Formální komponentový model pro mobilní architektury

Mgr. Marek Rychlý

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů

obhajoba disertační práce

Školitel: doc. Ing. Jaroslav Zendulka, CSc.

Obsah

1. Úvod
   - Současný stav
   - Motivace
   - Cíle disertace
2. Způsob realizace
   - Komponentový model
   - Modelování služeb a komponent v SOA
   - SOA a komponentový model ve vývoji softwarových systémů
3. Závěr
   - Dosažené výsledky a budoucí vývoj
   - Publikace a citace

Obsah

1. Úvod
   - Současný stav
   - Motivace
   - Cíle disertace
2. Způsob realizace
   - Komponentový model
   - Modelování služeb a komponent v SOA
   - SOA a komponentový model ve vývoji softwarových systémů
3. Závěr
   - Dosažené výsledky a budoucí vývoj
   - Publikace a citace

Komponentové systémy a jejich modely (kapitoly 3.2 a 3.3)

- **komponentový systém** je rozdělen dle funkcionality
  (způsob realizace systémů jako seskupení vzájemně spolupracujících částí)
  - **komponenty**
    (samostatně funkčně-orientované části systému s definovaným rozhraním)
  - **konektory**
    (realizují komunikaci mezi dvěma a více spolupracujícími komponentami)
  - **konfigurace**
    (konkrétní propojení komponent a konektorů, architektura systému)

Komponentový model

konkrétní metamodel komponentově-orientovaných softwarových architektur zahrnující pravidla pro chování komponent a konektorů, jejich vztahů (konfigurace), pravidla pro změny v rámci dynamické architektury (rekonfigurace), a definici jazyka pro popis takových architektur

Komponentové systémy a jejich modely (kapitoly 3.2 a 3.3)

- **komponentový systém** je rozdělen dle funkcionality
  (způsob realizace systémů jako seskupení vzájemně spolupracujících částí)
  - **komponenty**
    (samostatně funkčně-orientované části systému s definovaným rozhraním)
  - **konektory**
    (realizují komunikaci mezi dvěma a více spolupracujícími komponentami)
  - **konfigurace**
    (konkrétní propojení komponent a konektorů, architektura systému)

Komponentový model

konkrétní metamodel komponentově-orientovaných softwarových architektur zahrnující pravidla pro chování komponent a konektorů, jejich vztahů (konfigurace), pravidla pro změny v rámci dynamické architektury (rekonfigurace), a definici jazyka pro popis takových architektur

Komponentové modely a evoluce architektury (kapitola 3.1)

1. statická architektura
   (pevná struktura daná při návrhu systému)
   např. komponentové modely Wright, Darwin, jazyk ACME, diagramy v UML
2. dynamická architektura
   (struktura se vyvíjí za běhu, části systému vznikají a zanikají podle pravidel daných při návrhu systému)
   např. komponentový model Fractal, částečně SOFA
3. mobilní (plně dynamická) architektura
   (části systému se přesouvají za běhu, mění vztahy s okolím podle funkčních požadavků; volné vazby)
   částečně SOFA 2.0 („nested factory“ komponent a mobilita „utility interface“)

Komponentové modely a evoluce architektury (kapitola 3.1)

1. statická architektura
   (pevná struktura daná při návrhu systému)

   např. komponentové modely Wright, Darwin, jazyk ACME, diagramy v UML

2. dynamická architektura
   (struktura se vyvíjí za běhu, části systému vznikají a zanikají podle pravidel daných při návrhu systému)

   např. komponentový model Fractal, částečně SOFA

3. mobilní (plně dynamická) architektura
   (části systému se přesouvají za běhu, mění vztahy s okolím podle funkčních požadavků; volné vazby)

   částečně SOFA 2.0 („nested factory“ komponent a mobilita „utility interface“)

Komponentové modely a evoluce architektury (kapitola 3.1)

1. statická architektura
   (pevná struktura daná při návrhu systému)

   např. komponentové modely Wright, Darwin, jazyk ACME, diagramy v UML

2. dynamická architektura
   (struktura se vyvíjí za běhu, části systému vznikají a zanikají podle pravidel daných při návrhu systému)

   např. komponentový model Fractal, částečně SOFA

3. mobilní (plně dynamická) architektura
   (části systému se přesouvají za běhu, mění vztahy s okolím podle funkčních požadavků; volné vazby)

   částečně SOFA 2.0 („nested factory“ komponent a mobilita „utility interface“)

Nedostatky komponentových modelů (kapitola 5.1)

- Současné modely plně nepodporují mobilní architektury.
  (mobilita by měla být využita při základních operacích s komponentami, jako je spojení jejich rozhraní; neměla by to být „speciální“ vlastnost, jako je v SOFA 2.0)
- Řídící části komponent jsou odděleny od funkčních částí.
  (problémy se změnami architektury v důsledku funkčních požadavků)
- Modely nedefinují vztah k architekturám orientovaným na služby.
  (případně modelují služby jako klasické komponenty)
- Mobilní architektura není podporována použitým formalismem.
  (Wright/CSP, Darwin/Tracta, Fractal/Fractive, SOFA/BPs, atd.)
- Není definována metodika pro integraci formálních komponentových modelů do vývojového procesu SW systému.

Nedostatky komponentových modelů (kapitola 5.1)

- Současné modely plně nepodporují mobilní architektury.
  (mobilita by měla být využita při základních operacích s komponentami, jako je spojení jejich rozhraní; neměla by to být „speciální“ vlastnost, jako je v SOFA 2.0)
- Řídící části komponent jsou odděleny od funkčních částí.
  (problémy se změnami architektury v důsledku funkčních požadavků)
- Modely nedefinují vztah k architekturám orientovaným na služby.
  (případně modelují služby jako klasické komponenty)
- Mobilní architektura není podporována použitým formalismem.
  (Wright/CSP, Darwin/Tracta, Fractal/Fractive, SOFA/BPs, atd.)
- Není definována metodika pro integraci formálních komponentových modelů do vývojového procesu SW systému.

