Модуларизација измена путем примене аспектно-оријентисаног развоја софтвера

Валентино Вранић

Slovak University of Technology in Bratislava, Slovakia

Пословни факултет у Ваљеву, Универзитет Сингидунум

vranic@fiit.stuba.sk
http://fiit.stuba.sk/~vranic/

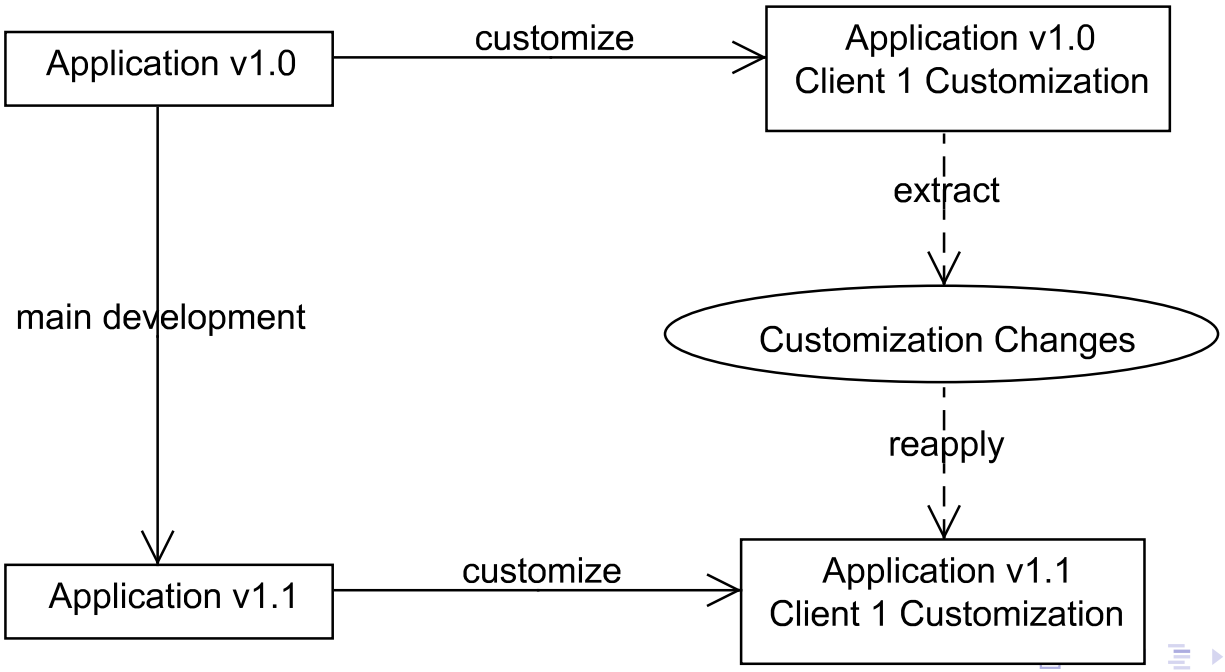
Мрежа 2011
Примена Веб технологија у раду пословних система
Ваљево
17. јун 2011.

Зашто користити аспекте у реализацији измена?

- Реализација измена је сложена и скупа
- Измене софтверских апликација имају пресецајућу природу
  - Често се односе на више различитих делова апликације
  - Бивају схваћене као самосталне јединице у смислу потребе за њиховим укључењем или искључењем из дате верзије
- Аспектно-оријентисано програмирање омогућава модуларну имплементацију измена

Ситуација из праксе

- Прилагођавање (customization) веб апликација
- Клијент прилагођава купљену апликацију својим потребама
- Апликација се развија даље и клијент добија нову верзију
- Нова верзија апликације захтева реапликацију измена за прилагођавање на страни клијента



Аспектно-оријентисани приступ

- У програмима се испољавају интереси које није могуће модуларизовати: *пресецајући интереси* (crosscutting concerns)
- Последица је преплетен и разбацан код
- Нпр. методе апликационе логике често садрже и код који се тиче других интереса, као што је безбедност, перзистенција, аутентификација...
- Имплементација ових интереса је разбацана у много метода уз понављање кода

