Hierarchical Component Models
A True Story

(Ph.D. Thesis Defense)

http://d3s.mff.cuni.cz/~jezek/DoctoralThesis

Department of Distributed and Dependable Systems

D3S

Pavel Ježek

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Project: Component Reliability Extensions for Fractal Component model (CREF)

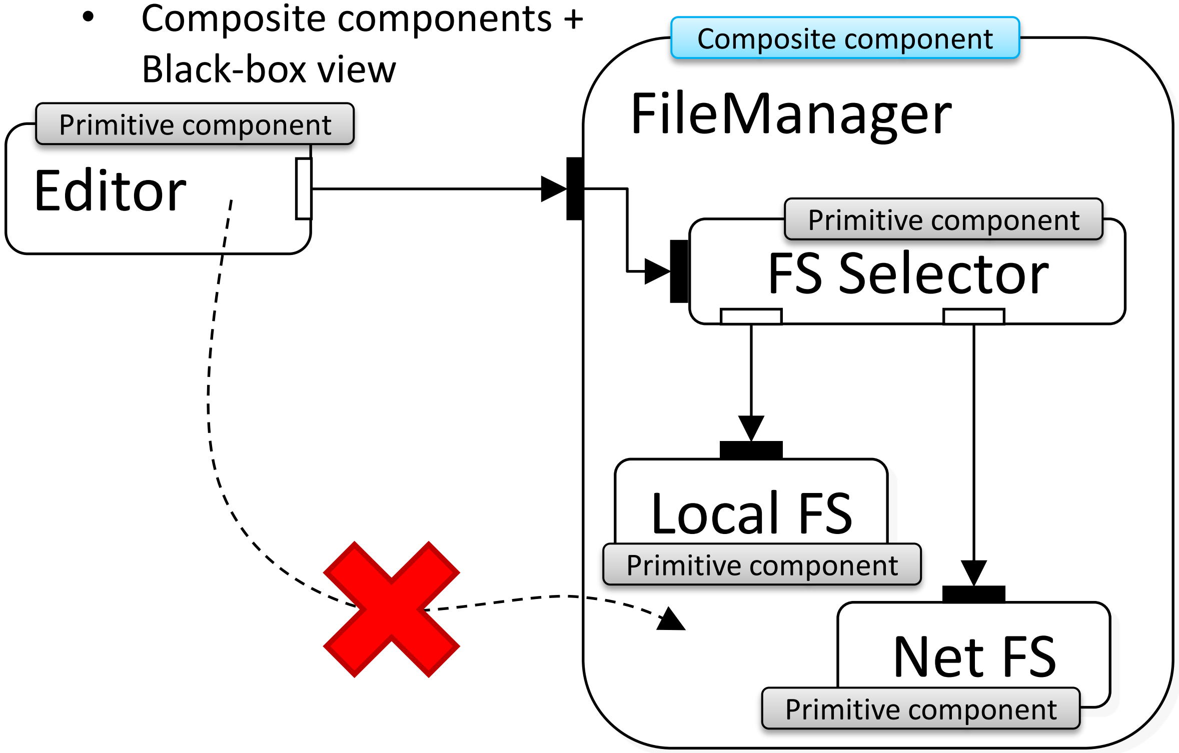
- In cooperation with Czech Academy of Sciences and France Telecom (2004-2006)
- Goal: Implement formal verification tools of “behavior protocols” in context of Fractal component model
- Why Fractal? → Hierarchical component model

Project: Component Reliability Extensions for Fractal Component model (CREF)

- In cooperation wi
  (2004-2006)
- Goal: Implement
  context of Fract
- **Why Fractal?**

Hierarchical component model (ideal):

- Provided and required interfaces
- Dataflow only through defined interfaces (no shortcuts in the architecture)
- Composite components + Black-box view

