Môžeme sa odkazovať na literatúru?

Prečo sa máme odkazovať na literatúru?

Ako sa odkazovať na literatúru?

„Čerpanie z literatúry“ je eufemizmus pre plagiátorstvo

Je dovolené prevziať obrázok?

Je zoznam odkazov na literatúru zlý?

quote / quotation – uviesť / citát

quotation marks – úvodzovky

cite / citation – citovať / citát

reference – odkaz, referencia

„Premature optimization is the root of all evil.“
– C. A. R. Hoare

Objektovo orientované programovanie predstavuje programovanie pomocou objektov. Objekt je entita, ktorá má [Boo94]:

- stav
- správanie
- identitu

Oddelenie záležitostí (separation of concerns) je pojem z článku Edsgera W. Dijkstru, v ktorom pojednával o „inteligentnom myslení“ vo všeobecnosti: vždy sa sústreďujeme len na jeden aspekt predmetu, ktorý skúmame [Dij82].

Hannemann a Kiczales implementovali vzory GOF aspektovo-orientovaným spôsobom [HK02].

Rozpisovanie atribútov tried podľa dielčom všeobecnej zásady princípu otvorenosti a uzavretosti (open-closed principle) [Mey98]:

Softvérové entity (triedy, moduly, funkcie atď.) majú byť *otvorené pre rozšírenie, ale uzavreté pre zmeny.*

Najdôležitejším kritériom pre použitie dedenia je zachovanie vzťahu špecializácie [Coy94]. Aby použitie dedenia bolo opodstatnené, nestačí len štrukturálna podobnosť, ale predovšetkým skutočnosť, či objekty nadtriedy budú môcť nahradiť objekty podtried. Túto vlastnosť presnejšie formuluje Liskovej princíp substitúcie (Liskov substitution principle) [Lis87]:

> Ak pre každý objekt o_1 typu S jestvuje objekt o_2 typu T taký, že pre všetky programy P definované v zmysle T správanie P je nezmenené, keď sa o_1 nahradí o_2, potom je S podtypom T.

Obrázok 1 [illegible]

Niekedy [illegible] použitie dedenia na rozšírenie triedy sa nazýva jej zovšeobecnenia. Príkladom môže byť odvodenie elipsy od kruhu a podobne ďalšieho obrazca a polomeru [Mar96].

Modern software tends to be large and complex. Therefore it would be very difficult for one person to develop such software and that is why people are developing in teams. The concept of a “team” is described as a small number of people with complementary skills who are equally committed to a common purpose, goals, and working approach for which they themselves mutually accountable. It is important to notice that getting a group of people to work together (physically) is not enough to make this group of people into a “team”. Teams are different from working groups. The first one promises greater performance than the last one [1].

2.1 Teams

The concept of a “team” is described as a small number of people with complementary skills who are equally committed to a common purpose, goals, and working approach for which they hold themselves mutually accountable. It is important to notice that getting a group of people to work together (physically) is not enough to make this group of people into a “team”. Teams are different from working groups. The first one promises greater performance than the last one [8].

Liskovej princíp substitúcie súvisí s návrhom podľa zmluvy (design by contract). V návrhu podľa zmluvy [illegible] [Mey97].

> Študentská práca s doslova prevzatým textom

Program v Jave možno pripraviť aj tak, aby sa dal spustiť ako applet, aj ako samostatná aplikácia [Eck02].

V modelovaní softvéru sa člení dva rozmery: statický/dynamický a logický/fyzický [Boo94]. Daný pohľad v modeli sa môže orientovať viac na štruktúru, t.j. byť statický, alebo na vykonávanie programu, t.j. byť dynamický. Statický pohľad určuje štruktúru softvérového systému, kým dynamický pohľad určuje správanie (vykonávanie) programu.

Hlas o Alexandrových vzoroch sa rozniesol aj mimo architektúry. Táto téma zaujala niektorých [illegible] v oblasti vývoja softvéru, ktorí v roku 1993 vytvorili skupinu známu ako Hillside Group: K. Auer, G. Booch, R. Johnson, H. Hildebrand, K. Beck, W. Cunningham a J. Coplien. Alexander sa rad [illegible] architektom [illegible], aby [illegible] k informatikom, a [illegible] jeho práca mala [illegible] vplyv na informatiku než na architektúru. [illegible] vzorov vo vývoji softvéru možno [illegible] aj ako špecifickú kultúru – kultúru vzorov [Cop96].