Аспектно-оријентисано програмирање

- Посебне конструкције – *аспекти* – утичу на објектно-оријентисани или процедурални код (али могу да утичу и на друге аспекте)
- Типичан пример: обухватање позива метода које одговарају одређеној сигнатури и извршење одређеног кода пре, после или уместо ових позива

AspectJ

- Најзаступљенији аспектно-оријентисани језик је AspectJ[1]
- Заснован је на језику Java и има подршку у радном окружењу Eclipse
- Стабилан је и користи се у пракси – нпр. обезбеђује изолацију функција различитих издања у IBM WAS[2]
- Низ других језика опонаша AspectJ: AspectC++, CeasarJ, AspectS...
- Подршка и на нивоу софтверских оквира: AspectWerkz, Spring, JBoss, Seasar...

[1] http://eclipse.org/aspectj/

[2] http://gateway.comp.lancs.ac.uk:8080/c/portal/layout?p_l_id=1.94

Измене као пресецајући интереси

- Измену иницира захтев за измену (change request)
  - Исказан је у појмовима домена (апликације)
  - Има тенденцију да буде фокусиран, али обично претпоставља више измена
- Апстраховањем и уопштавањем суштине измене могуће је идентификовати њен тип
- Овакав тип измена је применљив на низ апликација у тој истој области

Пример: увођење резервног SMTP сервера (1)

- Претпоставимо да је у апликацији потребно увести резервни SMTP сервер као замену за првобитни SMTP сервер у случају да овај не функционише
- Сам код првобитног SMTP сервера није у нашим рукама и не можемо га мењати
- Без коришћења аспектно-оријентисаног програмирања би било потребно увести условну употребу резервног SMTP сервера на свим местима где се користи првобитни SMTP сервер
- SMTP сервери су програмски представљени као објекти
- Помоћу аспектно-оријентисаног програмирања можемо да заменимо објекат првобитног SMTP сервера свугде где је то потребно без било каквих модификација кода

Пример: увођење резервног SMTP сервера (2)

```
class NewSMTPServer extends SMTPServer {
    . . .
}

public aspect BackupSMTPServer {
    public pointcut SMTPServerConstructor(URL url, String user, String password):
        call(SMTPServer.new(..)) && args (url, user, password);

    SMTPServer around(URL url, String user, String password):
        SMTPServerConstructor(url, user, password) {
        return getSMTPServerBackup(proceed(url, user, password));
    }

    SMTPServer getSMTPServerBackup(SMTPServer obj) {
        if (obj.isConnected()) {
            return obj;
        }
        else {
            return new SMTPServerBackup(obj.getUrl(), obj.getUser(),
                obj.getPassword());
        }
    }
}
```

Како доћи до одговарајуће АО имплементације?

- Из описа измене не мора да буде јасан начин њене реализације
- Могуће је идентификовати спецификациони тип измене који одговара датој измени
  - Тип измене из примера: Introduce Resource Backup
  - Ови типови измена су доменски специфични
- За сваки спецификациони тип измена могуће је одредити одговарајући тип реализације измена:
  - Утврђено је да Class Exchange одговара типу Introduce Resource Backup
  - Ови типови измена су опште применљиви

Модел АО реализације измена са два нивоа

- Развијен је приступ за аспектно-оријентисану реализацију измена[3]
- Заснован је на разликовању два нивоа типова реализације измена:
  - Спецификациони типови измена – идентификација типа реализације измене у самом захтеву за измену
    - нпр. увођење резервног ресурса
  - Реализациони типови измена – схеме програмског кода
    - нпр. замена класе

[3] V. Vranić, R. Menkyna, M. Bebjak, and P. Dolog. Aspect-Oriented Change Realizations and Their Interaction. e-Informatica Software Engineering Journal, 3(1):43-58, 2009.