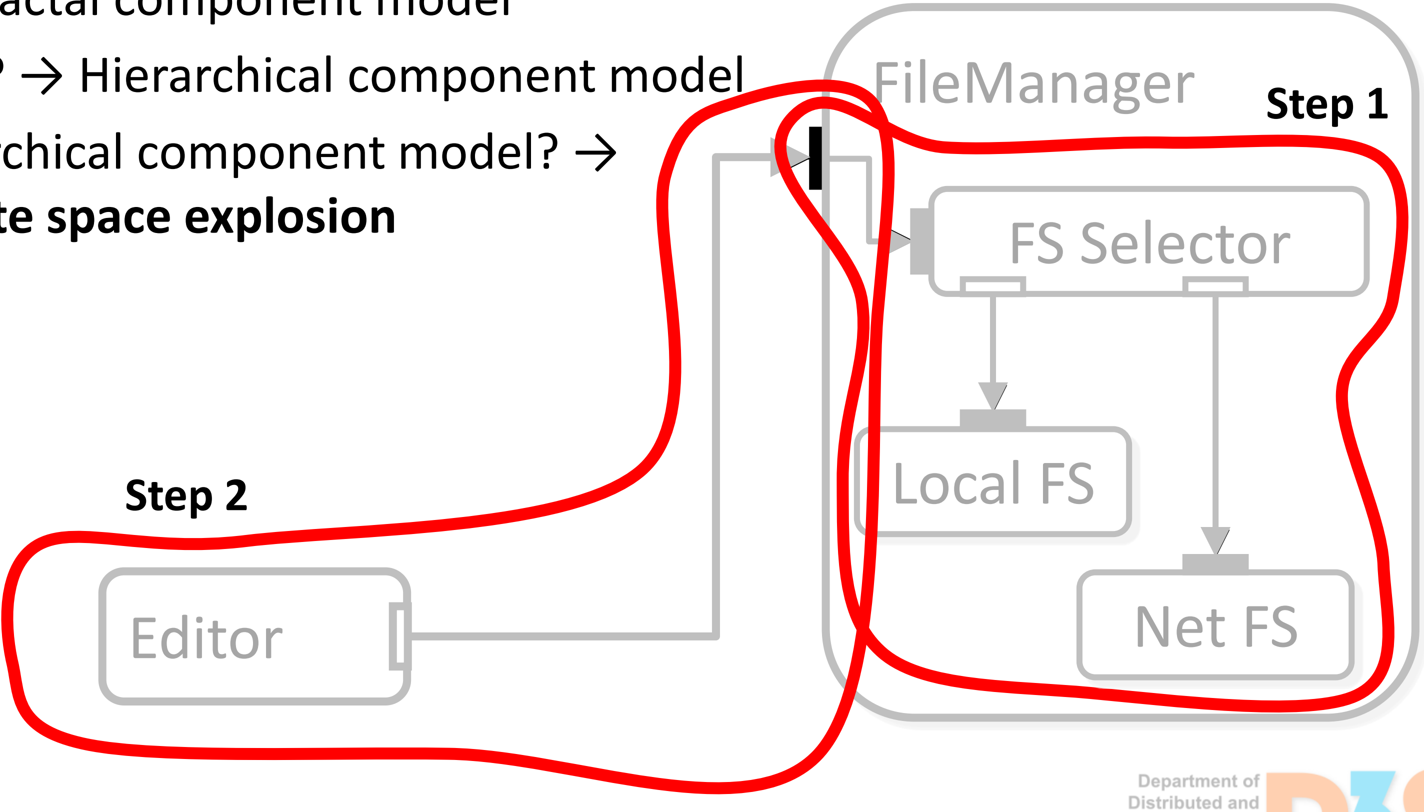


Project: Component Reliability Extensions for Fractal Component model (CREF)

All goals of the project were successfully fulfilled.

context of Fractal component model

- Why Fractal? → Hierarchical component model
- Why a hierarchical component model? → **Fighting state space explosion**

