Prednáška 1: Špecifikácia softvéru a prípady použitia

Metódy a prostriedky špecifikácie 2012/13

Valentino Vranić

Ústav informatiky a softvérového inžinierstva
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave

25. september 2012

Informácie o predmete

O čom bude tento predmet

- Čo získate:
  - prehľad spôsobov vyjadrenia špecifikácie softvéru na všetkých úrovniach vývoja adekvátnymi modelmi
  - hlbšie poznatky o modelovaní prípadov použitia a jazyku UML
  - poznatky o formálnych prístupoch k špecifikácii softvéru
- Čo sa vyžaduje:
  - základné vedomosti o UML a objektovo-orientovanom programovaní sú predpokladom
  - sledovanie prednášok
  - čítanie literatúry počas semestra
  - vypracovanie projektu

Základné informácie

- Rozsah: 2 hodiny prednášok – 2 hodiny cvičení
- 5 kreditov
- Predmet končí zápočtom a skúškou
- Stránka predmetu: `fiit.stuba.sk/~vranic/mps`

Plán prednášok

1. Špecifikácia softvéru a prípady použitia [25. september]
2. Prípady použitia a UML [2. október]
3. Od prípadov použitia k štruktúre v jazyku UML [9. október]
4. Modelovanie štruktúry v jazyku UML [16. október]
5. Jazyk OCL [23. október]
6. Metóda OOram [30. október]
   Test [6. november]
7. Jazyk Z [13. november]
8. Algebraický prístup k špecifikácii softvéru [20. november]
9. Konfigurovateľnosť v modelovaní softvéru [27. november]
10. Metamodel UML [4. december]
11. Scrum [11. december]

Literatúra

- Jim Arlow and Ila Neustadt. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley, 2nd edition, 2005.
- Ivar Jacobson and Pan-Wei Ng. *Aspect-Oriented Software Development with Use Cases*, Addison-Wesley, 2005.
- Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000.
- Gunnar Overgaard and Karin Palmkvist. *Use Cases: Patterns and Blueprints*, Addison-Wesley, 2005.
- Suzanne Robertson and James Robertson. *Mastering the Requirements Process*. Addison-Wesley, 2nd edition, 2005.
- Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997.
- J. M. Spivey. *The Z Notation: Reference Manual*. Prentice Hall, 1992.
- Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof*. Prentice Hall, 1996.
- Trygve Reenskaug. Working With Object: *The OOram Software Engineeringmethod*. Prentice Hall, 1995.
- Krzysztof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

Hodnotenie

- Hodnotenie: semester/skúška – 65/35
- Semestrálne hodnotenie:
  - semestrálny projekt – 45 bodov
  - semestrálny test – 20 bodov
- Semestrálny test bude 6. novembra v čase prednášky (tiež v DE-300)

Organizácia cvičení

- Cvičenie 2: spresnenie rámcového zadania
- Cvičenie 3–5: model prípadov použitia
- Cvičenie 6–7: prezentácia projektov
- Cvičenie 8–10: model štruktúry a správanie systému
- Cvičenie 11–12: prezentácia projektov

Projekt

1. Spresnenie rámcového zadania
   - dokument (1 strana)
   - odovzdanie predbežnej verzie na 2. cvičení
   - odovzdanie finálnej verzie do začiatku 3. cvičenia
2. Model prípadov použitia a iniciálny model správania
   - model prípadov použitia – max. 10 b
   - kolaborácie a diagramy sekvencií – max. 6 b
   - prezentácia – max. 4 b
   - odovzdanie do začiatku 6. cvičenia; prezentácia na 6. a 7. cvičení
3. Štruktúra a správanie systému
   - model štruktúry systému – max. 10 b
   - detailný model správania systému – max. 10 b
   - špecifikácia operácií v jazyku Z – max. 5 b
   - prezentácia – max. 4 b
   - odovzdanie do začiatku 11. cvičenia; prezentácia na 11. a 12. cvičení