Catalog of Changes in Web Application Domain (1)

- Integration Changes
  - One Way Integration: Performing Action After Event
  - Two Way Integration: Performing Action After Event
- Grid Display Changes
  - Adding Column to Grid: Performing Action After Event
  - Removing Column from Grid: Method Substitution
  - Altering Column Presentation in Grid: Method Substitution

Catalog of Changes in Web Application Domain (2)

- Input Form Changes
  - Adding Fields to Form: Enumeration Modification with Additional Return Value Checking/Modification
  - Removing Fields from Form: Additional Return Value Checking/Modification
  - Introducing Additional Constraint on Fields: Additional Parameter Checking or Performing Action After Event
- Introducing User Rights Management: Border Control with Method Substitution
- User Interface Restriction: Additional Return Value Checking/Modifications
- Introducing Resource Backup: Class Exchange

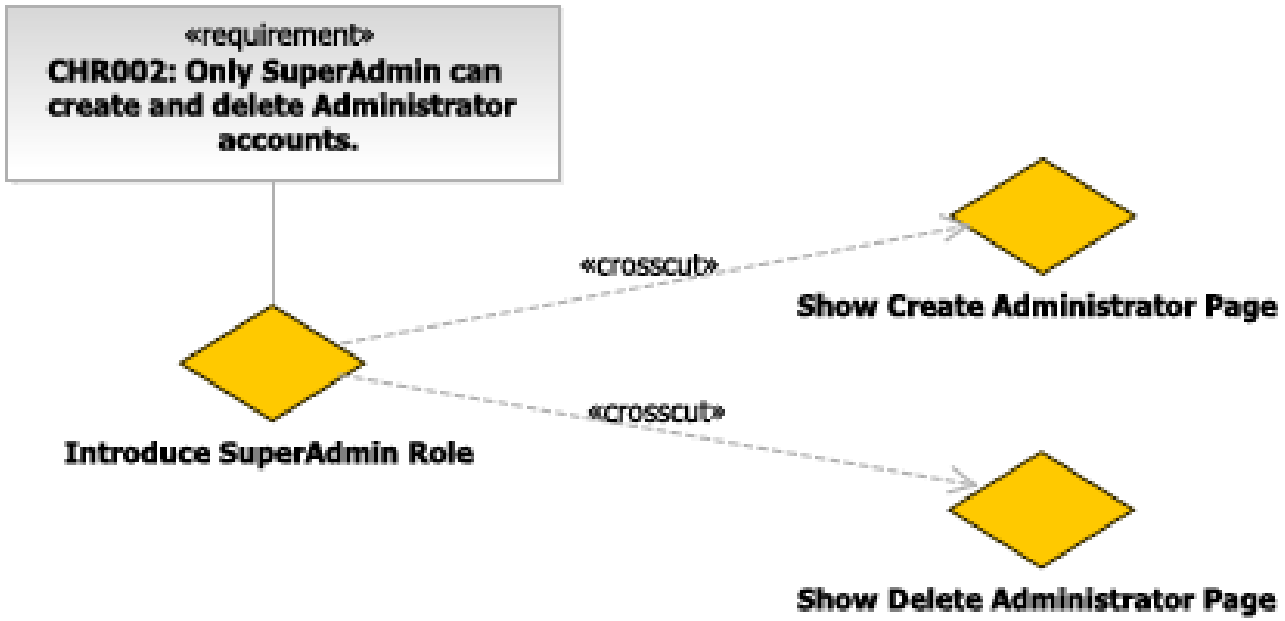
Моделирање измена

- Измене може бити једноставније планирати на нивоу модела
- Модел АО реализација измена на два нивоа је подржан и на нивоу моделирања
- Коришћен је приступ Theme заснован на језику UML
- Приступ ће бити приказан кроз пример измена у моделу webmail апликације[4]

[4] B. Kuliha. Realizing Changes by Aspects at the Design Level. Master's thesis, Slovak University of Technology in Bratislava, 2010.

Иницијално исказивање измене

- Претпоставимо да је у webmail апликацији потребно увести нову улогу: суперадминистратора
- Само суперадминистратор ће моћи да ствара и брише налоге администратора
- Обични администратори више неће имати ово право
- Ситуацију можемо да прикажемо у нотацији Theme/Doc



Идентификација спецификационог типа измене

- Следећи корак је идентификација спецификационог типа измене

Идентификација реализационог типа измене

- Спецификациони тип измене нас путем каталога води ка реализационом типу
- У овом случају постоје две могућности

