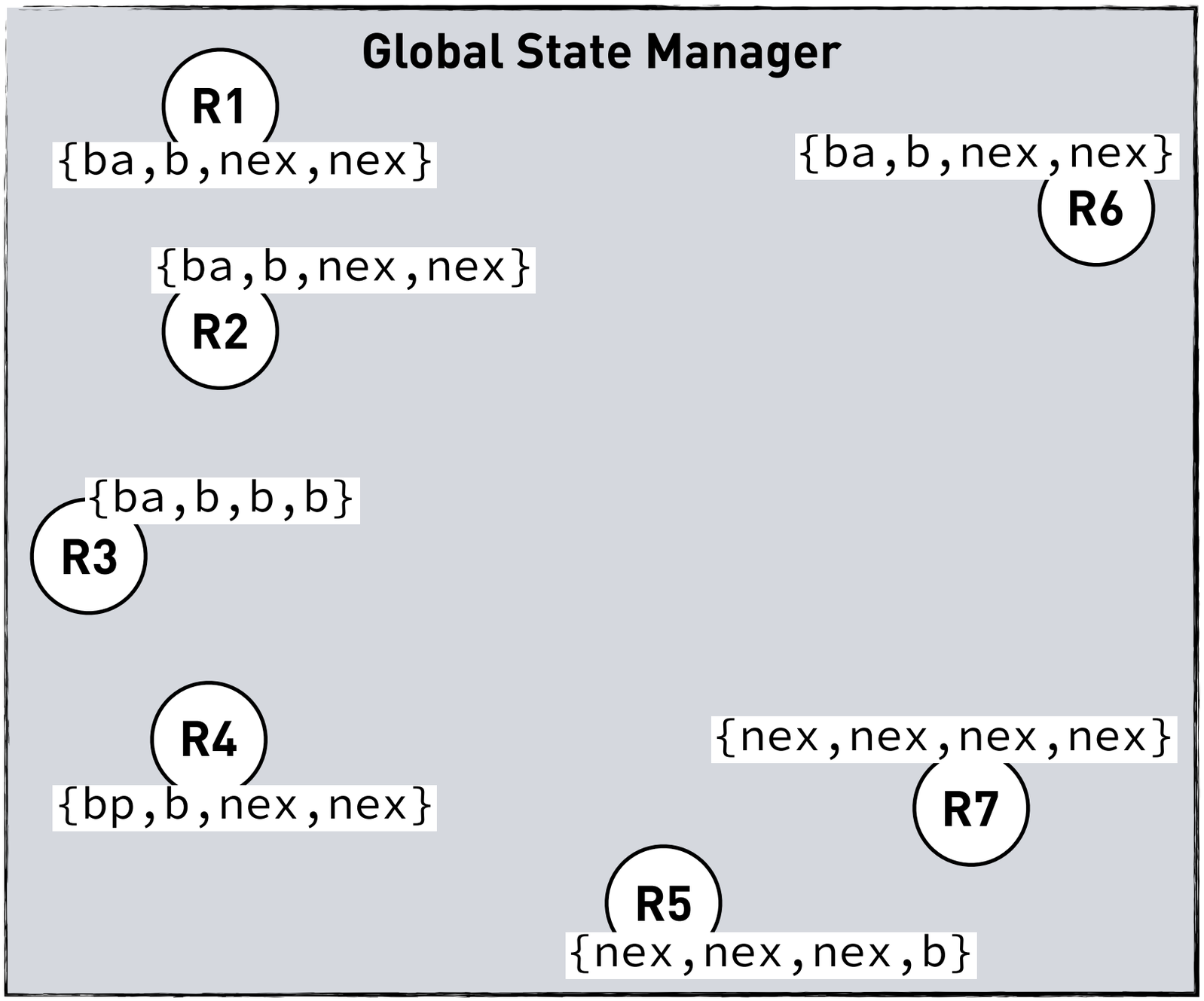
bound

Database Layer

bound

Global State Manager

The Global State Manager



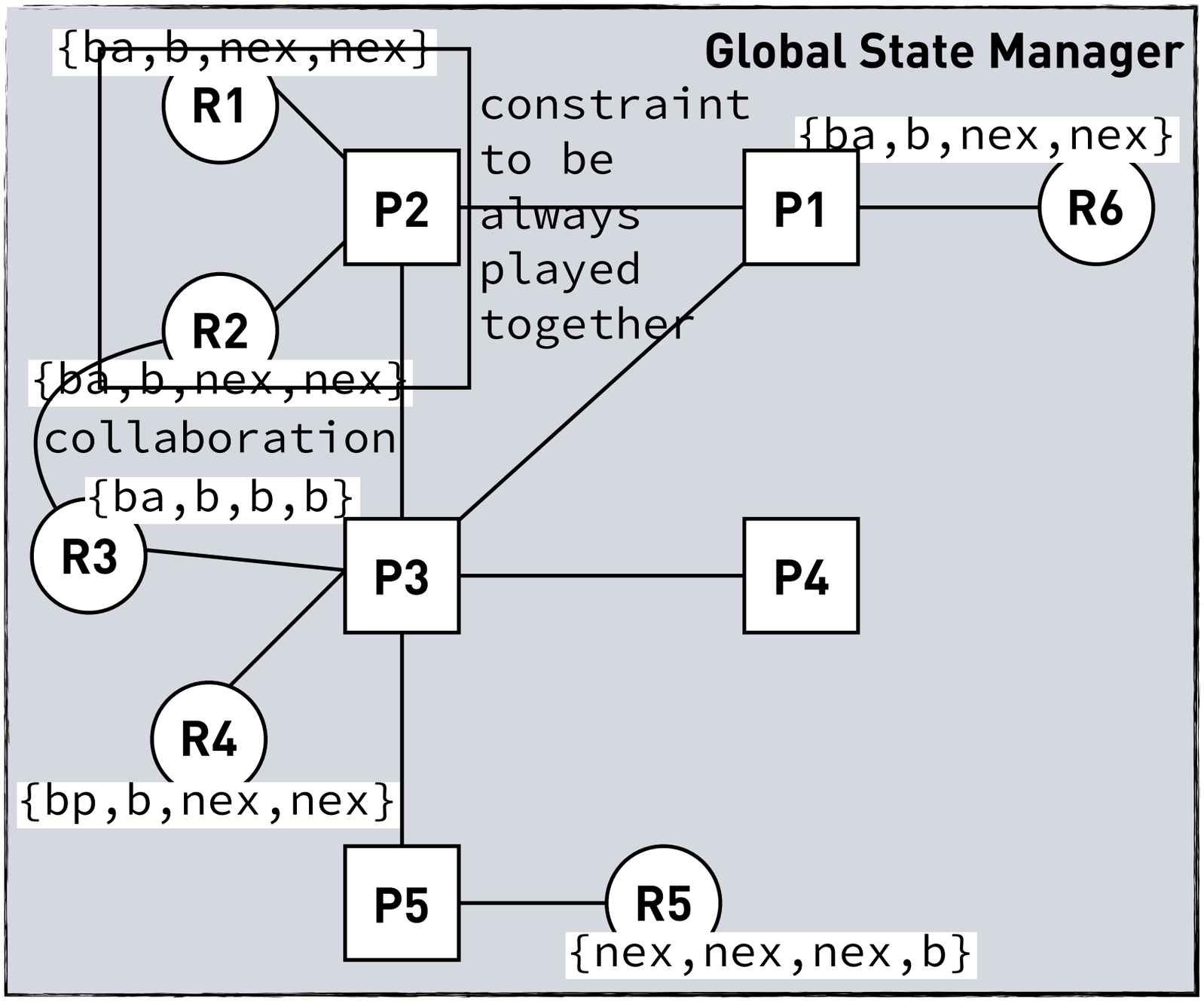
…knows…

- the roles in the system
- their current state on each layer

Benefits

- transitions of global role states can be checked at run time
- model can be used to check the system in advance for correctness
- possibly especially interesting in the case of constraints or collaborations between roles

The Global State Manager



…knows…

- the roles in the system
- their current state on each layer
- roles and their players
- the object net of players

{AL,REL,PL,DBL}

Benefits

- transitions of global role states can be checked at run time
  - *transactional changes of multiple roles can both be checked and guided*
- model can be used to check the system in advance for correctness
  - possibly especially interesting in the case of constraints or collaborations between roles

Conclusion

- role-concept on the modeling and programming language level
- exemplary investigation of the role-concept in an entire software application
- argument to incorporate the concept in a run-time model to allow for dynamic adaptation and validation

Future Work

- refinement of layers
- superset of role models that fulfil the state machine
- different quality / isolation levels
  - e.g. transactional storage of changes of role bindings / creations vs. lazy storage that allows information loss