Podmienky akceptovania projektu

- Všeobecné podmienky akceptovania:
  1. Projekt musí byť vlastnou prácou študenta, ktorý ho odovzdáva. Plagiáty nebudú tolerované.
  2. Časti projektu realizované v RSA musia byť dodané vo verzii dostupnej v laboratóriu, v ktorom sa realizujú cvičenia.
  3. Časti projektu 2 a 3 musia byť odovzdané a prezentované aspoň raz v pokročilom štádiu rozpracovania pred záverečným odovzdaním.
  4. Časti projektu 2 a 3 musia byť prezentované a obhájené ústne na adekvátnej úrovni.
- \+ Podmienky akceptovania špecifické pre každú časť projektu

Rámcové zadanie

Vypracujte model *systému na správu tokov práce* vo zvolenej organizácii. Systém by mal umožniť uplatnenie tokov práce, čo predpokladá distribúciu úloh relevantným používateľom systému v závislostí od ich rolí. Dokončením svojej úlohy používateľ generuje úlohu pre nasledujúcich používateľov v toku práce. Aj keď definície tokov práce závisia od typu a špecifík organizácie a v zásade budú súčasťou systému, bolo by žiaduce umožniť aspoň ich prispôsobenie.

Obsah prednášky

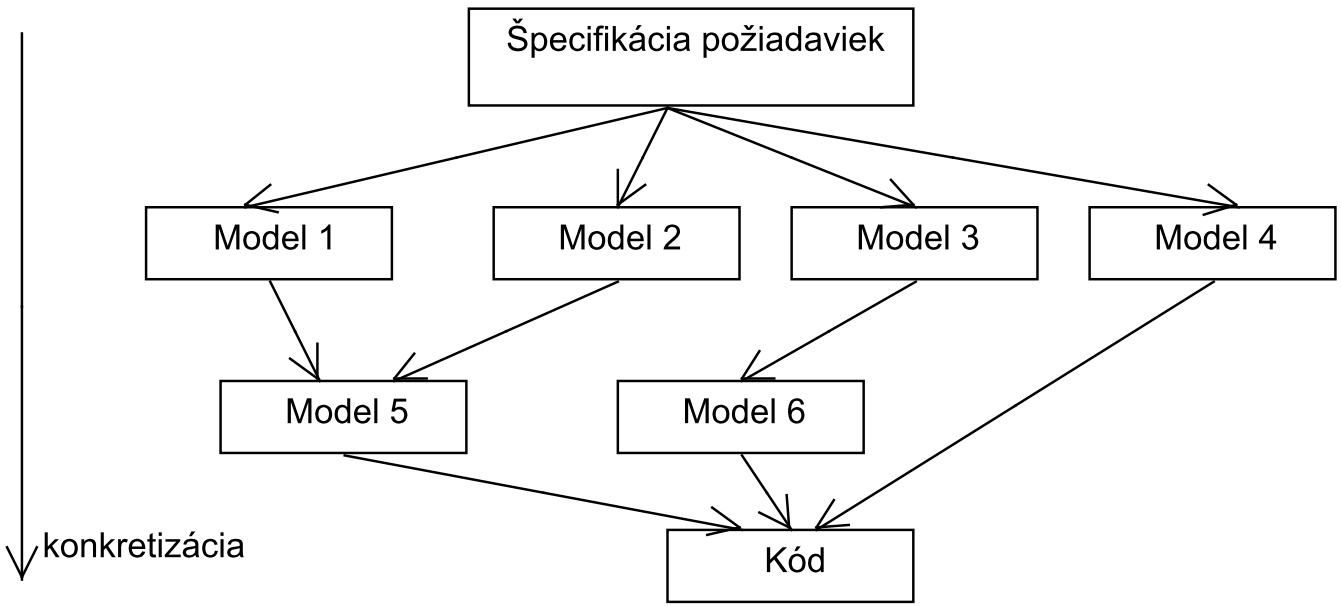
Špecifikácia a modelovanie softvéru

Špecifikácia

- Vo vývoji softvéru termín špecifikácia zvyčajne označuje špecifikáciu požiadaviek
- Špecifikáciu možno chápať širšie: vyjadrenie softvéru na každej úrovni abstrakcie
- Postupnou konkretizáciou získavame programový kód, ktorý predstavuje najpresnejšiu špecifikáciu
- Podľa toho sa tento proces ešte označuje aj ako spresňovanie (refinement)[1]
- Medzi špecifikáciou požiadaviek a kódom je zvyčajne rad modelov, ktoré špecifikujú rôzne aspekty softvéru
- Tieto modely môžu byť tiež postupne spresňované
- Metódy a prostriedky špecifikácie (softvéru)
  $\rightarrow$ *Modelovanie softvéru*