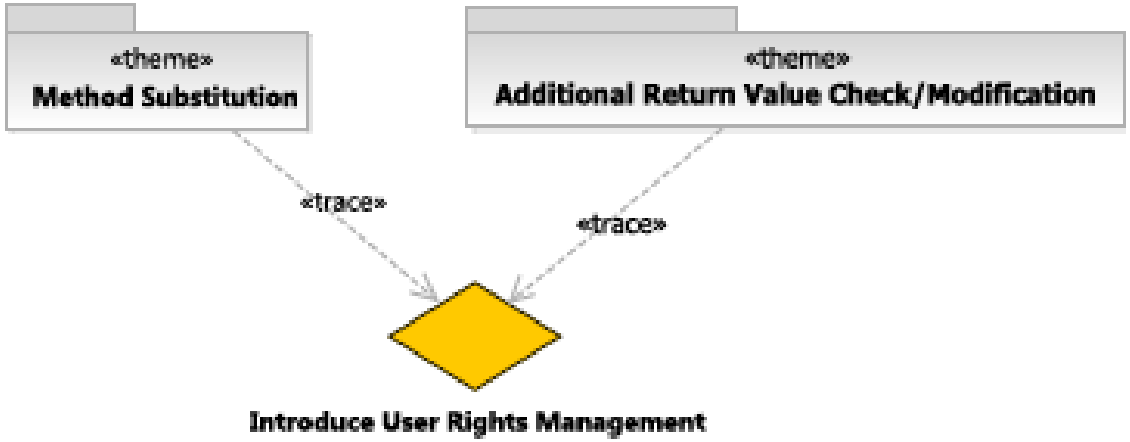
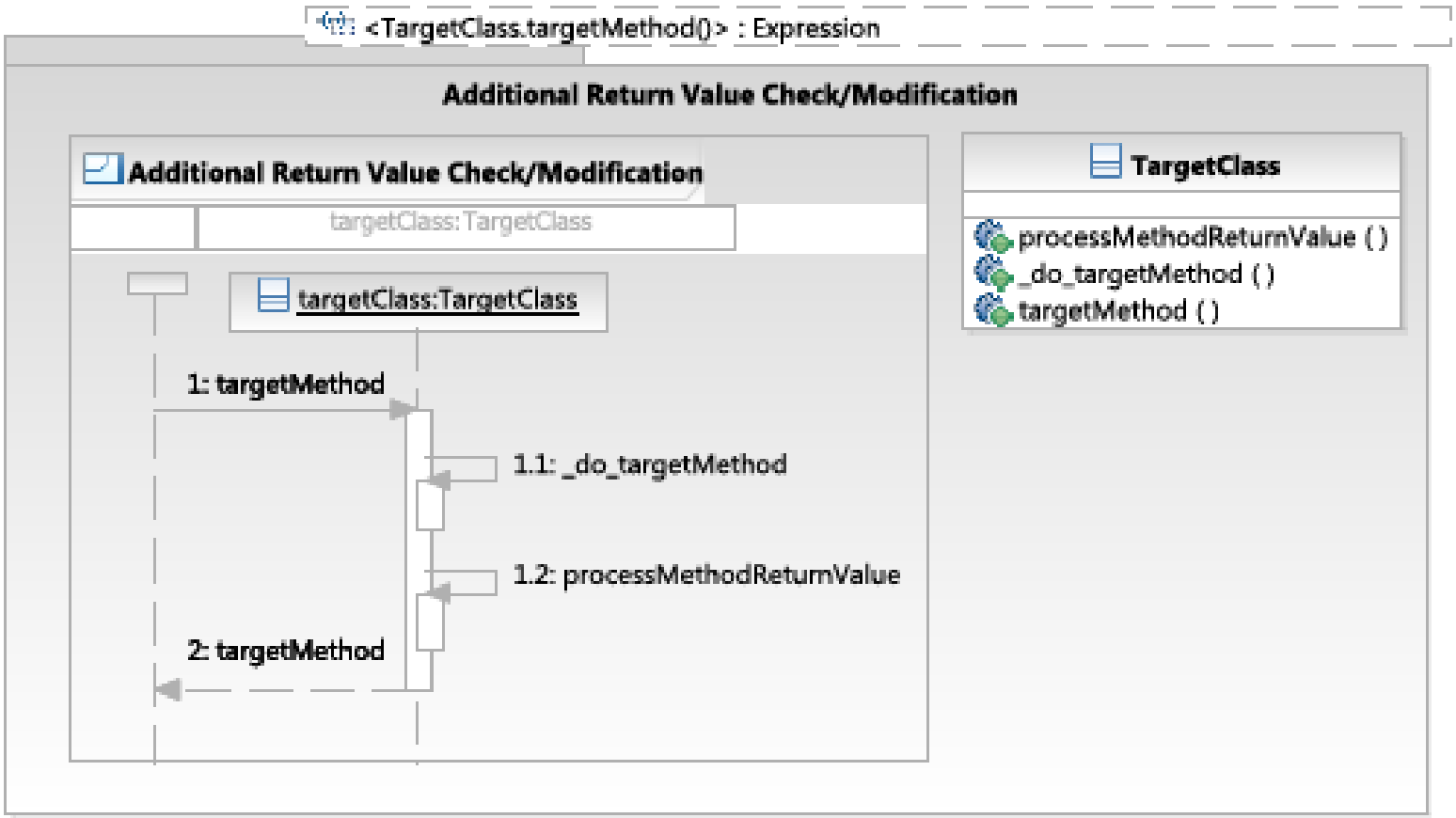