Prednáška 3:

Bibliografia a citovanie v technickom texte

Valentino Vranić

Ústav informatiky a softvérového inžinierstva

STU FIIT

vranic@stuba.sk fiit.sk/~vranic

MIP 2015/16

8. 10. 2015

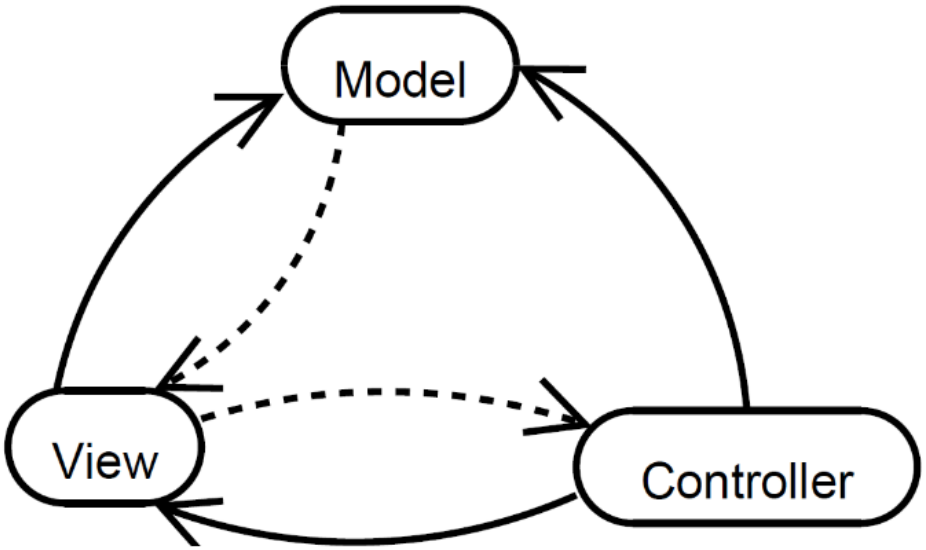
Máme sa odkazovať na literatúru?

Prečo sa máme odkazovať na literatúru?

Ako sa odkazovať na literatúru?

Je mať veľa odkazov na literatúru zlé?

Je dovolené prevziať obrázok?



Obrázok 15.1: Model-View-Controller (podľa [Hel]).

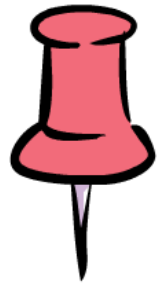
quote / quotation – uviesť / citát

quotation marks – uvodzovky

cite / citation – citovať / citát

reference – odkaz, referencia

„Premature optimization is the root of all evil.“
– C. A. R. Hoare



„Čerpanie z literatúry“ je eufemizmus pre plagiátorstvo

Objektovo-orientované programovanie predstavuje programovanie pomocou objektov. Objekt je entita, ktorá má [Boo94]:

- stav
- správanie
- identitu

Literatúra

[Boo94] Grady Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, second edition, 1994.

[Bra] Gilad Bracha. Generics in the Java programming language. Tutorial, July 2005. http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf.

[Cop92] James O. Coplien. *Advanced C++ Programming Styles and Idioms*. Addison-Wesley, 1992.

[Cop05] James O. Coplien. Culture of patterns. *Computer Science and Information Systems Journal (ComSIS)*, 1(2):1–26, 2005.

[Dij82] Edsger W. Dijkstra. On the role of scientific thought. In *Selected Writings on Computing: A Personal Perspective*, pages 60–66. Springer, 1982. EWD 447, `http://www.cs.utexas.edu/users/EWD/ewd04xx/EWD447.PDF`.

[Eck02] Bruce Eckel. *Thinking in Java*. Prentice Hall, third edition, 2002. Electronic edition, revision 4.0 (final print version).

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GJSB05] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification*. Prentice Hall, third edition, 2005. `http://java.sun.com/docs/books/jls/`.

[Hel] Dean Helman. Model-View-Controller. `http://ootips.org/mvc-pattern.html`. comp.object news transcript, May 14, 1998.

[HK02] Jan Hannemann and Gregor Kiczales. Design pattern implementation in Java and AspectJ. In *Proceedings of 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2002*, pages 161–173, Seattle, USA, November 2002.

[KLM+97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Christina Vidiera Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In Mehmet Aksit and Satoshi Matsuoka, editors, *Proc. of 11th European Conference on Object-Oriented Programming (ECOOP'97)*, LNCS 1241, Jyväskylä, Finland, June 1997. Springer.