Nedostatky komponentových modelů (kapitola 5.1)

- Současné modely plně nepodporují mobilní architektury.
  (mobilita by měla být využita při základních operacích s komponentami, jako je spojení jejich rozhraní; neměla by to být „speciální“ vlastnost, jako je v SOFA 2.0)
- Řídící části komponent jsou odděleny od funkčních částí.
  (problémy se změnami architektury v důsledku funkčních požadavků)
- Modely nedefinují vztah k architekturám orientovaným na služby.
  (případně modelují služby jako klasické komponenty)
- Mobilní architektura není podporována použitým formalismem.
  (Wright/CSP, Darwin/Tracta, Fractal/Fractive, SOFA/BPs, atd.)
- Není definována metodika pro integraci formálních komponentových modelů do vývojového procesu SW systému.

Nedostatky komponentových modelů (kapitola 5.1)

- Současné modely plně nepodporují mobilní architektury.
  (mobilita by měla být využita při základních operacích s komponentami, jako je spojení jejich rozhraní; neměla by to být „speciální“ vlastnost, jako je v SOFA 2.0)
- Řídící části komponent jsou odděleny od funkčních částí.
  (problémy se změnami architektury v důsledku funkčních požadavků)
- Modely nedefinují vztah k architekturám orientovaným na služby.
  (případně modelují služby jako klasické komponenty)
- Mobilní architektura není podporována použitým formalismem.
  (Wright/CSP, Darwin/Tracta, Fractal/Fractive, SOFA/BPs, atd.)
- Není definována metodika pro integraci formálních komponentových modelů do vývojového procesu SW systému.

Nedostatky komponentových modelů (kapitola 5.1)

- Současné modely plně nepodporují mobilní architektury.
  (mobilita by měla být využita při základních operacích s komponentami, jako je spojení jejich rozhraní; neměla by to být „speciální“ vlastnost, jako je v SOFA 2.0)
- Řídící části komponent jsou odděleny od funkčních částí.
  (problémy se změnami architektury v důsledku funkčních požadavků)
- Modely nedefinují vztah k architekturám orientovaným na služby.
  (případně modelují služby jako klasické komponenty)
- Mobilní architektura není podporována použitým formalismem.
  (Wright/CSP, Darwin/Tracta, Fractal/Fractive, SOFA/BPs, atd.)
- Není definována metodika pro integraci formálních komponentových modelů do vývojového procesu SW systému.

Cíle disertace (kapitola 5.2)

1. Navrhnout a realizovat komponentový model s podporou mobilních architektur a formální sémantikou.
2. Pomocí komponentového modelu popsat služby v architektuře SOA jako komponentové systémy.
3. Poskytnout metodiku použití SOA a komponentového modelu ve vývojovém procesu softwarových systémů.
4. Ověřit navržený přístup na případové studii.

Cíle disertace (kapitola 5.2)

1. Navrhnout a realizovat komponentový model s podporou mobilních architektur a formální sémantikou.
2. Pomocí komponentového modelu popsat služby v architektuře SOA jako komponentové systémy.
3. Poskytnout metodiku použití SOA a komponentového modelu ve vývojovém procesu softwarových systémů.
4. Ověřit navržený přístup na případové studii.

Cíle disertace (kapitola 5.2)

1. Navrhnout a realizovat komponentový model s podporou mobilních architektur a formální sémantikou.
2. Pomocí komponentového modelu popsat služby v architektuře SOA jako komponentové systémy.
3. Poskytnout metodiku použití SOA a komponentového modelu ve vývojovém procesu softwarových systémů.
4. Ověřit navržený přístup na případové studii.

Cíle disertace (kapitola 5.2)

1. Navrhnout a realizovat komponentový model s podporou mobilních architektur a formální sémantikou.
2. Pomocí komponentového modelu popsat služby v architektuře SOA jako komponentové systémy.
3. Poskytnout metodiku použití SOA a komponentového modelu ve vývojovém procesu softwarových systémů.
4. Ověřit navržený přístup na případové studii.

Obsah

1. Úvod
   - Současný stav
   - Motivace
   - Cíle disertace
2. Způsob realizace
   - Komponentový model
   - Modelování služeb a komponent v SOA
   - SOA a komponentový model ve vývoji softwarových systémů
3. Závěr
   - Dosažené výsledky a budoucí vývoj
   - Publikace a citace

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) × **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
2. složené komponenty.

Tři typy rozhraní komponent:

1. funkční rozhraní,
2. řídící rozhraní,
3. odkazující rozhraní.

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) × **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
   (konečný stupeň dekompozice, jejich architekturu nepopisujeme)
2. složené komponenty.

Tři typy rozhraní komponent:

1. funkční rozhraní,
2. řídící rozhraní,
3. odkazující rozhraní.

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) × **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
2. složené komponenty.
   (jsou dále rozložitelné na systémy (pod)komponent)

Tři typy rozhraní komponent:

1. funkční rozhraní,
2. řídící rozhraní,
3. odkazující rozhraní.

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) $\times$ **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
2. složené komponenty.

Tři typy rozhraní komponent:

1. funkční rozhraní,
   (základní „business“ operace nabízené nebo požadované komponentou)
2. řídící rozhraní,
3. odkazující rozhraní.

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) × **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
2. složené komponenty.

Tři typy rozhraní komponent:

1. funkční rozhraní,
2. řídící rozhraní,
   (řídící spojení rozhraní komponent, změny jejich chování a architektury)
3. odkazující rozhraní.

Komponentový model (kapitola 6)

Dva pohledy na komponentový model:

logický pohled (metamodel) × **procesní pohled** (popis chování)

Dva druhy komponent:

1. primitivní komponenty,
2. složené komponenty.