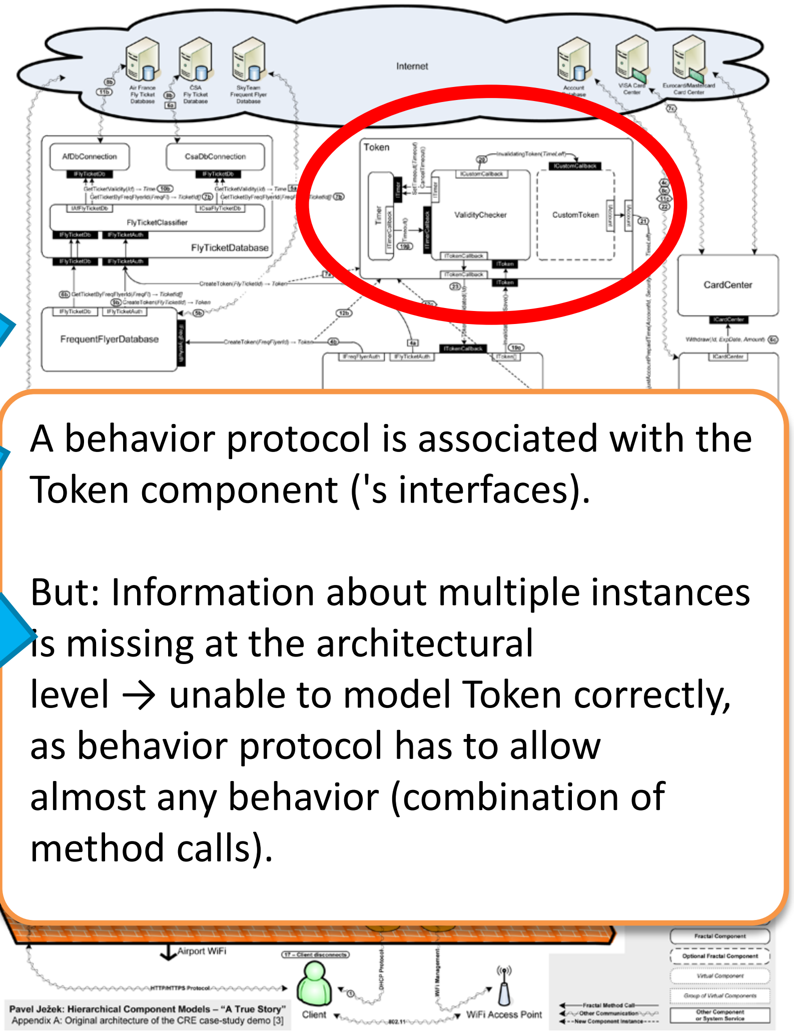


CREF Project – Issues

- Validation of BPChecker/Fractal integration in a real-life scenario required … but suitable case-study with complex hierarchical architecture was missing!
- Solution: “FT demo” case-study
- oken
- sen
- Should the Token concept be a component in the architecture or not?

Motivated by a real France Telecom project:

- System for providing internet access in airport lobbies
- Support for different types of payment and free access



A behavior protocol is associated with the Token component ('s interfaces).

But: Information about multiple instances is missing at the architectural level → unable to model Token correctly, as behavior protocol has to allow almost any behavior (combination of method calls).

Software Component – Once Again

- A common computer science term – so its understanding should be unambiguous … is it?
- JavaBeans technology is often cited in research papers as a typical component model, i.e. a JavaBeans' bean = a component (i.e. Java class is not a component)
- But: JavaBeans' features are core part of .NET platform, i.e. every .NET class ≈ a bean → Is every .NET class a component?

- Software component is … What? There are several dozens of definitions.
- Research papers often cite a definition by Clemens Szyperski → But many component models do not fit (cited as well) – detailed analysis in the thesis

Our Unique View on Components (as Runtime Entities)

- A single definition is not sufficient.
- New view: **Purpose of components in a specific component model is a defining aspect of components**→ Categorization of component models according to a newly selected criteria

Token as a component = OK?

Hierarchical component models:

- Purpose of components? Rather unclear.
- For development of typical desktop and enterprise applications too complex – more architectural freedom is beneficial + only service orientation (remote/queued components, etc.) and dependency injection required by software developers as sufficient (→ target of current industrial component models)

Goals of the Thesis

1) Identify the key goals of hierarchical component models (purpose of components) and domains where these can be most advantageous – *The True Story*.

Ideally suited for:

- Dynamic updates (replacing component internals with a new version or a different implementation)
- Application's performance modeling and prediction
- Verification of application's correctness ← CREF project

Purpose of components in HCMs is not (just) to provide a clean architecture (SE point of view), but to provide additional information about application's structure and behavior (to advanced tools).

→ **Token as a component = OK** → HCM + Behavior Protocols need to be enhanced to capture information about architectural changes.