[1] niekedy aj „zjemňovanie“

Konkretizácia modelov softvéru



Modelovanie softvéru (1)

- Analýza a návrh
- Modelovanie:
  - Model ako rámec pre ďalší vývoj (neobsahuje všetky detaily – abstrakcia)
  - Úplný model (úplne špecifikuje softvérový systém)
- Program je tiež model
- Grafické notácie umožňujú ľahšie pochopenie
  - Rôzne notácie
  - Snaha o zjednotenie a štandardizáciu

Modelovanie softvéru (2)

- Dva rozmery modelovania:[2]
  - Statický (štruktúra – zdrojový text)/dynamický (správanie – vykonávanie programu)
  - Logický (problémová oblasť)/fyzický (implementácia)

[2] G. Booch. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 2nd edition, 1994.

Formálna špecifikácia

- Nepresnosť a nejednoznačnosť prirodzeného jazyka sa dá prekonať matematickou formalizáciou
- Možnosť matematického narábania so špecifikáciou: overenie špecifikácie prostredníctvom matematických dôkazov zodpovedajúcich teorém
- Presné, formálne vyjadrenie je často náročné pochopiť
- Jeho interpretácia zase – vyjadrená v prirodzenom jazyku – sa ľahšie chápe, ale vnáša nepresnosť; markantné vo fyzike
- Formalizovať sa dá aj prirodzený jazyk: dochádza k tomu istému javu
- Iný príklad mimo vývoja softvéru: právo – potrebujeme interpretáciu zákonov

Neúplnosť špecifikácie

- Neúplnosť je vnútornou vlastnosťou špecifikácie požiadaviek
- Zákazník nevie, čo chce
- Ako by veci mali byť, zákazník najlepšie posúdi pri skúšaní aplikácie
- Vodopádový model ide priamo proti prirodzenému stavu vecí
- Iteračné a inkrementálne prístupy
- Skoré dodanie funkčných častí softvéru
- Scrum

Požiadavky a prípady použitia

Požiadavky

- Funkcionálne – čo softvér bude poskytovať
- Nefunkcionálne – vlastnosti softvéru, ktoré nepredstavujú bezprostredne funkcionalitu
- Ohraničenia – čo softvér nebude poskytovať – v kontexte funkcionálnych a nefunkcionálnych požiadaviek, ale niekedy aj vo všeobecnosti (ako ochrana)
- Bežný spôsob zachytenia požiadaviek je vo forme textového dokumentu
- Inžinierstvo požiadaviek – requirements engineering
  - Formulácia požiadaviek, ich zaznamenávanie a analýza
  - Generovanie ďalších softvérových artefaktov zo špecifikácie

Požiadavky

- Snažíme sa dosiahnuť úplnú a konzistentnú špecifikáciu požiadaviek
- Vývoj softvéru je však iteračný a inkrementálny
- Požiadavky vznikajú aj počas modelovania a implementácie systému

Úloha špecifikácie požiadaviek

1. Rámec pre vývoj softvéru – pre vyvojárov
2. Stanovenie zmluvného rámca so zákazníkom – má právny dopad, ale slúži aj pre komunikáciu so zákazníkom

Prípady použitia (1)

- Špecifikácia požiadaviek môže zachytávať to, čo používateľ chce, ale nemusí vytvárať jasnú predstavu o typických *prípadoch použitia* systému
- Prípad použitia je vždy nejaká ucelená funkcionalita, ktorá má zmysel a hodnotu pre používateľa
- Základom je zámer používateľa: čo chce dosiahnuť
- Niekedy sa pristupuje priamo k tvorbe prípadov použitia – vtedy hrajú rolu špecifikácie požiadaviek
- Diagram prípadov použitia v jazyku UML – ale prípady použitia sú primárne textové opisy
- Prevažuje opis vo forme postupnosti krokov (v čiastočnom usporiadaní)