Skrývanie atribútov tried je dôsledkom všeobecnejšieho princípu otvorenosti a uzavretosti (open-closed principle) [Mar96b]:

> Softvérové entity (triedy, moduly, funkcie atd.) majú byt otvorené pre rozšírenie, ale uzavreté pre zmeny.

Program v Jave možno pripravit aj tak, aby sa dal spúštat ako applet, aj ako samostatná aplikácia [Eck02].

Liskovej princíp substitúcie súvisí s návrhom podla zmluvy (design by contract). V návrhu podľa zmluvy operácie vystupujú ako zmluvné strany [Mey97].

V modelovaní softvéru sa sledujú dva rozmery: statický/dynamický a logický/fyzický [Boo94]. Daný pohľad v modeli sa môže orientovat viac na štruktúru, t.j. byť statický, alebo na vykonávanie programu, t.j. byť dynamický. Statický pohlad určuje štruktúru zdrojového textu, kým dynamický pohlad urcuje správanie (vykonávanie) programu.

Hlas o Alexandrových vzoroch sa rozniesol aj mimo architektúry. Táto idea zaujala niektorých prominentov v oblasti vývoja softvéru, ktorí v roku 1993 vytvorili skupinu známu ako Hillside Group: K. Auer, G. Booch, R. Johnson, H. Hilerbrand, K. Beck, W. Cunnigham a J. Coplien.[1] Alexander sa stal prvým architektom pozvaným, aby prehovoril k informatikom, a zdá sa, že jeho práca mala väcší vplyv na informatiku než na architektúru. Fenomén vzorov vo vývoji softvéru možno vnímat dokonca aj ako špecifickú kultúru – kultúru vzorov [Cop05].

Niekedy láka nesprávne použitie dedenia na rozšírenie triedy na úcely jej zovšeobecnenia. Príkladom môže byt odvodenie elipsy od kruhu s pridaním dalšieho ohniska a polomeru [Mar96a].

Najdôležitejšie kritérium pre použitie dedenia je existencia vzťahu typ-podtyp [Cop92]. Aby použitie dedenia bolo opodstatnené, nestačí teda štrukturálna podobnosť, ale predovšetkým sledujeme, či objekty uvažovanej nadtriedy bude možné nahradiť objektmi jej podtried. Túto vlastnosť presnejšie formuluje Liskovej princíp substitúcie (Liskov substitution principle) [Lis87]:

> Ak pre každý objekt o1 typu S jestvuje objekt o2 typu T taký, že pre všetky programy P definované v zmysle T správanie P je nezmenené, keď sa o1 nahradí o2, potom je S podtypom T.

Hannemann a Kiczales implementovali vzory GOF aspektovo-orientovaným spôsobom [HK02].

Oddelenie záležitostí (separation of concerns) je pojem z článku Edsgera W. Dijkstru, v ktorom pojednával o „inteligentnom uvažovaní" vo všeobecnosti: vždy sa sústredujeme len na jeden aspekt predmetu, ktorý skúmame [Dij82].

Modern software tends to be large and complex. Therefore it would be very difficult for one person to develop such software and that is why people are developing in teams. The concept of a "team" is described as a small number of people with complementary skills who are equally committed to a common purpose, goals, and working approach for which they themselves mutually accountable. It is important to notice that getting a group of people to work together (physically) is not enough to make this group of people into a "team". Teams are different from working groups. The first one promises greater performance than the last one [1].