Department of Distributed and Dependable Systems

Overview of the Main Contribution (1/2)

- **CREF project**
  - Proposal of a new case-study
  - Validation of HCM (Hierarchical Component Models) concepts

    ***Ježek P.*, Kofroň J., Plášil F.**: *Model Checking of Component Behavior Specification: A Real Life Experience*, in Electronic Notes in Theoretical Computer Science, Vol. 160, pp. 197-210, Elsevier B.V., ISSN: 1571-0661, Aug 2006 (**10 citations** in total)

    Kofroň J., Adámek J., Bureš T., *Ježek P.*, Mencl V., Parízek P., Plášil F.: *Checking Fractal Component Behavior Using Behavior Protocols*, presented at the 5th Fractal Workshop (part of ECOOP'06), July 3rd, 2006, Nantes, France, Jul 2006 (**3 citations** in total)

- **CoCoME project**
  - Validation of HCM concepts

    Bulej L., Bureš T., Coupaye T., Děcký M., *Ježek P.*, Parízek P., Plášil F., Poch T., Rivierre N., Šerý O., Tůma P.: *CoCoME in Fractal*, Chapter in The Common Component Modeling Example: Comparing Software Component Models, Springer-Verlag, LNCS 5153, Aug 2008 (**11 citations** in total)

Overview of the Main Contribution (2/2)

- **Software entities in component architectures**
  - Introduction of a novel concept of dynamic entities + validation of the concept in context of HCM & CREF project case-study

    Bureš T., *Ježek P.*, Malohlava M., Poch T., Šerý O.: *Strengthening Component Architectures by Modeling Fine-grained Entities*, in proceedings of 37th Euromicro SEAA 2011, Oulu, Finland, IEEE CS, Aug 2011
- **Modeling Windows driver environment**
  - Introduction of a novel approach to specify environment of software components

    Matoušek T., *Ježek P.*: *DeSpec: Modeling the Windows Driver Environment*, in proceedings of FESCA, ETAPS'07, Braga, Portugal, ENTCS, Mar 2007

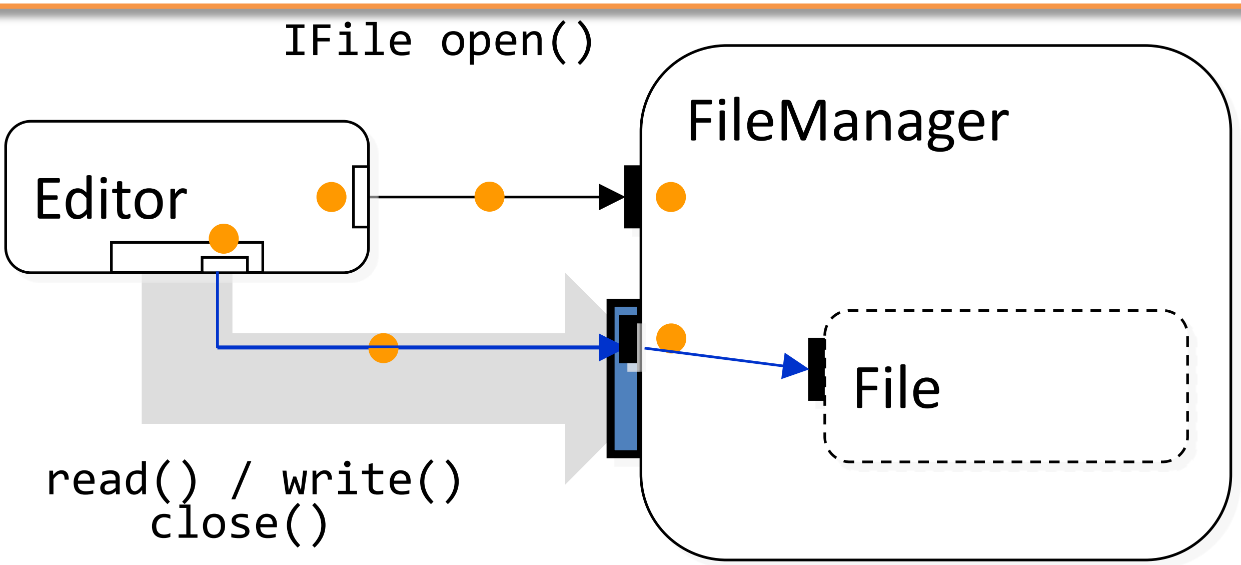
***Ježek P.*, Bureš T., Hnětynka P.**: *Supporting Real-life Applications in Hierarchical Component Systems*, in proceedings of SERA 2009, Haikou, China, Studies in Computational Intelligence (SCI), Springer, Dec 2009 (**3 citations** in total)

Department of Distributed and Dependable Systems

Entities – New CBSE Concepts

- Goals:
  - Provide ability to express dynamism in hierarchical component architectures
  - Do not break existing CBSE concepts + allow reasonable implementation in HCM runtime

How to get information about architectural changes at runtime?
How to get information about all possible architectural changes at design time?