- distributed settings & decentralization aspects

TECHNISCHE UNIVERSITÄT DRESDEN

ROSI

Department of Computer Science Institute of Systems Architecture, Computer Networks Group

Thank you for your attention!

Questions?

DFG

References

[1] Nguonly Taing, Thomas Springer, Nicolás Cardozo, Alexander Schill: **A Dynamic Instance Binding Mechanism Supporting Run-Time Variability of Role-Based Software Systems**. Modularity Companion 2016, Màlaga, Spain

[2] Nguonly Taing, Markus Wutzler, Thomas Springer, Nicolás Cardozo, Alexander Schill: **Consistent Unanticipated Adaptation for Context-Dependent Applications**. To appear at COP'16, 2016, Rome, Italy

[3] Thomas Kühn, Max Leuthäuser, Sebastian Götz, Christoph Seidl, Uwe Aßmann: **A Metamodel Family for Role-Based Modeling and Programming Languages**, In Proc. of the 7th International Conference on Software Language Engineering (SLE 2014), 2014.

[4] Tobias Jäkel, Thomas Kühn, Hannes Voigt, Wolfgang Lehner: **RSQL - A Query Language for Dynamic Data Types**. In Proc. of the 18th International Database Engineering & Applications Symposium (IDEAS '14), 2014.

[5] Thomas Kühn, Stephan Böhme, Sebastian Götz: **A Combined Formal Model for Relational Context-Dependent Roles.** In Proc. of SLE'15, ACM, 2015.

[6] Max Leuthäuser: **SCROLL - A Scala-based library for Roles at Runtime**. In Proc. of the 3rd Workshop on Domain-Specific Language Design and Implementation (DSLDI 2015)