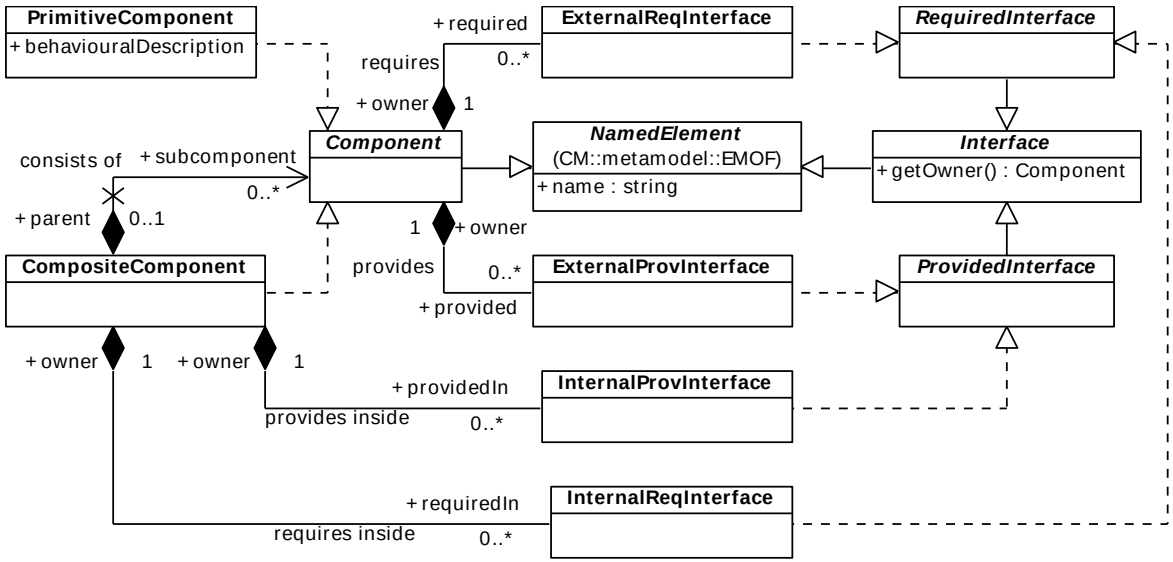
Tři typy rozhraní komponent:

1. funkční rozhraní,
2. řídící rozhraní,
3. odkazující rozhraní.
   (umožňují předávat odkazy na komponenty nebo na jejich (funkční) rozhraní)

Komponentový model – „logický“ pohled (kapitola 6.1)

Zaveden vlastní metamodel, nad kořenovými třídami z EMOF[1] definuje:

- abstraktní komponentu, primitivní a složené komponenty, vnější rozhraní komponenty a vnitřní rozhraní složené komponenty,
- požadovaná a poskytovaná rozhraní, jejich vzájemná propojení,
- typy rozhraní, poskytované operace a vyžadovaná omezení,
- konkrétní syntaxi systémových modelů.

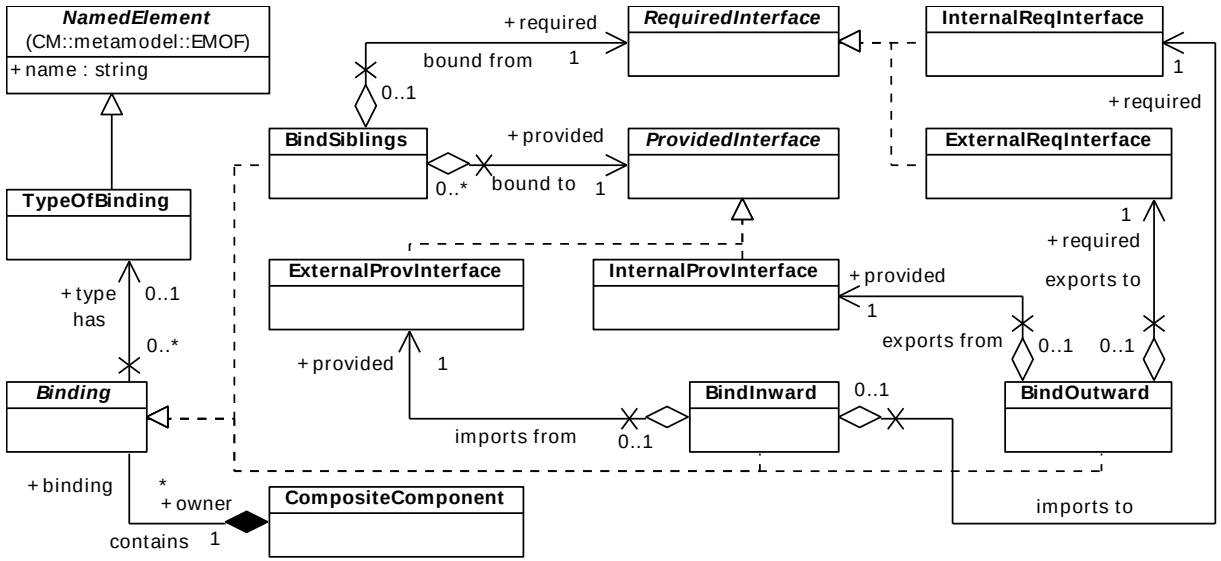


[1] třídy „NamedElement“ a „Operation“ z OMG's Essential Meta Object Facility

Komponentový model – „logický“ pohled (kapitola 6.1)

Zaveden vlastní metamodel, nad kořenovými třídami z EMOF[1] definuje:

- abstraktní komponentu, primitivní a složené komponenty, vnější rozhraní komponenty a vnitřní rozhraní složené komponenty,
- požadovaná a poskytovaná rozhraní, jejich vzájemná propojení,
- typy rozhraní, poskytované operace a vyžadovaná omezení,
- konkrétní syntaxi systémových modelů.