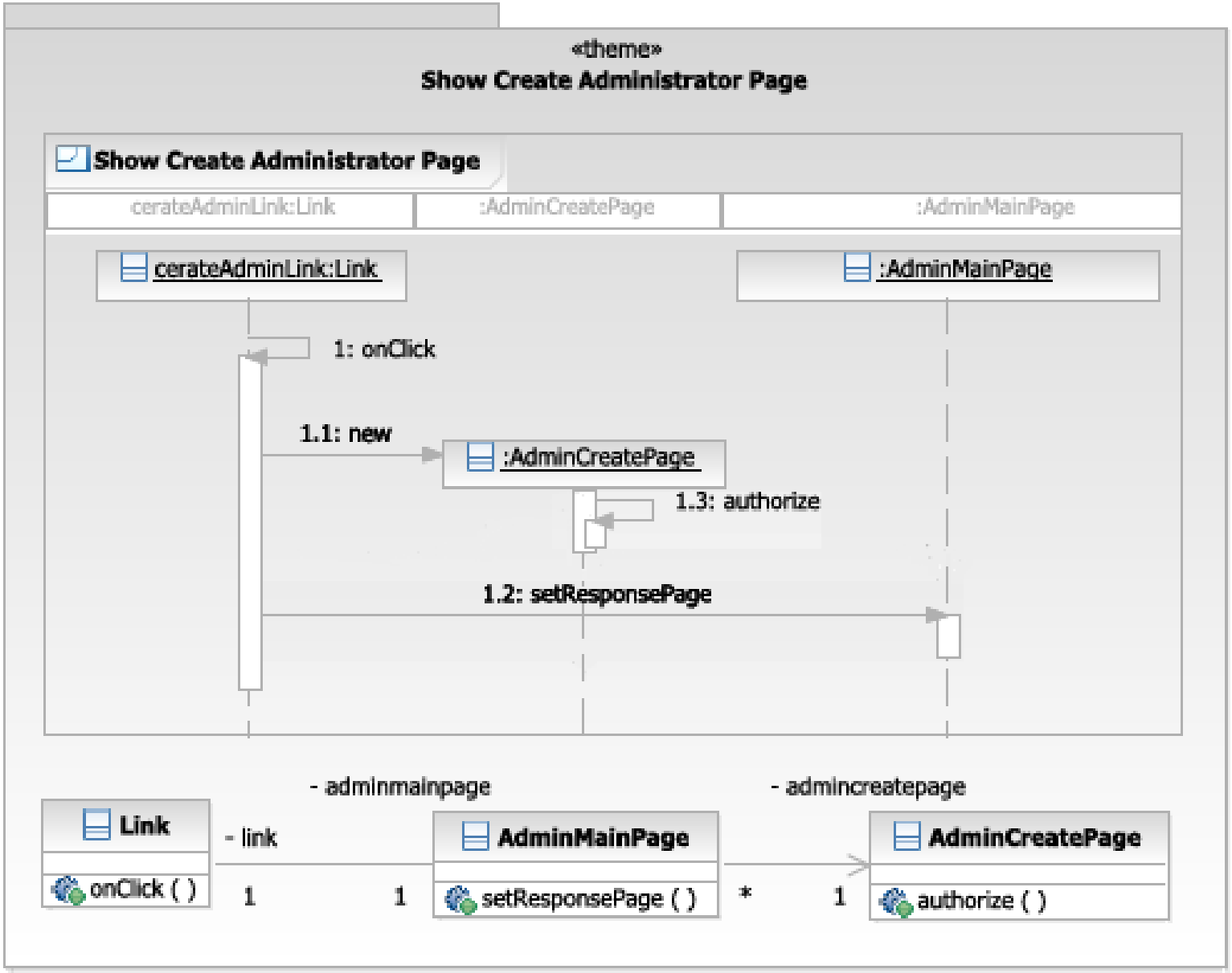
Схема реализационог типа измене

- Каталог садржи и схему реализационог типа измена



