# Specification-Based Component Substitutability and Revision Identification

An overview of the PhD Thesis

Přemysl Brada
Charles University, Faculty of Mathematics and Physics
Thesis advisor: prof. František Plášil

23rd January, 2003

# Contents of the Talk

- Motivation and Goals
- Key results
  - » the ENT component meta-model
  - » contextual substitutability
  - » specification-based revision identification
- Conclusions
  - » lessons learned
  - » published papers

# The Scene

- Components

```
frame FAddressBook {
 provides:  IAddressBook  book;
 requires:  ::system::FileAccess  files;
 property   short  sortOrder;
 property   long  maxSize;
 protocol:  ?book.addPerson { !files.create?
   ; (!files.read + !files.write)* ... }
}
```

SOFA; CORBA CM, JavaBeans

– less widespread than hoped for
– technocracy, manual effort

# The Issues & The Goals

- Modelling and meta-modelling
  - » SOFA, CCM; MOF, UML EDOC Profile; Rastofer, Seyler et al.
  - – fragile meta-models
  - – model as design instrument vs. wiring standard
- Component evolution
  - – version identification needed but missing
  - – current schemes: ad-hoc tags or low level
    - » RCS-based, .NET; GANDALF, Larsson, Ragnarok
  - – substitutability relation for reliable replacement
  - – current foundations limiting
    - » contravariant subtyping with relaxations (Perry, Zaremski)
    - » behavioural subtyping (Liskov&Wing, Simons et al.)

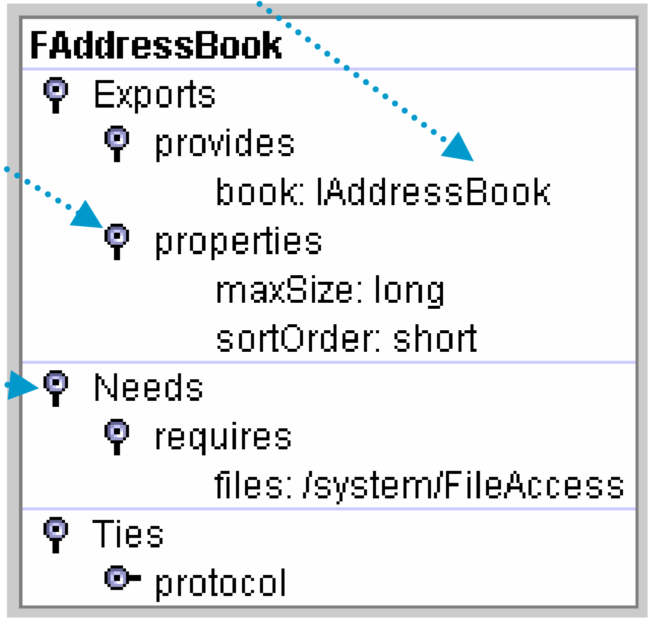
# Key Results

» component meta-model
» revision identification
» substitutability in context

# » Meta-Modelling: ENT Model

Exports - Needs - Ties

- How does a user perceive component’s interface?
  - » nature, kind, role, arity, construct, presence, lifecycle
- Element = component interface unit
  - – name and type information
  - – meta-type, classifier
- Trait = analogous elements
  - – equal classification
    - » property ∧ data, provided, multi, instance, mandatory, ...
  - – set of traits defines model/language
- Category = similar traits
  - – user-defined aggregation
    - » role == required
  - – separation of concerns (ENT, FD, ...)



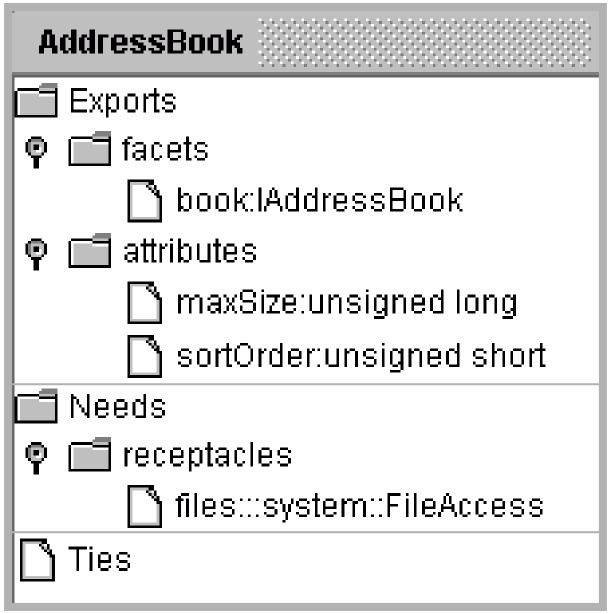
FAddressBook
- Exports
  - provides
    - book: IAddressBook
  - properties
    - maxSize: long
    - sortOrder: short
- Needs
  - requires
    - files: /system/FileAccess
- Ties
  - protocol

# Usage of the ENT Model

- Contribution:
  - interpretation of element’s meaning >>> human focus, separation of concerns, notation
  - based on analysis >>> platform independence, forward thinking (EDOC)

**CORBA Component Model (IDL3)**

```
component AddressBook {
 attribute long maxSize;
 provides IAddressBook book;
 attribute short sortOrder;
 uses
   ::system::FileAccess files;
}
```



