TEORETICKÝ PŘÍSTUP K TVORBĚ UŽIVATELSKÉHO ROZHRAŇÍ SOFTWAREVÝCH SYSTÉMŮ

M. Mišovič

Došlo: 30. června 2008

Abstract

MIŠOVIČ, M.: *A theoretical approach to graphical user interface development of software systems*. Acta univ. agric. et silvic. Mendel. Brun., 2009, LVII, No. 3, pp. 107–114

A graphical user interface (GUI) of any web-based object or classical structural enterprise business software constantly hasn't lost its user importance. It really stays as a one of very relevant manners for a client communication with any contemporary developed software units (programs, applications, modules and Software Complex Systems). In addition to this mission, every GUI is regarded as a one of three considerable basic software unit layers. GUI units can be inherently positioned inside software units or we can separate them and construct a special GUI system over GUI units. Naturally, we have to define a manner of communication between software units and GUI units. This approach enables, on a system platform, to investigate not only structure and functionality of GUI units, but also relations among them. We can use the system platform on three levels, a theoretical level, a level of GUI modeling and a level of GUI development. Especially the theoretical investigation of the GUI system can bring for analysts and developers-programmers a new knowledge about the GUI units' behavior and relations. It can equip them by very detailed information about formal tools concerning any GUI system functionality description and help them to improve a process of GUI system construction.

GUI unit, GUI system, GUI resources, GUI modeling techniques, GUI functionality, GUI relations, GUI development techniques

Uživatelské rozhraní představuje jednu ze základních vrstev každého složitěho programového vybavení, tedy rovněž business software podnikového informačního systému (PIS). Mluví se o tom, že mnohé z jednotek GUI (GUI units) jsou asociovány s jistými podnikovými procesy/službami, z jejichž funkcionalit vycházejí. Pochopitelně, existují rovněž jednotky GUI asociované s nepodnikovými procesy/službami. Informační a softwarové inženýrství poskytují ve svých metodikách rovněž metody, techniky a nástroje pro modelování a implementaci uživatelského rozhraní.

Problematicu GUI můžeme vedle rovin modelování a implementace zkoumat rovněž v teoretické – systémové rovině a doplnit obě zmíněné roviny o cenné obecné poznatky. Uvedené roviny sledují zejména následující cíle:

1. Rovina teoretická – systémová. V této rovině je GUI složitěho business software chápáno především jako systém, postavený na tzv. jednotkách

GUI, které jsou svázány jistými interními vazbami – relacemi. Systém GUI jednotek (GUI System) má okolí, ve kterém jsou podnikové procesy nebo podnikové služby a klienti informačního systému. Vedle již zmíněných interních vazeb potom existují rovněž vazby externí na prvky v okolí. Systém GUI jednotek je cílevědomě analyzován – poznáván, určuje se jeho struktura (prvky a vazby) a jeho funkcionalita. Cílem uplatnění systémové roviny je tedy oddělení jednotek GUI od zdrojových podnikových procesů/služeb, z nichž vycházejí a analýza jejich interních a externích vazeb. Formalizace zmíněných vazeb a jejich grafická vizualizace může vést k modelu, který je výchozím pro rovinu modelování a odráží se rovněž v rovině implementační. Jinými slovy, výsledky teoretické roviny by měly ovlivňovat obě dvě zbývající roviny.

2. Rovina modelování reálného systému GUI. Tato rovina zahrnuje analýzu návrhu jednotek GUI,

kteřé vycházejí z podnikových procesů/služeb. Výchozí jsou výsledky předchozí systémové roviny. Analytik, který pochopil podnikové a nepodnikové procesy/služby, využije současných metod, technik a nástrojů pro vytvoření modelu systému GUI. Vzniklý model v UML jazyku je potom základem pro implementační rovinu.

3. **Rovina implementační.** Je to rovina fyzického návrhu GUI a jeho realizace s ohledem na plánovanou architekturu podnikového aplikačního software (business software). Implementace probíhá buď výlučně na platformě jednoho ze známých modelovacích a implementačních paradigmat (strukturovaný klasický přístup, objektový přístup), nebo je smíšená. V souvislosti s vybraným paradigmatem se rovněž vybírá konzistentní vývojové prostředí pro software PIS a jeho GUI.