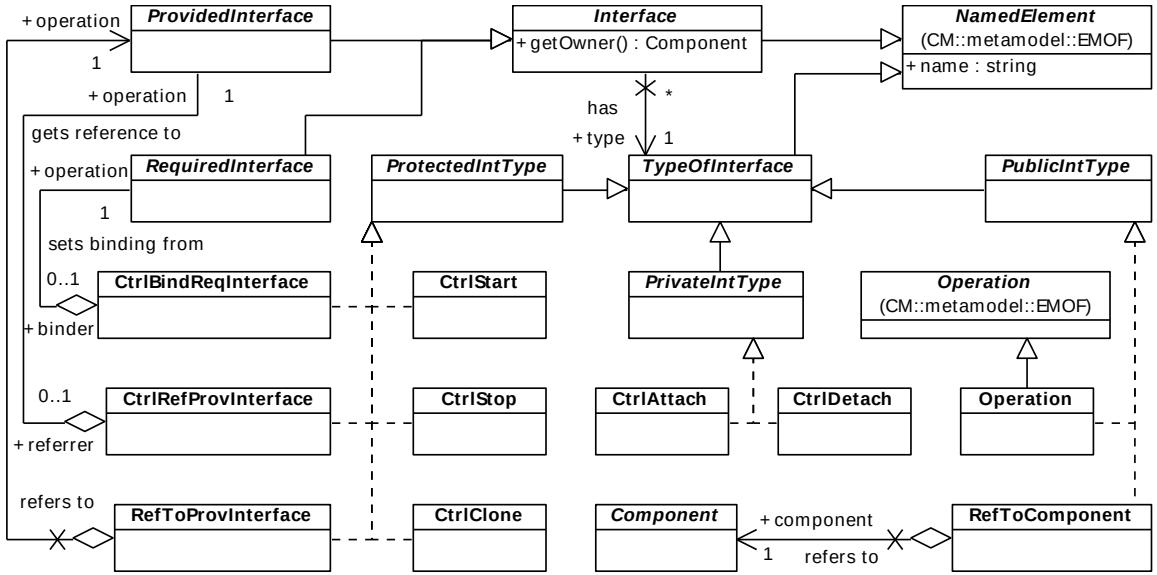


[1] třídy „NamedElement“ a „Operation“ z OMG's Essential Meta Object Facility

Komponentový model – „logický“ pohled (kapitola 6.1)

Zaveden vlastní metamodel, nad kořenovými třídami z EMOF[1] definuje:

- abstraktní komponentu, primitivní a složené komponenty, vnější rozhraní komponenty a vnitřní rozhraní složené komponenty,
- požadovaná a poskytovaná rozhraní, jejich vzájemná propojení,
- typy rozhraní, poskytované operace a vyžadovaná omezení,
- konkrétní syntaxi systémových modelů.

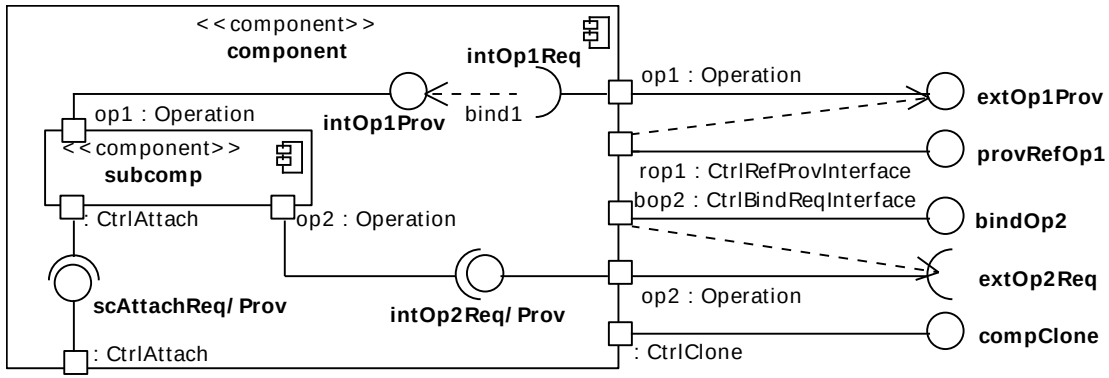


[1] třídy „NamedElement“ a „Operation“ z OMG's Essential Meta Object Facility

Komponentový model – „logický“ pohled (kapitola 6.1)

Zaveden vlastní metamodel, nad kořenovými třídami z EMOF[1] definuje:

- abstraktní komponentu, primitivní a složené komponenty, vnější rozhraní komponenty a vnitřní rozhraní složené komponenty,
- požadovaná a poskytovaná rozhraní, jejich vzájemná propojení,
- typy rozhraní, poskytované operace a vyžadovaná omezení,
- konkrétní syntaxi systémových modelů.

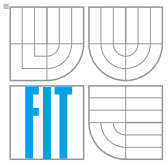


[1] třídy „NamedElement“ a „Operation“ z OMG's Essential Meta Object Facility

Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

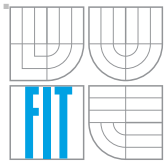
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

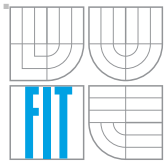
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

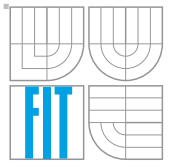
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

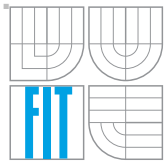
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

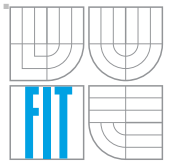
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

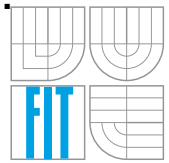
jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Komponentový model – „procesní“ pohled (kapitola 6.2)

- pro popis chování zvolen polyadický π-kalkul
  (algebra pro popis systémů souběžných a mobilních procesů; procesy a jména)
- chování **primitivních komponent** zahrnuje
  - chování jádra komponenty
    (obsluha funkčních a odkazujících rozhraní, změn životního cyklu)
  - chování obsluhy řídících rozhraní
    (vytváření odkazů poskytovaných rozhraní a jejich napojení na požadovaná rozhraní, kopírování komponent)
- chování **složených komponent** zahrnuje
  - chování jejich podkomponent a komunikaci mezi jejich rozhraními
  - chování obsluhy vnějších i vnitřních řídících rozhraní
    (navíc připojení podkomponent a distribuci změn životního cyklu)
  - komunikaci mezi vnitřními a vnějšími rozhraními

jednotlivé části chování jsou v práci přesně popsány,
až na chování jader primitivních komponent



Charakteristika zvoleného způsobu popisu chování

- popis chování respektuje hierarchickou strukturu systému
  (je možno automaticky sestavit chování složených komponent)
- systém je popsán jediným procesem (výrazem) v π-kalkulu
  (popisuje chování při konfiguraci dané systémovým modelem)
- proces se mění (redukuje) společně s evolucí architektury
  - obsluha funkčních a odkazujících rozhraní
    $\Rightarrow$ drobné a dočasné změny procesu
    (zachycují průběh zpracování funkčních požadavků)
  - obsluha řídících rozhraní
    $\Rightarrow$ výrazné a trvalé změny procesu
    (nový proces popisuje chování systému po rekonfiguraci)
- chování systémů je možno porovnávat a verifikovat
  (relace redukce a bisimulace, existují vhodné nástroje)