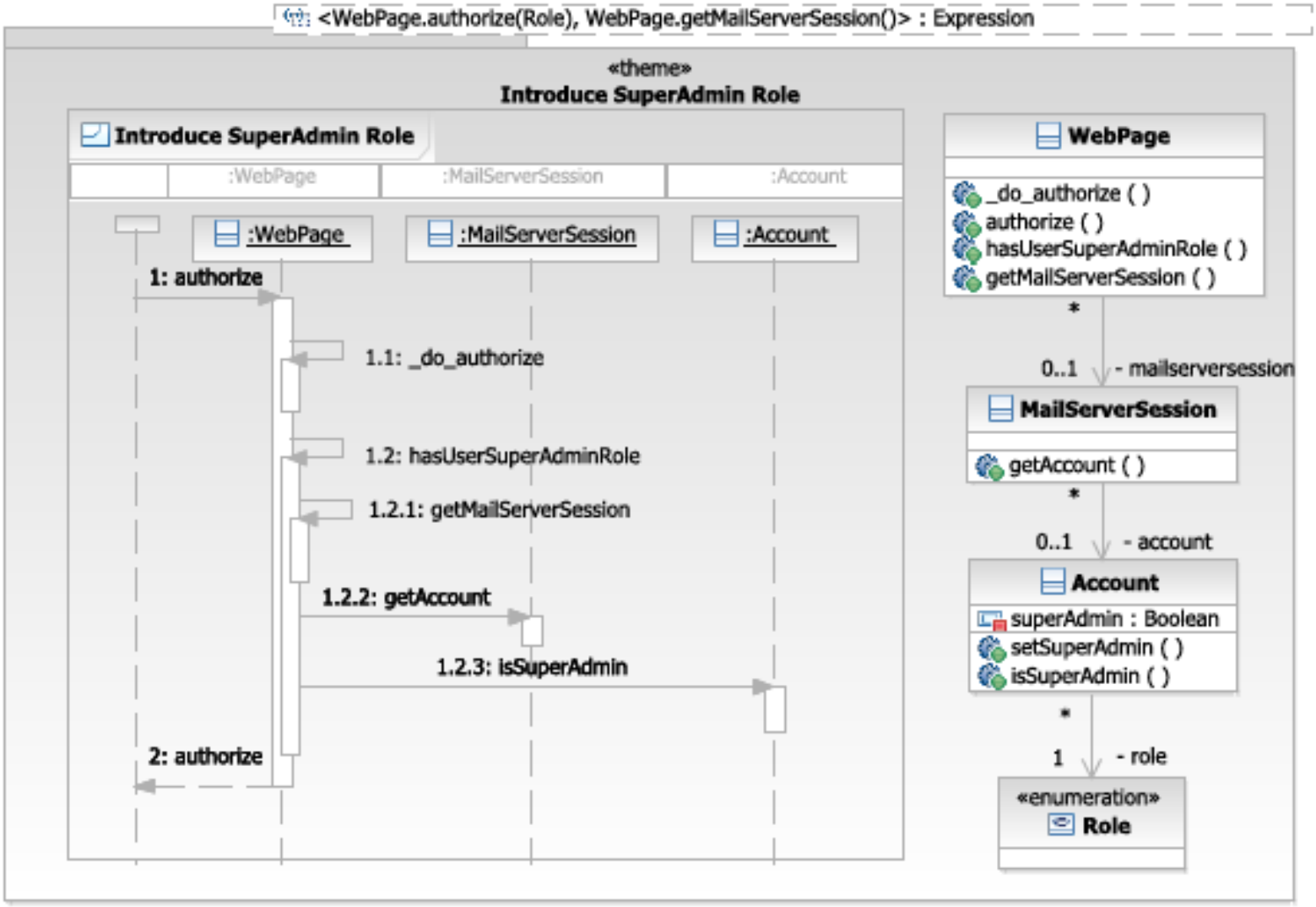
Контекст примене измене

- Контекст у коме треба применити измену



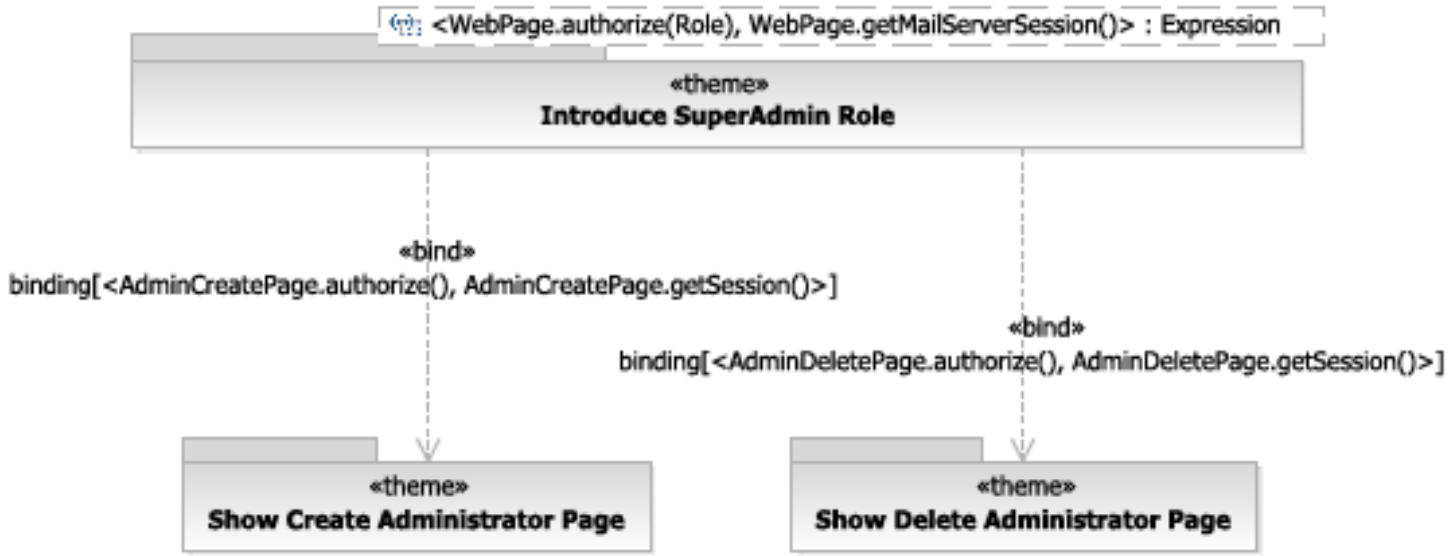
Примена реализационог типа измене (1)

- Схему реализационог типа измена прилагодимо контексту



Примена реализационог типа измене (2)