Vývoj GUI systému je týmová záležitost, ale rozhodující roli hrají systémový architekt PIS (System

Architect), systémový analytik (System Analyst), business analytik (Business Analyst) a implementační vývojář (Developer). Obr. 1 dokumentuje vztah členů týmu a jednotlivých rovin zkoumání problematiky GUI.

Ačkoliv jednotky GUI vystupují v GUI systému jako nedělitelné celky, přesto je v implementační rovině významná jejich výstavba pomocí tzv. elementárních jednotek GUI (tlačítko, textové políčko, seznam aj.) a rovněž je významná vnitřní struktura a vazby mezi elementárními jednotkami. Stavební elementární GUI jednotky jsou v mnoha vývojových prostředích instancemi vestavěných objektových tříd a mohou se standardně používat. GUI jednotky potom působí jako ucelené kontejnery vzájemně svázaných instancí elementárních jednotek GUI, přičemž tato struktura a vazby ovlivňují strukturu komunikačního interface každé jednotky GUI. Tato problematika není v příspěvku rozebírána.

1: Účast v jednotlivých rovinách vývoje GUI

Pokusme se teď dát odpovědi na dvě otázky:

1. Co vše má vliv na vyvíjené GUI jednotky business software?
2. Co vše ovlivňuje vývoj GUI jednotek?

Vyvíjené GUI jednotky jsou ovlivňovány:

- architekturou business software PIS
- podnikovými/nepodnikovými procesy/službami
- interními vazbami mezi jednotkami GUI a externími vazbami mezi jednotkami GUI a procesy/podnikovými službami
- rozsahem stavebních elementárních jednotek GUI vývojového prostředí.

Mezi vlivy na vývoj GUI jednotek musíme řadit:

- teoreticko-systémové poznatky
- techniky a nástroje modelování

- použité vývojové paradigma a jeho metody, techniky a nástroje.

Problematicu obou odpovědí rozvineme v následujících kapitolách.

MATERIÁL A METODY

Současný podnikový business software je postaven na jedné z dvou architektur: aplikační architektury (AA) nebo servisně orientované architektury (SOA). Architektura AA používá jako nejnížší programovou jednotku klasickou softwarovou aplikaci. Vyšší jednotky balíky a nižší balíčky aplikací jsou vestavěny na aplikacích náležejících k témuž procesnímu setu (ERP, SCM, CRM, BI aj.), resp. procesnímu subsetu. Podnikový aplikační software je potom popisován skupinou symbolických rovnic:

$BS = PA^{ERP} \cup PA^{SCM} \cup PA^{CRM} \cup PA^{BI} \cup \dots$ ..... hlavní strukturální rovnice,

$PA^{(i)} = PCA_1^{(i)} \cup PCA_2^{(i)} \cup \dots \cup PCA_k^{(i)}$ ..... rovnice struktury balíků

BS ... označení business software

$PA^{(i)}$ ... označení aplikací pro libovolný procesní set

$PCA^{(i)}$ ... balíček aplikací pro jeden procesní subset. Dále, $i \in \{ERP, SCM, CRM, BI, \dots\}$, $k \in \mathbb{N}$, $k \geq 1$

Pro vývoj GUI jednotek se používají vestavěné grafické editory a klasické skriptové programování. Elementární stavební GUI jednotky mají ovšem objek-

tovou povahu a vznikají jako instance vestavěných objektových tříd ve vývojovém prostředí. U architektury AA jsou GUI jednotky pevně spjaté s podni-

kovými procesy, nemají samostatnou povahu a jejich modifikace je náročnější než v architektuře SOA.

Pro servisně orientovanou architekturu je zdrojem tzv. servisně orientovaná architektura podniku (SOE – Service Oriented Enterprise) postavená na **podnikových službách**. Každá podniková služba může obsahovat transakci nebo jeden a více podnikových procesů spojených na základě výsledku work-flow analýzy. Podnikový business software je potom zalo-