Prípady použitia (2)

Vyhľadaj produkt

1. Zákazník zvolí vyhľadanie produktu.
2. Systém zobrazí zoznam všetkých kategórií produktov v tvare stromu iniciálne len s prvou úrovňou kategórií.
3. Zákazník môže otvoriť kategóriu a systém zobrazí zoznam produktov zaradených do nej a podstrom jej podkategórií.
4. Zákazník môže zavriet kategóriu a systém skryje jej obsah.
5. Zákazník môže označiť jeden produkt.
6. Ak zákazník neoznačil žiaden produkt, systém za označený pokladá prvý produkt prvej kategórie.
7. Prípad použitia končí.

Prípadmi použitia riadený vývoj softvéru

- Unified Process – use-case driven
- *Správanie najprv*
  - Conwayov zákon – návrh softvéru kopíruje komunikačné štruktúry v organizácii, ktorá ho vyrába[3]
- Správanie vyjadríme primárne prípadmi použitia
- Presnejšie/názornejšie vyjadrenie správania, ktoré definujú prípady použitia, pomocou (grafických) modelov
- Primárna štruktúra systému sa odvodí z prípadov použitia alebo základného (grafického) modelu správania
- Ďalej sa iteračne a inkrementálne rozpracúva model správania a štruktúry

[3] http://egov.blogs.com/eaglossary/2004/06/conways_law.html

Modelovanie v UML

UML

- *Unified Modeling Language* – zjednotený jazyk modelovania
- UML vyvinula organizácia OMG – Object Management Group
  - Zakladateľ: Rational Software Corporation – Grady Booch, Ivar Jacobson a Jim Rumbaugh
  - Neziskové konzorcium s otvoreným členstvom
  - Rozhodnutia sa prijímajú na základe hlasovania
- Založený na najrozšírenejších OO metódach
- Používa sa predovšetkým pre OO analýzu a návrh – *de facto* štandard
  - Použiteľný aj na iné účely (business modeling)
  - UML je rozšíriteľný

Nástroje pre UML

- Podpora notácie modelovania nástrojom je esenciálna pre jej reálne použitie
- Computer Aided Software Engineering – CASE
- CASE nástroje pre UML:[4]
  - IBM Rational Software Architect and Modeler
  - Enterprise Architect
  - Ďalšie: ArgoUML, Poseidon for UML...
- V nedostatku CASE nástroja, možno použiť editor diagramov:
  - MS Visio
  - Dia[5]
  - UMLet[6]

[4] http://www.uml.org/#Links-UML2Tools
[5] http://www.gnome.org/projects/dia/
[6] http://www.umlet.com/

Druhy diagramov v UML

- Diagramy štruktúry (structure diagrams)
  - Balíky, triedy, objekty a fyzické rozloženie systému
- Diagramy správania (behavior diagrams)
  - Použitie systému, komunikácia, interakcia a stavový model
- Vzťah diagram–model
- Užitočný stručný prehľad UML 2.0 od Sparx Systems[7]

[7] http://sparxsystems.com.au/UML2_tutorial/UML2_Tutorial_Intro.htm, http://www.sparxsystems.com.au/UML_Tutorial.htm

Diagramy štruktúry

- Diagram tried – class diagram
- Diagram objektov – object diagram
- Diagram kompozitnej štruktúry – composite structure diagram
- Diagram rozloženia – deployment diagram
- Diagram komponentov – component diagram
- Diagram balíkov – package diagram

Diagramy správania

- Diagram aktivít – activity diagram
- Diagramy interakcie – interaction diagrams:
  - Diagram sekvencií – sequence diagram
  - Diagram komunikácie – communication diagram (predtým diagram spolupráce – collaboration diagram)
  - Diagram prehľadu interakcií – interaction overview diagram
  - Časový diagram – timing diagram
- Diagram prípadov použitia – use case diagram
- Stavový diagram – statechart diagram

Sumarizácia

Sumarizácia

Sumarizácia

- Špecifikácia softvéru
- Špecifikácia ako model
- Spresnenie špecifikácie
- Prípady použitia
- Modely v jazyku UML