Charakteristika zvoleného způsobu popisu chování

- popis chování respektuje hierarchickou strukturu systému
  (je možno automaticky sestavit chování složených komponent)
- systém je popsán jediným procesem (výrazem) v π-kalkulu
  (popisuje chování při konfiguraci dané systémovým modelem)
- proces se mění (redukuje) společně s evolucí architektury
  - obsluha funkčních a odkazujících rozhraní
    $\Rightarrow$ drobné a dočasné změny procesu
    (zachycují průběh zpracování funkčních požadavků)
  - obsluha řídících rozhraní
    $\Rightarrow$ výrazné a trvalé změny procesu
    (nový proces popisuje chování systému po rekonfiguraci)
- chování systémů je možno porovnávat a verifikovat
  (relace redukce a bisimulace, existují vhodné nástroje)

Charakteristika zvoleného způsobu popisu chování

- popis chování respektuje hierarchickou strukturu systému
  (je možno automaticky sestavit chování složených komponent)
- systém je popsán jediným procesem (výrazem) v π-kalkulu
  (popisuje chování při konfiguraci dané systémovým modelem)
- proces se mění (redukuje) společně s evolucí architektury
  - obsluha funkčních a odkazujících rozhraní
    $\Rightarrow$ drobné a dočasné změny procesu
    (zachycují průběh zpracování funkčních požadavků)
  - obsluha řídících rozhraní
    $\Rightarrow$ výrazné a trvalé změny procesu
    (nový proces popisuje chování systému po rekonfiguraci)
- chování systémů je možno porovnávat a verifikovat
  (relace redukce a bisimulace, existují vhodné nástroje)

Charakteristika zvoleného způsobu popisu chování

- popis chování respektuje hierarchickou strukturu systému
  (je možno automaticky sestavit chování složených komponent)
- systém je popsán jediným procesem (výrazem) v π-kalkulu
  (popisuje chování při konfiguraci dané systémovým modelem)
- proces se mění (redukuje) společně s evolucí architektury
  - obsluha funkčních a odkazujících rozhraní
    $\Rightarrow$ drobné a dočasné změny procesu
    (zachycují průběh zpracování funkčních požadavků)
  - obsluha řídících rozhraní
    $\Rightarrow$ výrazné a trvalé změny procesu
    (nový proces popisuje chování systému po rekonfiguraci)
- chování systémů je možno porovnávat a verifikovat
  (relace redukce a bisimulace, existují vhodné nástroje)

Charakteristika zvoleného způsobu popisu chování

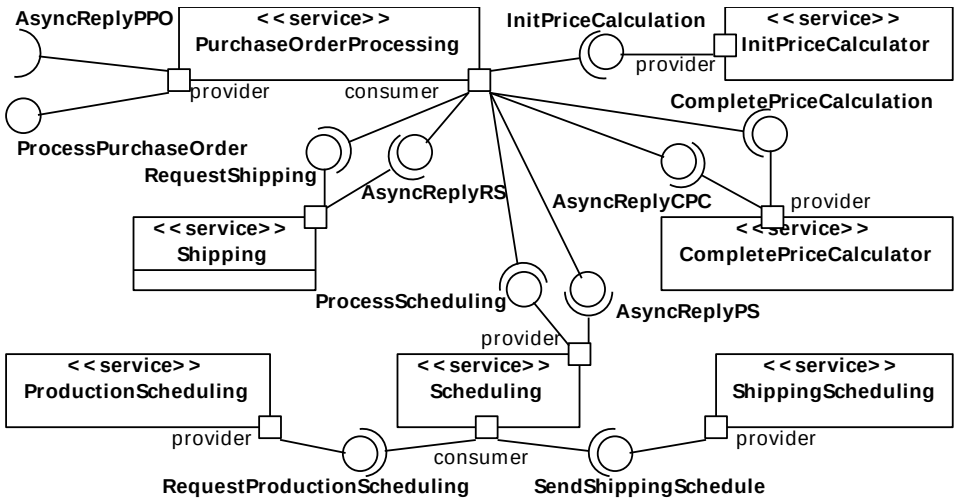
- popis chování respektuje hierarchickou strukturu systému
  (je možno automaticky sestavit chování složených komponent)
- systém je popsán jediným procesem (výrazem) v π-kalkulu
  (popisuje chování při konfiguraci dané systémovým modelem)
- proces se mění (redukuje) společně s evolucí architektury
  - obsluha funkčních a odkazujících rozhraní
    $\Rightarrow$ drobné a dočasné změny procesu
    (zachycují průběh zpracování funkčních požadavků)
  - obsluha řídících rozhraní
    $\Rightarrow$ výrazné a trvalé změny procesu
    (nový proces popisuje chování systému po rekonfiguraci)
- chování systémů je možno porovnávat a verifikovat
  (relace redukce a bisimulace, existují vhodné nástroje)

Modelování architektury SOA (kapitola 7.3)

- SOA představuje aplikaci komponentového modelu
- služby jsou na nižší úrovni realizovány jako komponentové systémy
- na vyšší úrovni zavádí disertace pro modelování SOA diagramy UML
  - komponentové diagramy – propojení služeb a role jejich rozhraní
  - diagramy tříd – poskytovaná a požadovaná rozhraní, jejich operace
  - sekvenční diagramy – popis volání služeb, orchestrace

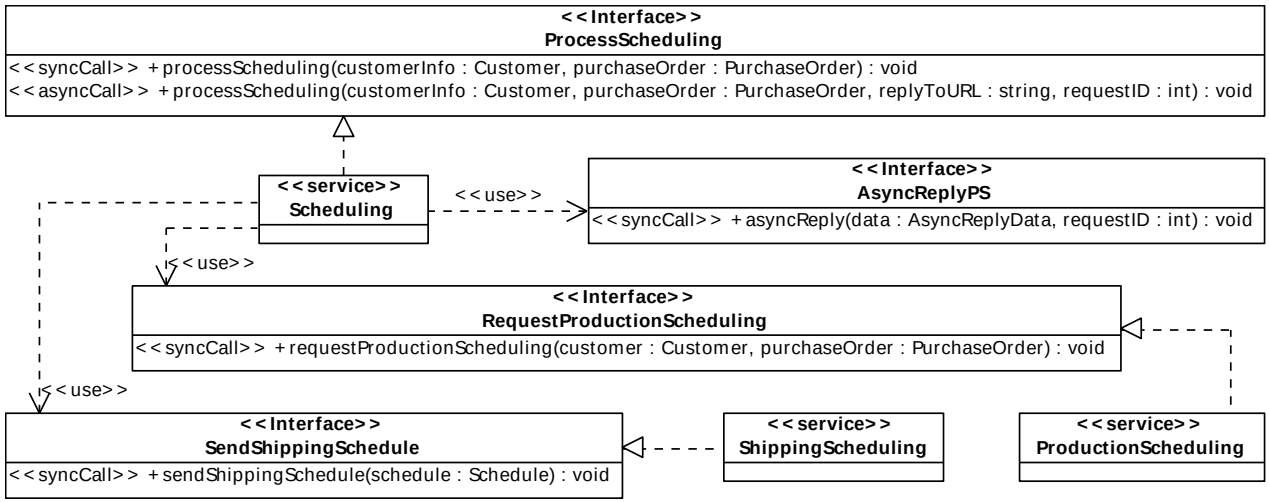
Modelování architektury SOA (kapitola 7.3)