žen na volání komputelizovaných podnikových služeb, které mohou být interní (podnikové) a externí (podniku poskytované). Obecně je možné realizovat zabalení odkazů na komputelizované podnikové služby opět do aplikací a shromáždit aplikace do balíčků a balíků podle procesních subsetů a setů. Podnikový business software je potom popisován následující skupinou symbolických rovnic:

$$\begin{aligned} \mathbf{BS} &= \mathbf{PS}^{\text{ERP}} \cup \mathbf{PS}^{\text{SCM}} \cup \mathbf{PS}^{\text{CRM}} \cup \mathbf{PS}^{\text{BI}} \cup \dots \dots \dots \text{hlavní strukturální rovnice} \\ \mathbf{PS}^{(i)} &= \mathbf{PCS}_1^{(i)} \cup \mathbf{PCS}_2^{(i)} \cup \dots \cup \mathbf{PCS}_k^{(i)} \dots \dots \dots \text{rovnice struktury balíčků} \\ \mathbf{PS}^{(i)} &\dots \text{balík aplikací pro podnikové služby jednoho procesního setu} \\ \mathbf{PCS}^{(i)} &\dots \text{balíček aplikací pro podnikové služby jednoho procesního subsetu.} \end{aligned}$$

Servisně orientovaná architektura podnikového business software je založena na webové platformě a objektovém paradigmatu. Objektové paradigma plně ovlivňuje rovněž implementaci každé GUI jednotky, která je asociovaná s jistou podnikovou službou a spolupracuje s ní na základě komunikačního protokolu a je od ní oddělena. Tedy, nejen pro každou komputelizovanou podnikovou službu se deklaruje jistá třída, ale podobně to bude i pro každou jednotku GUI.

Pro další úvahy, týkající se implementační roviny, budeme zásadně uvažovat jen SOE a implementační SOA architekturu. To nám umožní zavést implementační třídy a instance pro jednotlivé podnikové služby a totéž pro jednotky GUI. Tím se rovněž v implementační rovině, a nejen v teoretické a rovině modelování, oddělí podnikové služby od asociovaných jednotek GUI.

Filozofie GUI systému

Vysoká společenská poptávka na komputizaci reálných procesů ve společnosti, školách, podnicích apod. silně evokuje snahy informatiků urychlit a zkvalitnit vývoj software a přinést do směru RAD (Rapid Application Development) nové metody, techniky a nástroje. Současné dva směry, zahrnuté v RAD, mají odlišnou orientaci. První se orientuje na tvorbu nových metodik pro vývoj software a jeho řízení, kdežto druhý se orientuje na metody, techniky a nástroje urychlující a zkvalitňující vývoj základních vrstev software. To se pochopitelně týká i vrstvy prezentační, jejíž základem je GUI. Pokusíme se proto analyzovat situaci vztahu business software a z něj vyděleného GUI, založeného na GUI jednotkách asociovaných s podnikovými aktivitami, přesněji podnikovými službami, které jsou základem již zmíněné softwarové architektury SOA.

Formální pohled na GUI systém

Hlavním zdrojem GUI podnikového business software s architekturou SOA jsou podnikové a nepodnikové služby, vytvořené z podnikových a nepodnikových procesů nebo transakcí. Mnoho nepodnikových služeb vznikne právě v době návrhu

business software. Každá GUI jednotka je potom asociovaná alespoň s jednou podnikovou nebo nepodnikovou službou.

Proč chceme mluvit o GUI systému? Jsou pro to některé relevantní důvody? Ano, jsou. Např. jimi mohou být následující důvody:

- GUI jednotky jsou celky, které můžeme považovat za prvky jistého systému a na těchto prvcích definovat strukturu systému.
- GUI jednotky mají vzájemné vazby – relace, dále mají vazby na okolí GUI systému, ve kterém jsou podnikové služby a klienti služeb.
- GUI jednotky lze využívat různými podnikovými/nepodnikovými službami a tak pro ně zabezpečit vizualizaci zpracování dat a komunikaci s klienty.

U GUI jednotek můžeme mluvit o vlastnostech, struktuře, operacích, událostech a stavech.

V systémové rovině jsou relevantní zejména všechny výše uvedené důvody. Označme libovolnou podnikovou/nepodnikovou službu symbolem **s** a libovolnou GUI jednotku symbolem **g**. Konečnou množinu podnikových a nepodnikových služeb označme symbolem **S** a konečnou množinu všech GUI jednotek symbolem **G**.

GUI systém je definován notací $\mathbf{S}_{\text{GUI}} = (\mathbf{G}, \mathbf{R}, \mathbf{R}')$, kde $\mathbf{R} \subseteq \mathbf{G} \times \mathbf{G}$ a $\mathbf{R}' \subseteq \mathbf{G} \times \mathbf{S}$. Množina **R** je množinou vnitřních vazeb a **R'** zase vnějších vazeb. Prvky množiny **S**, tj. služby, jsou umístěné v okolí systému GUI, což rovněž ilustruje obr. 2. Povahy vazeb, reprezentovaných společnou neorientovanou tečkovanou spojnici $\dots\dots\dots$, jsou upřesněny v dalším textu.