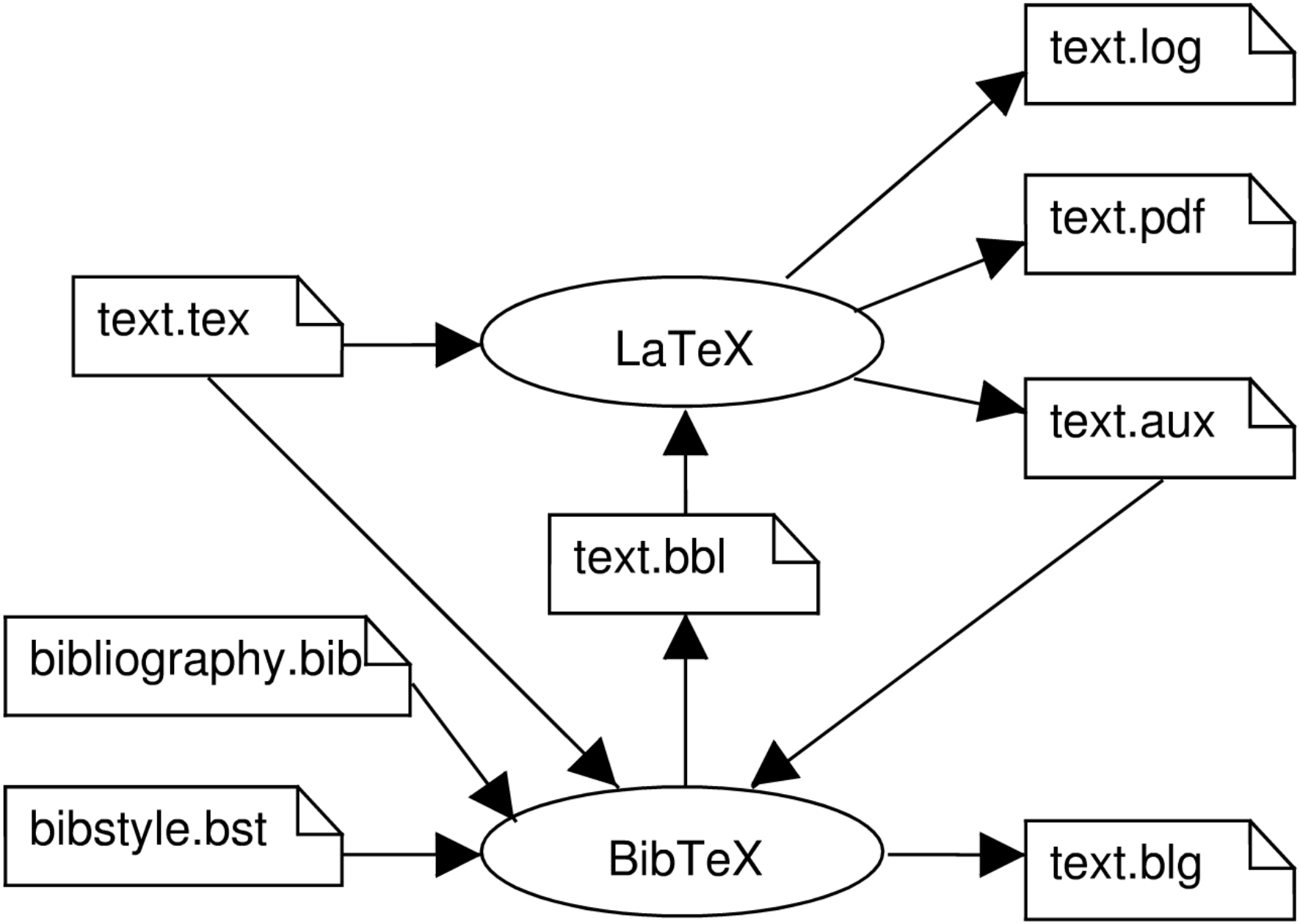
Modern software tends to be large and complex. Therefore it would be very difficult for one person to develop such software and that is why people are developing in teams. The concept of a "team" is described as a small number of people with complementary skills who are equally committed to a common purpose, goals, and working approach for which they themselves mutually accountable. It is important to notice that getting a group of people to work together (physically) is not enough to make this group of people into a "team". Teams are different from working groups. The first one promises greater performance than the last one [1].

2.1 Teams

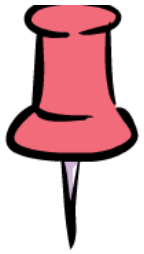
The concept of a “team” is described as a small number of people with complementary skills who are equally committed to a common purpose, goals, and working approach for which they hold themselves mutually accountable. It is important to notice that getting a group of people to work together (physically) is not enough to make this group of people into a “team”. Teams are different from working groups. The first one promises greater performance than the last one [8].

M. M. N. Zenun, G. Loureiro, and C. S. Araujo. The Effects of Teams' Co-location on Project Performance. In Complex Systems Concurrent Engineering. Springer 2007.

> Študentská práca s doslova prevzatým textom


text.log
text.pdf
text.tex
LaTeX
text.aux
text.bbl
bibliography.bib
bibstyle.bst
BibTeX
text.blg

http://tinyurl.com/mip-pred03



> Odkazujte sa na literatúru: lepšie viac ako menej

> Z literatúry nečerpajte, ale poukazujte na to, čo ste tam našli, a konfrontujte to s inými názormi, vrátane vlastného

> LaTeX a BibTeX za vás zabezpečia technickú stránku odkazovania sa na literatúru