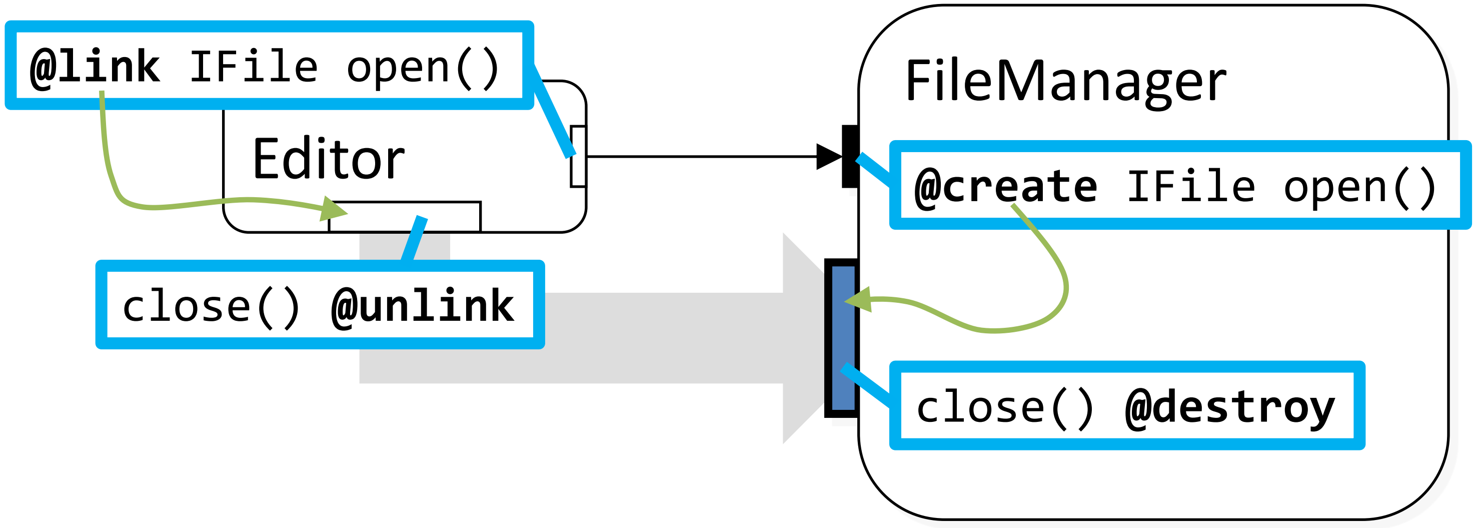


itectures by

Key feature: The black-box view still there!

Entities – Reconfiguration Actions

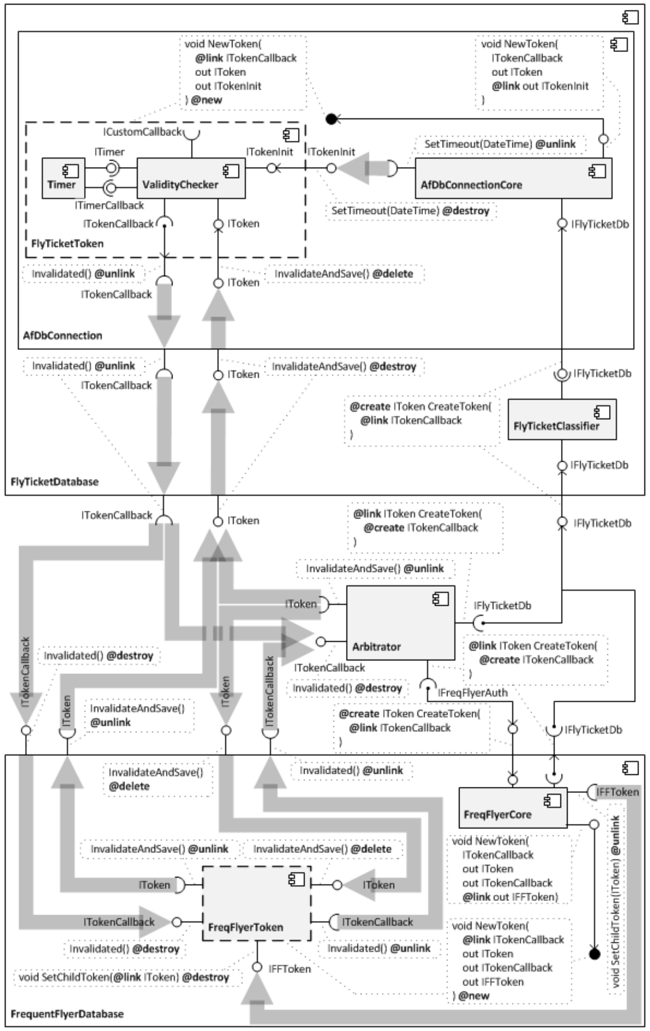
- Information about architectural changes is gathered via special annotations (reconfiguration actions – defining allowed set of changes) – triggered by method calls
- Interface publication:
  - **create**/**destroy** reconfiguration actions
- Usage of published interfaces:
  - **link**/**unlink** reconfiguration actions
- Dynamic creation of component instances:
  - **new**/**delete** reconfiguration actions