Dále budeme předpokládat, že každá GUI jednotka **g** nebo podniková služba **s** mají vstupní datový protokol **Input**, vnitřní paměť **Mem** (memory) a výstupní datový protokol **Output** a množinu operací **Oper** (operations). Příslušnost protokolů a pamětí určujeme indexy, tj. jmény podnikových služeb nebo GUI jednotek, např. **Input_g**, **Mem_g**, **Output_g**, **Oper_g**. Systémovou spolupráci mezi podnikovými službami a GUI jednotkami a mezi GUI jednotkami samotnými potom můžeme formalizovat prostřednictvím operací z množiny **Oper_g**, které zmíněné protokoly zpracovávají.

2. Pochopitelně, u každé z relací $\mathfrak{R}_1$ až $\mathfrak{R}_7$ můžeme zkoumat její vlastnosti (reflexivnost, symetričnost, tranzitivnost a dichotomii). Např. relace π je tranzitivní, protože platí implikace $(g \pi g' \pi g'') \Rightarrow g \pi g''$. Takovou ovšem není relace $\rightarrow$. Relace $\parallel$, $\perp$ jsou zase reflexivní. Tento směr zkoumání relací mezi GUI jednotkami ponecháme stranou. Rovněž není uskutečněno důsledné zkoumání vzájemných souvislostí jednotlivých relací.
3. Nad mnoha relacemi můžeme vytvářet řetězce, které jsou výsledkem GUI work-flow analýzy (implementace work-flow analýzy na GUI systém).
4. Přehled relací v GUI systému není úplný, množinu relací lze jistě rozšířit.

Operace nad GUI jednotkami

V našich úvahách byla GUI jednotka povýšena na samostatný prvek a podle toho s ní musíme mani-

pulovat. Každou GUI jednotku, prvek systému GUI, můžeme potom vyšetřovat alespoň ze čtyř relevantních hledisek: struktury, vlastností, funkcionality, událostí a stavů. Z těchto hledisek budeme diskutovat jen první tři.

Struktura GUI jednotky je postavena na vnořených elementárních stavebních jednotkách GUI a jiných GUI jednotkách a vazbách mezi nimi. Pojetí struktury lze rozvinout až ke strukturálním vzorům, tj. opakovaně se vyskytujícím GUI jednotkám se stejnou strukturou.

Struktura GUI jednotky vychází z nároků na vizualizaci zpracování dat nebo nároků komunikace s klientem v dané podnikové službě, s níž je GUI jednotka funkcionálně spojena. V dalších úvahách budeme každou GUI jednotku považovat za nedělitelný celek.

Vlastnosti GUI jednotky g představují její deskriptce, podobné následujícím:

poloha na pracovní ploše a parametry velikosti obrysového rámce elementární GUI jednotka, která získá aktivitu po oživení	vnitřní paměť Mem_g a její členění vstupní protokol Input_g výstupní protokol Output_g viditelnost/neviditelnost

Uvažujme symbol $\Rightarrow$ jako operátor ukládání. Do funkcionality GUI jednotky g často řadíme např. následující operace, základ množiny Oper_g :

- o_1 **Oživení GUI jednotky.** Oživení proběhne podle typu jednotky. Např. pro grafický typ se zobrazí grafická podoba GUI jednotky g na pracovní ploše a oživí se „fokusová“ elementární GUI jednotka (operace je předznačena ve volání od asociované služby s nebo od jiné GUI jednotky g'). Vždy se zobrazí paměť Mem_g rozprostřená na elementární GUI jednotky.
- o_2 **Umrtní GUI jednotky.** GUI jednotka g se odstraní z pracovní plochy. Operace může být vyvolána ze služby s nebo z jiné GUI jednotky g' , která je rovněž na pracovní ploše. Paměť Mem_g se zachovává.
- o_3 **Převzetí komunikačního protokolu.** Jednotka g musí být živá. Vstupní protokol Input_g se převzme a uloží do paměti Mem_g jednotky g . Tedy $\text{Output}_{s,g'} \Rightarrow \text{Input}_g \Rightarrow \text{Mem}_g$. Paměť Mem_g se zobrazí.
- o_4 **Množina specifických operací.** Operace se provádí nad pamětí Mem_g . Jsou to operace, které jednoznačně vycházejí z funkcionality asociované podnikové služby. Tedy, $\otimes \text{Mem}_g \Rightarrow \text{Mem}_{g'}$, kde $\otimes$ je nespecifikovaná unární operace. Zobrazení paměti Mem_g závisí od dané operace.
- o_5 **Množina modifikačních operací GUI jednotky.** Je možno měnit vlastnosti, strukturu, apod. GUI jednotky g .
- o_6 **Operace pro modifikační efekty.** Jde převážně o nastavování událostí GUI jednotky g .
- o_7 **Reakce na události.** Jsou to operace specificky reagující na každou zachycenou událost GUI jednotky g .