- SOA představuje aplikaci komponentového modelu
- služby jsou na nižší úrovni realizovány jako komponentové systémy
- na vyšší úrovni zavádí disertace pro modelování SOA diagramy UML
  - komponentové diagramy – propojení služeb a role jejich rozhraní
  - diagramy tříd – poskytovaná a požadovaná rozhraní, jejich operace
  - sekvenční diagramy – popis volání služeb, orchestrace



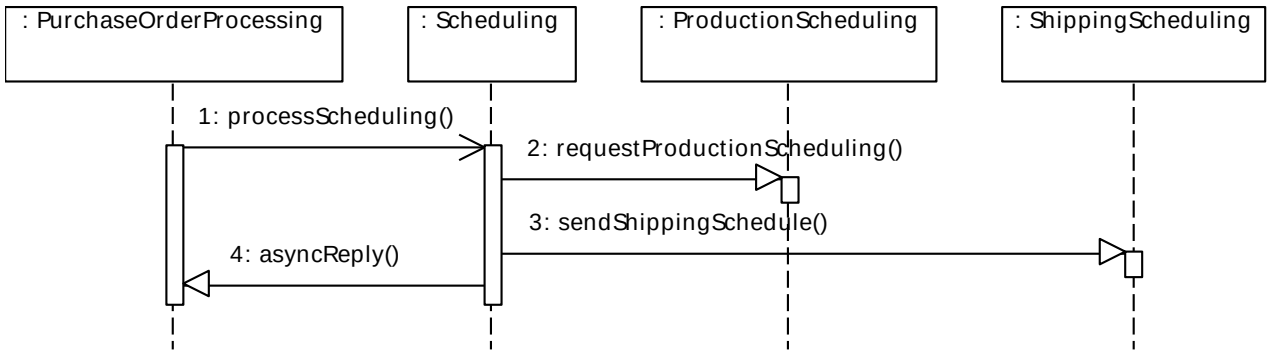
Modelování architektury SOA (kapitola 7.3)

- SOA představuje aplikaci komponentového modelu
- služby jsou na nižší úrovni realizovány jako komponentové systémy
- na vyšší úrovni zavádí disertace pro modelování SOA diagramy UML
  - komponentové diagramy – propojení služeb a role jejich rozhraní
  - diagramy tříd – poskytovaná a požadovaná rozhraní, jejich operace
  - sekvenční diagramy – popis volání služeb, orchestrace



Modelování architektury SOA (kapitola 7.3)

- SOA představuje aplikaci komponentového modelu
- služby jsou na nižší úrovni realizovány jako komponentové systémy
- na vyšší úrovni zavádí disertace pro modelování SOA diagramy UML
  - komponentové diagramy – propojení služeb a role jejich rozhraní
  - diagramy tříd – poskytovaná a požadovaná rozhraní, jejich operace
  - sekvenční diagramy – popis volání služeb, orchestrace



Popis chování služeb v SOA (kapitoly 7.1 a 7.2)

Definovány dva pohledy na popis chování služby

1. jako entity architektury SOA,
   (chování ve vztahu k ostatním službám architektury)
2. jako komponentového systému, který ji realizuje.
   (chování systému komponent, které společně danou službu implementují)

- popis chování služby jako entity v SOA pomocí π-kalkulu
  (návaznost chování služby na chováním příslušného komponentového systému)
- celý systém služeb je popsán jediným procesem v π-kalkulu
  (chování architektury SOA vč. v ní obsažených komponentových systémů)

Popis chování služeb v SOA (kapitoly 7.1 a 7.2)

Definovány dva pohledy na popis chování služby

1. jako entity architektury SOA,
   (chování ve vztahu k ostatním službám architektury)
2. jako komponentového systému, který ji realizuje.
   (chování systému komponent, které společně danou službu implementují)

- popis chování služby jako entity v SOA pomocí π-kalkulu
  (návaznost chování služby na chováním příslušného komponentového systému)
- celý systém služeb je popsán jediným procesem v π-kalkulu
  (chování architektury SOA vč. v ní obsažených komponentových systémů)

Popis chování služeb v SOA (kapitoly 7.1 a 7.2)

Definovány dva pohledy na popis chování služby

1. jako entity architektury SOA,
   (chování ve vztahu k ostatním službám architektury)
2. jako komponentového systému, který ji realizuje.
   (chování systému komponent, které společně danou službu implementují)

- popis chování služby jako entity v SOA pomocí π-kalkulu
  (návaznost chování služby na chováním příslušného komponentového systému)
- celý systém služeb je popsán jediným procesem v π-kalkulu
  (chování architektury SOA vč. v ní obsažených komponentových systémů)

Od business procesů ke službám (kapitola 8.1)

1. krok vývojového procesu

1. analýza business procesů, identifikace kandidátů na služby
   (stanovení cílů služeb a jejich sdružení do „candidate services“)
2. modelování architektury SOA, služby a jejich rozhraní a interakce
   (UML diagramy tříd služeb a rozhraní s jejich operacemi, diagramy interakce)
3. výběr služeb modelovaných dále jako komponentové systémy
   (u ostatních jen popis chování na úrovni SOA, dle kapitoly 7)

Výsledkem jsou následující výstupy:

- UML diagramy s definicí služeb,
- popis chování služeb na úrovni SOA,
- definice rozhraní komponentových systémů pro vybrané služby.