Entities – Validation (1)

- Prototype implementation in SOFA 2 (a master thesis)
- Successful remodeling of CREF case-study using entities concepts and reconfiguration actions:

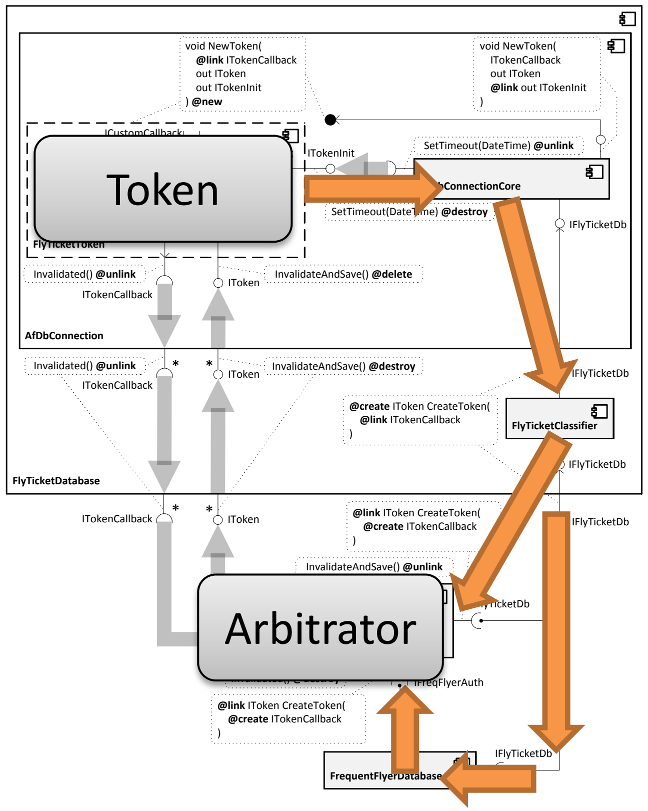
Token component + all related allowed architectural changes (new Token, passing Token reference, destroy Token) are visible in the architecture→ **Token component problem solved!**