- o_8 **Předání komunikačního protokolu.** Jednotka g předá aktuální stav své paměti Mem_g jako komunikační protokol Output_g , když spouští službu s nebo jinou GUI jednotku g' . Tedy, $\text{Mem}_g \Rightarrow \text{Output}_g \Rightarrow \text{Input}_{g,s}$.

Na základě uvedených operací a symbolu $\Rightarrow$ pro kooperaci podnikových služeb a GUI jednotek můžeme pro vývojáře sestavit množinu kooperačních diagramů, které jsou konzistentní s grafem relací v GUI systému. Hrany kooperačních diagramů mají tvar

$s, g \stackrel{\Rightarrow}{\rightarrow} g'$. Nad symbolem kooperace potom píšeme operaci, o jejíž provedení kooperace

GUI jednotku g' žádá. Dokonce lze oba zmíněné diagramy sloučit v jeden relačně-kooperační diagram.

Poznámka 2

Vedle GUI jednotek, které jsou asociovány s podnikovými službami, existují rovněž GUI jednotky asociované s nepodnikovými službami. Mnoho takových služeb vzniká ve fázi hrubého a detailního návrhu business software PIS. Naše úvahy platí i pro tyto služby a s nimi asociované GUI jednotky.

Modelování GUI systému

K modelování GUI systému můžeme použít běžných diagramů používaných v objektovém paradigmatu. Je to cesta jednotného přístupu prostřednictvím objektového paradigmatu ke zdrojům GUI a samotným jednotkám GUI. To znamená, že podobně jak navrhujeme třídy (business třídy) pro podnikové služby, navrhujeme i třídy pro jejich jednotky GUI. Sestrojíme dva kooperující anebo jeden jednotný model pro oba typy tříd. Model tříd GUI jednotek bude obsahovat jejich požadované dědičnosti a relace (informační asociace, agregace aj.).

Dále, podobně jako jsme modelovali spolupráci mezi podnikovými službami, budeme modelovat i jejich spolupráci s jednotkami GUI a spolupráci GUI jednotek navzájem. Spolupráci podnikových služeb a GUI jednotek, GUI jednotek mezi sebou navzájem můžeme modelovat pomocí UML diagramů objektové spolupráce a sekvenčních diagramů na základě zasílání a přijímání zpráv. Diagram vzájemné objektové spolupráce GUI jednotek je potom vlastně GUI navigačním diagramem (GUI Navigation Diagram).

Vedle těchto velmi důležitých diagramů se nevyhneme diagramu struktury GUI jednotek (GUI Layout Diagram). Tento diagram načrtne business analytik pro každou GUI jednotku jednotlivě, vycházející z funkcionálních požadavků a požadavků vizualizace a komunikace s klientem dané asociované podnikové/nepodnikové služby. Všechny uvedené diagramy, tj. diagram tříd pro podnikové služby, diagram tříd pro jednotky GUI, diagram objektové spolupráce služeb a GUI jednotek, diagram

struktury GUI, těsně souvisejí a prezentují jednotný pohled na zdroje GUI a GUI samotné.

Business software a vývoj jeho GUI systému

Implementace rozhraní aplikací prošla rychlým vývojem. Na jedné straně klasické programování těsně svázalo GUI se zdrojem aktivit v aplikacích a našlo efektivní prostředky pro jeho modelování a implementaci, na druhé straně se dnes žádá relativní nezávislost zdrojů a z nich vycházejících GUI jednotek. O důvodech této snahy jsme se již zmínili. Současný stav, kdy se do popředí dostávají webové objektové aplikace, představuje objektové paradigma velmi výhodný přístup k implementaci GUI zmíněných aplikací (Page Jones, 2001). Objektové paradigma umožňuje podnikové služby relativně oddělit od s nimi svázaného GUI. Tak se podnikové služby a GUI jednotky dostávají na stejnou realizační platformu. Je to konzistentnost implementačního přístupu ke GUI a jeho zdrojům.