**AddressBook**

- Exports
  - facets
    - book:IAddressBook
  - attributes
    - maxSize:unsigned long
    - sortOrder:unsigned short
- Needs
  - receptacles
    - files:::system::FileAccess
- Ties

# » Revision Identification

- What about a revision ID scheme with relevant change indication and machine-usable semantics?
  - » branching, variants out of scope
- Specification-based revision markers
  - $(r_1, \ldots, r_n)$; $r_i$ for a part of ENT structure $\xi_i$
  - $r_i^r = r_i^c+1 \Leftrightarrow diff(\xi_i^c,\xi_i^r) \neq none$ where *diff* based on $\xi_i^r <: \xi_i$
- Component revisions
  - $\xi_i$ are the E,N,T categories
- Contribution:
  - structured by the ENT model >>> semantics (relation to code)
  - *diff* >>> relevant changes
  - fit version placeholders

```
frame FAddressBook
[ @rev = 2.3.1; ]
{
  provides:
  IAdrBook#rev=2 book;
  ...
}
```

# » Substitutability

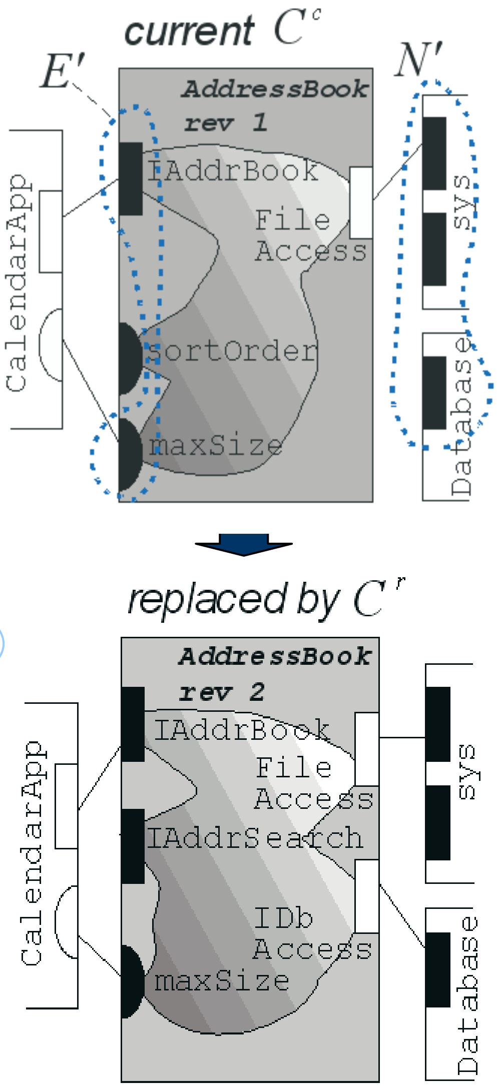
- Can we do away with contravariance?
  - covariance would be really welcome
- Components may change assumptions ...
  - requirements part of type information
  - small # of instances, infrequent changes
  - known environment, automated bindings

# Contextual Substitutability

- Environment: context
  - pseudo-component $Cx = (E',N',T')$
  - represents actual situation of $C^c$
    - » what is actually used + available
- Can substitute if $C^r <: Cx$
  - uses subtyping on ENT structures
  - defines substitutability at meta-level, contravariance via *role*

$E^r <: E'$ and $N^r :> N'$ and $T^r/A <: T'/A$

- Contribution:
  - subtyping on context >>> allow “reversed contravariance” >>> increase chances on substitution
  - couple with revision data >>> well-defined compatibility



# Conclusions

» open issues
» lessons learned

# Achievements

- Meta-model for components
  - covers wide range of current frameworks
  - prototype implementations for SOFA and CORBA CM
- Robust revision identification
  - well-defined semantics
  - prototype implementation for SOFA
- Environment-aware substitutability
  - use of context increases chances over subtyping
  - partial prototype for SOFA

# Weaknesses & Open Issues

- Meta-model
  - definition and implementation for platforms without spec.language (EJB)
- Revisions
  - do not capture the extent of differences
  - tight integration with spec.language and platform
- Substitutability
  - type-based ⇒ susceptible to simple changes, adaptation might be employed

# Lessons Learned

- Component modelling
  - language support, rich feature set needed
    - instead of naming conventions and design patterns
- Language mapping vs type rules
  - specification language type rules become part of carrier language mapping
  - watch binary compatibility (COM, Java)
- Versioning for components
  - needs integration with naming, spec languages, repository implementation, discovery methods

# Key papers: Reviewed

- P.Brada. *Towards Automated Component Compatibility Assessment*. Position paper. WCOP 2001, Budapest.
  - cited by: Jian Yang, Mike Papazoglou: *Service Components for Managing the Life-Cycle of Service Compositions*. In Information Systems 29 (2004), Elsevier 2004
- P.Brada. *Component Revision Identification based on IDL/ADL Component Specification*. Poster. Proceedings of ESEC/FSE'01, Vienna. IEEE CS Press 2001
- P.Brada. *Metadata Support for Safe Component Upgrades*. Proceedings of COMPSAC 2002, Oxford. IEEE CS Press 2002

# Not reviewed

- P.Brada. *The ENT Model: A General Model for Software Interface Structuring*. Technical report DCSE/TR-2002-10

**Thank you.**