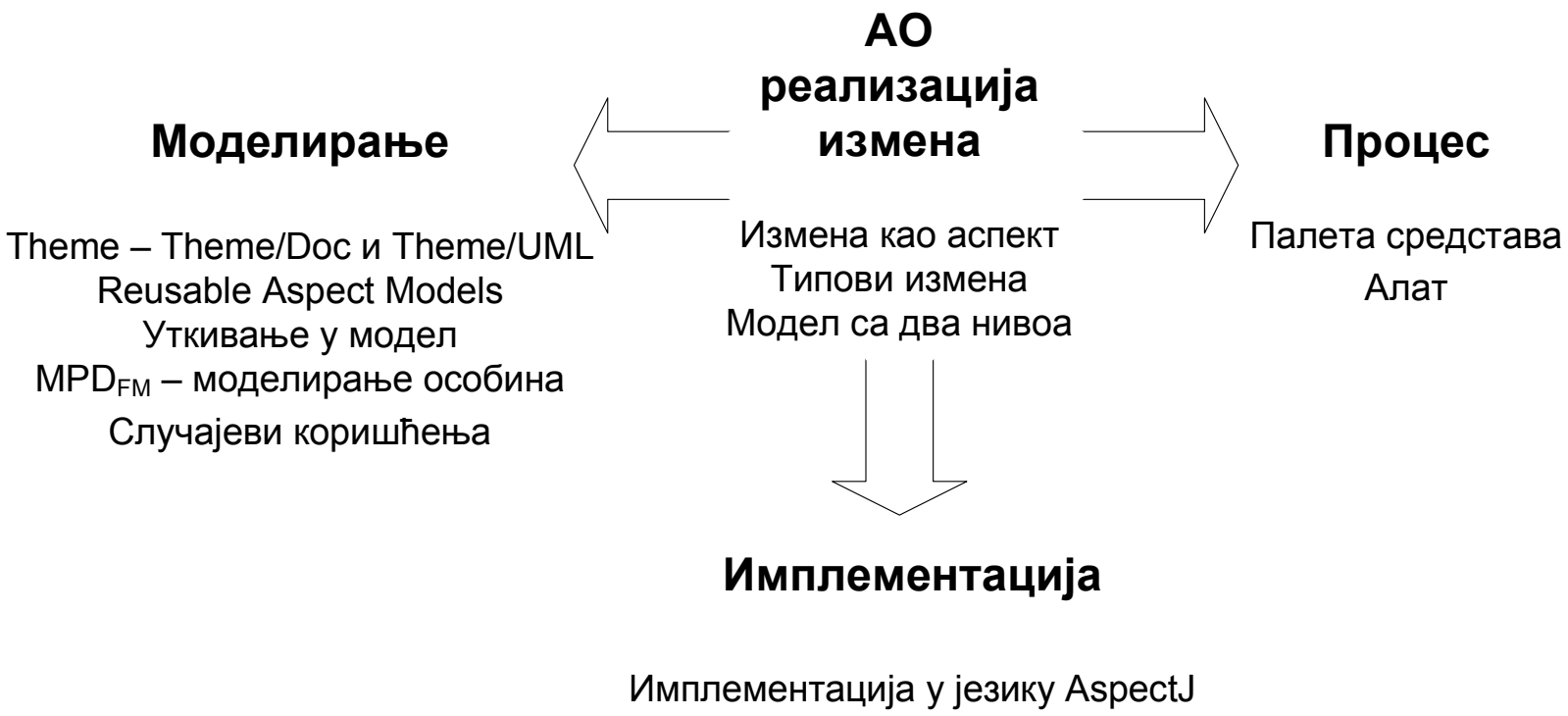
- Измена се везује за контекст путем параметара



Имплементација

- Овакав модел измене може директно да се искаже у језику AspectJ у облику неколико аспеката
- Могуће је применити и схему програмског кода коју каталог такође садржи
- Није потребно вршити било какве директне измене у постојећем програмском коду
- У случају да у одређеним верзијама апликације не желимо суперадминистратора, довољно је изоставити аспекте

Преглед истраживања



Евалуација приступа

- Евалуација на нивоу имплементације
  - YonBan – реално коришћена веб апликација, FIIT STU (менаџмент студентских пројеката)
  - HealthWatcher – евалуациона апликација, Lancaster University (прикупљање података о здравственој ситуацији од јавности)
  - ProductParser – студентски пројект, FIIT STU (претраживање интернет продавница)
- Евалуација на нивоу моделирања
  - Webmail, студентски пројект, FIIT STU
  - Cinema Inviter, студентски пројект, FIIT STU

Сумаризација

- Аспектно-оријентисана реализација измена – мотивација из праксе
- Омогућава модуларизацију измена – једноставно укључивање и искључивање измена
- Приступ подесан за некорективе измене
- Поједностављење примене посредством модела аспектно-оријентисане реализације измена са два нивоа – каталог
- Имплементација у језику AspectJ
- Моделирање у приступу Theme
- Даљи рад: подршка процеса примене аспектно-оријентисане реализације измена

http://fiit.stuba.sk/~vranic/