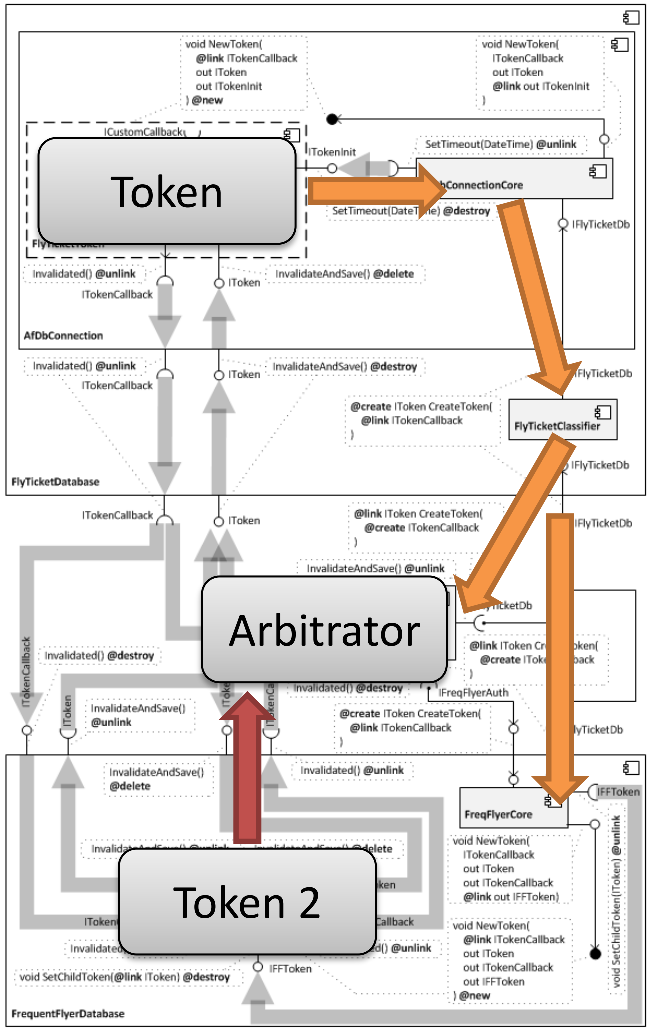


void NewToken(@link ITokenCallback out IToken out ITokenInit) @new
void NewToken(ITokenCallback out IToken @link out ITokenInit)
ICustomCallback
ITimer
ITokenInit
ITokenInit
SetTimeout(DateTime) @unlink
Timer
ValidityChecker
AfDbConnectionCore
ITimerCallback
ITokenCallback
IToken
SetTimeout(DateTime) @destroy
IFlyTicketDb
FlyTicketToken
Invalidated() @unlink
ValidateAndSave() @delete
AfDbConnection
Invalidated() @unlink
InvalidateAndSave() @destroy
IFlyTicketDb
@create IToken CreateToken(@link ITokenCallback)
FlyTicketClassifier
FlyTicketDatabase
@link IToken CreateToken(@create ITokenCallback)
InvalidateAndSave() @unlink
IFlyTicketDb
Arbitrator
Invalidated() @destroy
@link IToken CreateToken(@create ITokenCallback)
Invalidated() @destroy
IFreqFlyerAuth
InvalidateAndSave() @unlink
@create IToken CreateToken(@link ITokenCallback)
InvalidateAndSave() @delete
Invalidated() @unlink
IFFToken
FreqFlyerCore
InvalidateAndSave() @unlink
InvalidateAndSave() @delete
void NewToken(ITokenCallback out IToken out ITokenCallback @link out IFFToken)
FreqFlyerToken
void NewToken(@link ITokenCallback out IToken out ITokenCallback out IFFToken) @new
Invalidated() @destroy
Invalidated() @unlink
void SetChildToken(@link IToken) @destroy
void SetChildToken(IToken) @unlink
FrequentFlyerDatabase

Department of Distributed and Dependable Systems

Entities – Validation (2)





- Desired properties of the proto-binding and proto-interfaces concepts verified on the case-study:
  - Replacing FrequentFlyerDatabase implementation with a more complex one (providing own instances of Token components) does not require any changes of Arbitrator component interfaces or reconfiguration actions!

Conclusion & Future Work

- All goals fulfilled
- Future work:
  - Prepare a detailed analysis of component models with respect to presented categories.
  - Enhancements of entities concepts and merging the nested factory pattern.
  - Introduction of a formal method to describe and verify behavior in architectures with dynamic entities.
  - Apply DeSpec language to domain of hierarchical component models.
- Summary of publications (6) and citations (27):
  - **Bulej L., Bureš T., Coupaye T.,Děcký M., *Ježek P.*, Parízek P., Plášil F., Poch T., Rivierre N., Šerý O., Tůma P.**: *CoCoME in Fractal*, Chapter in The Common Component Modeling Example: Comparing Software Component Models, Springer-Verlag, LNCS 5153, Aug 2008 (**11 citations** in total)
  - **Bureš T., *Ježek P.*, Malohlava M., Poch T., Šerý O.**: *Strengthening Component Architectures by Modeling Fine-grained Entities*, in proceedings of 37th Euromicro SEAA 2011, Oulu, Finland, IEEE CS, Aug 2011
  - ***Ježek P.*, Bureš T., Hnětynka P.**: *Supporting Real-life Applications in Hierarchical Component Systems*, in proceedings of SERA 2009, Haikou, China, Studies in Computational Intelligence (SCI), Springer, Dec 2009 (**3 citations** in total)
  - ***Ježek P.*, Kofroň J., Plášil F.**: *Model Checking of Component Behavior Specification: A Real Life Experience*, in Electronic Notes in Theoretical Computer Science, Vol. 160, pp. 197-210, Elsevier B.V., ISSN: 1571-0661, Aug 2006 (**10 citations** in total)
  - **Kofroň J., Adámek J., Bureš T., *Ježek P.*, Mencl V., Parízek P., Plášil F.**: *Checking Fractal Component Behavior Using Behavior Protocols*, presented at the 5th Fractal Workshop (part of ECOOP'06), July 3rd, 2006, Nantes, France, Jul 2006 (**3 citations** in total)
  - **Matoušek T., *Ježek P.***: *DeSpec: Modeling the Windows Driver Environment*, in proceedings of FESCA, ETAPS'07, Braga, Portugal, ENTCS, Mar 2007