Od business procesů ke službám (kapitola 8.1)

1. krok vývojového procesu

1. analýza business procesů, identifikace kandidátů na služby
   (stanovení cílů služeb a jejich sdružení do „candidate services“)
2. modelování architektury SOA, služby a jejich rozhraní a interakce
   (UML diagramy tříd služeb a rozhraní s jejich operacemi, diagramy interakce)
3. výběr služeb modelovaných dále jako komponentové systémy
   (u ostatních jen popis chování na úrovni SOA, dle kapitoly 7)

Výsledkem jsou následující výstupy:

- UML diagramy s definicí služeb,
- popis chování služeb na úrovni SOA,
- definice rozhraní komponentových systémů pro vybrané služby.

Od business procesů ke službám (kapitola 8.1)

1. krok vývojového procesu

1. analýza business procesů, identifikace kandidátů na služby
   (stanovení cílů služeb a jejich sdružení do „candidate services“)
2. modelování architektury SOA, služby a jejich rozhraní a interakce
   (UML diagramy tříd služeb a rozhraní s jejich operacemi, diagramy interakce)
3. výběr služeb modelovaných dále jako komponentové systémy
   (u ostatních jen popis chování na úrovni SOA, dle kapitoly 7)

Výsledkem jsou následující výstupy:

- UML diagramy s definicí služeb,
- popis chování služeb na úrovni SOA,
- definice rozhraní komponentových systémů pro vybrané služby.

Od business procesů ke službám (kapitola 8.1)

1. krok vývojového procesu

1. analýza business procesů, identifikace kandidátů na služby
   (stanovení cílů služeb a jejich sdružení do „candidate services“)
2. modelování architektury SOA, služby a jejich rozhraní a interakce
   (UML diagramy tříd služeb a rozhraní s jejich operacemi, diagramy interakce)
3. výběr služeb modelovaných dále jako komponentové systémy
   (u ostatních jen popis chování na úrovni SOA, dle kapitoly 7)

Výsledkem jsou následující výstupy:

- UML diagramy s definicí služeb,
- popis chování služeb na úrovni SOA,
- definice rozhraní komponentových systémů pro vybrané služby.

Komponentové systémy jako realizace služeb (kapitola 8.2)

2. krok vývojového procesu

Každá z vybraných služeb je popsána jako komponentový systém:

1. model architektury systému,
   (systém je složená komponenta, která je dále hierarchicky dekomponována)
2. popis chování primitivních komponent jejich návrhářem,
   (tyto komponenty nelze dále rozložit, tvoří listy stromu hierarchie komponent)
3. sestavení popisu chování složených komponent.
   (z chování podkomponent a jejich vzájemné komunikace; možno automaticky)

Pro každou službu (komponentový systém) jsou výsledkem:

- systémový model,
  (logický pohled dle komponentového modelu)
- výraz v π-kalkulu popisující celkové chování.
  (procesní pohled dle komponentového modelu)

Komponentové systémy jako realizace služeb (kapitola 8.2)

2. krok vývojového procesu

Každá z vybraných služeb je popsána jako komponentový systém:

1. model architektury systému,
   (systém je složená komponenta, která je dále hierarchicky dekomponována)
2. popis chování primitivních komponent jejich návrhářem,
   (tyto komponenty nelze dále rozložit, tvoří listy stromu hierarchie komponent)
3. sestavení popisu chování složených komponent.
   (z chování podkomponent a jejich vzájemné komunikace; možno automaticky)

Pro každou službu (komponentový systém) jsou výsledkem:

- systémový model,
  (logický pohled dle komponentového modelu)
- výraz v π-kalkulu popisující celkové chování.
  (procesní pohled dle komponentového modelu)

Komponentové systémy jako realizace služeb (kapitola 8.2)

2. krok vývojového procesu

Každá z vybraných služeb je popsána jako komponentový systém:

1. model architektury systému,
   (systém je složená komponenta, která je dále hierarchicky dekomponována)
2. popis chování primitivních komponent jejich návrhářem,
   (tyto komponenty nelze dále rozložit, tvoří listy stromu hierarchie komponent)
3. sestavení popisu chování složených komponent.
   (z chování podkomponent a jejich vzájemné komunikace; možno automaticky)

Pro každou službu (komponentový systém) jsou výsledkem:

- systémový model,
  (logický pohled dle komponentového modelu)
- výraz v π-kalkulu popisující celkové chování.
  (procesní pohled dle komponentového modelu)

Komponentové systémy jako realizace služeb (kapitola 8.2)

2. krok vývojového procesu

Každá z vybraných služeb je popsána jako komponentový systém:

1. model architektury systému,
   (systém je složená komponenta, která je dále hierarchicky dekomponována)
2. popis chování primitivních komponent jejich návrhářem,
   (tyto komponenty nelze dále rozložit, tvoří listy stromu hierarchie komponent)
3. sestavení popisu chování složených komponent.
   (z chování podkomponent a jejich vzájemné komunikace; možno automaticky)

Pro každou službu (komponentový systém) jsou výsledkem:

- systémový model,
  (logický pohled dle komponentového modelu)
- výraz v π-kalkulu popisující celkové chování.
  (procesní pohled dle komponentového modelu)

Uplatnění formálního popisu chování (kapitola 8.3)

3. krok vývojového procesu

Máme k dispozici:

- modely architektury systému,
  („ploché“ modely služeb i modely hierarchický komponentových systémů)
- popis chování systému.
  (komunikaci služeb a komponent a vývoj architektury)

Výsledky můžeme využít pro:

- ověření chování služeb a komponent,
  (schopnost reagovat na požadavky, pracovat v určitých kontextech)
- porovnání chování služeb a komponent,
  (ekvivalentní chování, jsou-li procesy popisující jejich chování v relaci bisimulace)
- ověření vývoje architektury.
  (dosažitelnost (ne)vhodných rekonfigurací architektury, mobilita komponent)

Uplatnění formálního popisu chování (kapitola 8.3)

3. krok vývojového procesu

Máme k dispozici:

- modely architektury systému,
  („ploché“ modely služeb i modely hierarchický komponentových systémů)
- popis chování systému.
  (komunikaci služeb a komponent a vývoj architektury)

Výsledky můžeme využít pro:

- ověření chování služeb a komponent,
  (schopnost reagovat na požadavky, pracovat v určitých kontextech)
- porovnání chování služeb a komponent,
  (ekvivalentní chování, jsou-li procesy popisující jejich chování v relaci bisimulace)
- ověření vývoje architektury.
  (dosažitelnost (ne)vhodných rekonfigurací architektury, mobilita komponent)