Department of Distributed and Dependable Systems

Q&A: PB: HCM Superior?

Q: ‘*The analysis of component models does not answer questions “why are hierarchical models superior”*’

A: Goal of the analysis in Chapters 2 and 3 is not to show HCM are superior, but to identify domains and problems that they are very well suited to cope with.

Q&A: PB: Component Definitions vs. Component Frameworks

Q: ‘*Secondly, the author suggests that it’s a problem of the definition that many (especially run-time) frameworks are not conformant with it. Insufficient consideration is given to a reverse view: that such systems should actually not claim to be component-based.*’

A: The problem is that not only some of these systems claim to be component-based, but common CBSE related research papers do put them in such a category as well. Thus, the goal was to clear this misunderstanding in the CBSE community.

Also a very common view is that one has to choose a “best” component model for a specific situation. But we show that multiple component models can be (and will be) used at once.

Q&A: PB: Other Architectural Reconfigurations

Q: ‘*Also, more realistic cases of architectural reconfigurations could have been considered (What if [a component for] Lufthansa Fly Ticket Database should be added at runtime? What if a new Arbitrator needs to be installed which uses additional VIPCustomerDatabase as an authorization source through a newly added IVipCustomerAuth interface?).*’

A: These are all valid questions and in fact we have considered them as well. But the problem is quite complex, and we found out that previous attempts to model reconfigurations in HCM were not sufficiently solving our problems. Thus, we targeted a clear and compact extension of HCM concepts to have a viable solution to begin with. But we plan to enhance it in our future work in similar way as proposed by the reviewer.

Q&A: PB: Component Models → Industry

Q: ‘*how do your contributions help in concrete terms make a concrete component model more relevant in industrial setting, were they (or at least how can they be) turned into practical realizations?*’

A: Enhancement of SOFA 2 hierarchical component model with dynamic entities (proto-bindings) allows us to model dynamic reconfigurations as implied by the real-life case-studies. Now the final step has to be done: prepare a formal method to take advantage of this information and to use it to verify correctness of such applications. This would make SOFA 2 interesting for software correctness verification in industrial settings.

Q&A: PB: Capturing Reconfigurations in Architecture

Q: ‘*Explain how the reconfigurations should be “properly captured in application’s architecture” and how the verification techniques proposed apply when such reconfigurations occur.*’

A: From a point of view of compositional correctness verification: All information on any possible/allowed runtime reconfigurations is needed at the time of the analysis. So, in a typical setting, all information should be present in the design-time architecture (accomplished by the reconfiguration actions of our dynamic entities concept).

Q&A: IC: Non-functional Properties

Q: '*For example non-functional properties are hardly mentioned, and these are very important and the most challenging in compositions, and in particular in hierarchical composition. For this reason it would be useful to define the hierarchy property in a more formal way and by this precisely define the scope of the research.*'

A: Yes, we definitely agree with this point. However as the domain of non-functional properties is so complex, it would be out of the scope of this thesis to provide any reasonable analysis in the domain. Thus, we focus purely on problems related to compositional correctness verification. But there are other groups in our department that are addressing problems in the domain mentioned with respect to hierarchical composition.

Department of Distributed and Dependable Systems

Q&A: IC: Addressing Desired Properties of HCM

Q: ‘*Of the desired properties of hierarchical component models that you have listed, not all elements are addressed later in the thesis. For example you identified “performance prediction” yet it seems that the term “performance” is referred only in this list, and never elaborated later. Again a precise specification of “performance” is missing.*’

A: This point is related to the previous one: We wanted to show we believe HCMs are beneficial in many domains, however as the performance related analysis of HCMs is done in another our group, provision of any detailed insight in this context was considered out of the scope of the thesis. However better references to the relevant related work should have been included in the thesis.

Department of Distributed and Dependable Systems