Uplatnění formálního popisu chování (kapitola 8.3)

3. krok vývojového procesu

Máme k dispozici:

- modely architektury systému,
  („ploché“ modely služeb i modely hierarchický komponentových systémů)
- popis chování systému.
  (komunikaci služeb a komponent a vývoj architektury)

Výsledky můžeme využít pro:

- ověření chování služeb a komponent,
  (schopnost reagovat na požadavky, pracovat v určitých kontextech)
- porovnání chování služeb a komponent,
  (ekvivalentní chování, jsou-li procesy popisující jejich chování v relaci bisimulace)
- ověření vývoje architektury.
  (dosažitelnost (ne)vhodných rekonfigurací architektury, mobilita komponent)

Obsah

1 Úvod
- Současný stav
- Motivace
- Cíle disertace

2 Způsob realizace
- Komponentový model
- Modelování služeb a komponent v SOA
- SOA a komponentový model ve vývoji softwarových systémů

3 Závěr
- Dosažené výsledky a budoucí vývoj
- Publikace a citace

Dosažené výsledky a budoucí vývoj

- Navržen komponentový model, který plně podporuje mobilní architektury a jejich formální popis.
  (modely SOA a CBS, popis chování služeb a komponent, vývoj architektury)
- Model umožňuje popsat „komponentu-agenta“.
  (dynamická rekonfigurace a mobilita za účelem poskytnutí funkce/služby)
- Model pokrývá proces návrhu SOA od služeb až po komponenty.
  (chování celé architektury je popsatelné jedním procesem π-kalkulu)
- Model ověřen na případové studii.
  (kapitola 10, „a testing environment of a railway interlocking control system“)

Budoucí práce:

- vytvořit UML 2 profil pro konkrétní syntaxi modelu,
- vytvořit nebo přizpůsobit implementační rámec,
- rozšířit modelovací nástroj o práci s popisem chování.
  (v rámci práce byl vytvořen prototyp nástroje pro komponentové modelování)

Dosažené výsledky a budoucí vývoj

- Navržen komponentový model, který plně podporuje mobilní architektury a jejich formální popis.
  (modely SOA a CBS, popis chování služeb a komponent, vývoj architektury)
- Model umožňuje popsat „komponentu-agenta“.
  (dynamická rekonfigurace a mobilita za účelem poskytnutí funkce/služby)
- Model pokrývá proces návrhu SOA od služeb až po komponenty.
  (chování celé architektury je popsatelné jedním procesem π-kalkulu)
- Model ověřen na případové studii.
  (kapitola 10, „a testing environment of a railway interlocking control system“)

Budoucí práce:

- vytvořit UML 2 profil pro konkrétní syntaxi modelu,
- vytvořit nebo přizpůsobit implementační rámec,
- rozšířit modelovací nástroj o práci s popisem chování.
  (v rámci práce byl vytvořen prototyp nástroje pro komponentové modelování)

Dosažené výsledky a budoucí vývoj

- Navržen komponentový model, který plně podporuje mobilní architektury a jejich formální popis.
  (modely SOA a CBS, popis chování služeb a komponent, vývoj architektury)
- Model umožňuje popsat „komponentu-agenta“.
  (dynamická rekonfigurace a mobilita za účelem poskytnutí funkce/služby)
- Model pokrývá proces návrhu SOA od služeb až po komponenty.
  (chování celé architektury je popsatelné jedním procesem π-kalkulu)
- Model ověřen na případové studii.
  (kapitola 10, „a testing environment of a railway interlocking control system“)

Budoucí práce:

- vytvořit UML 2 profil pro konkrétní syntaxi modelu,
- vytvořit nebo přizpůsobit implementační rámec,
- rozšířit modelovací nástroj o práci s popisem chování.
  (v rámci práce byl vytvořen prototyp nástroje pro komponentové modelování)

Dosažené výsledky a budoucí vývoj

- Navržen komponentový model, který plně podporuje mobilní architektury a jejich formální popis.
  (modely SOA a CBS, popis chování služeb a komponent, vývoj architektury)
- Model umožňuje popsat „komponentu-agenta“.
  (dynamická rekonfigurace a mobilita za účelem poskytnutí funkce/služby)
- Model pokrývá proces návrhu SOA od služeb až po komponenty.
  (chování celé architektury je popsatelné jedním procesem π-kalkulu)
- Model ověřen na případové studii.
  (kapitola 10, „a testing environment of a railway interlocking control system“)

Budoucí práce:

- vytvořit UML 2 profil pro konkrétní syntaxi modelu,
- vytvořit nebo přizpůsobit implementační rámec,
- rozšířit modelovací nástroj o práci s popisem chování.
  (v rámci práce byl vytvořen prototyp nástroje pro komponentové modelování)

Dosažené výsledky a budoucí vývoj

- Navržen komponentový model, který plně podporuje mobilní architektury a jejich formální popis.
  (modely SOA a CBS, popis chování služeb a komponent, vývoj architektury)
- Model umožňuje popsat „komponentu-agenta“.
  (dynamická rekonfigurace a mobilita za účelem poskytnutí funkce/služby)
- Model pokrývá proces návrhu SOA od služeb až po komponenty.
  (chování celé architektury je popsatelné jedním procesem π-kalkulu)
- Model ověřen na případové studii.
  (kapitola 10, „a testing environment of a railway interlocking control system“)

Budoucí práce:

- vytvořit UML 2 profil pro konkrétní syntaxi modelu,
- vytvořit nebo přizpůsobit implementační rámec,
- rozšířit modelovací nástroj o práci s popisem chování.
  (v rámci práce byl vytvořen prototyp nástroje pro komponentové modelování)

Publikace a citace

- 6 plných příspěvků na mezinárodních konferencích
  (ENASE, CEE-SET, ISIM, MEMICS, vše recenzováno)
- 1 článek v mezinárodním časopise
  (eInformatica)
- 1 příspěvek v LNCS k obecnému vývoji informačních systémů
  (název „A Tool for Supporting Feature-Driven Development“)

v době odevzdání práce známé 4 citace uvedených publikací
(v IEEE a LNCS sbornících)

Děkuji za pozornost.