SOUHRN

Metoda vydělení GUI z podnikových služeb a vytvoření GUI systému má reálné opodstatnění v požadavku flexibility možných změn služeb a asociovaného GUI. Na druhé straně se ale stěžuje samotná komunikace mezi podnikovými službami a vydělenými GUI jednotkami. Obě skutečnosti předkládaný příspěvek plně potvrzuje. Výsledky z teoretické a systémové roviny byly rovněž diskutovány v rovinách modelování a implementace.

V mnoha směrech příspěvek ale nevyčerpal pro GUI systém všechny možnosti uplatnění teorie systémů. Na úrovni rovin modelování a implementace bylo ukázáno, že teoretický a systémový přístup jsou v obou rovinách akceptovatelné a zvládnutelné v objektovém paradigmatu vývoje software. Rozhodně není nevhodná poznámka o tom, že GUI je poměrně široká záležitost a jeden příspěvek nemůže diskutovat všechny známé problémy spojené s jeho vývojem.

Příspěvek je zaměřen na uplatnění systémového pohledu na vazby podnikových služeb a s nimi asociovaného GUI. Tato asociace je specifikována jako jedna z relací v GUI systému. Obvykle je GUI podnikové služby považováno za její přirozenou součást. Příspěvek se, oproti této myšlence, pokouší vydělit jednotlivé celky GUI z daných služeb a vytvořit z nich samostatné GUI jednotky, prvky GUI systému. Podstatou příspěvku je ukázat, kam může vést důsledné uplatnění teorie systémů. Systémová analýza vedla ke specifikaci samotných GUI jednotek a povahy vazeb mezi GUI jednotkami a GUI systémem a podnikovými službami z jeho okolí. Byla navržena množina relací, které reprezentují v praktické práci s GUI často se vyskytující souvislosti. Pro reprezentaci vazeb v GUI systému byl navržen relační graf, který by měl dát vývojáři-programátorovi potřebné informace o jejich povaze. Tento graf nahrazuje často velmi obsírnou verbální deskripci zmíněných vazeb.

GUI jednotka, GUI systém, GUI zdroje, modelovací techniky pro GUI, funkcionalita GUI, vazby mezi jednotkami GUI, vývojové techniky pro GUI

SUMMARY

The methods for GUI separation from enterprise services and System GUI creation have foundation concerning requirement flexibility of possible service changes and associated GUI. On the other hand, the communication between enterprise services and separated GUI units is harder. The submitted article fully confirms these possibilities. The results from theoretical level were also discussed in modeling and implementation levels.

The article has not exhausted all possibilities for system theory implementation over GUI in all directions.

On the level of modeling and implementation there were showed that theoretical and system approach is acceptable in object paradigm for software development. Anyway, the GUI area is too large and one article cannot discuss all problems connected with its development.

This article is oriented to a system approach implementation on enterprise services and with them associated GUI. This association is specified as a one relation in the GUI system. Usually the enterprise service GUI unit is regarded as a natural component of a mentioned enterprise service. Against this

idea, the article tries to separate GUI parts from their enterprise services and construct from them GUI units and a GUI system altogether.

The main purpose of the article leads to showing what we can achieve by thorough system theory implementation. The system analysis leads to GUI unit specifications and nature of relations between GUI units and enterprise services from GUI system environment. Some set of relations were suggested that represent in practice very often existing associations between GUI units. Some relation graph was introduced for representation of GUI unit relations that should provide to developers-programmers all needed information about them. This graph can replace very often used verbal description of mentioned relations.

LITERATURA

PAGE JONES, M., 2001: Základy objektově orientovaného návrhu v UML. Praha, Grada Publishing. ISBN 80-247-0210-X.

ARLOW, J., NEUSTADT, I., 2003: UML a unifikovaný proces vývoje aplikací. Praha, Česká republika, Computer Press. ISBN 978-80-251-1503-9.

TIDWELL, J., 2005: Designing Interfaces. Patterns for Effective Interaction Design. Tokyo: O'REILLY. ISBN 0-596-00803-1.

MIŠOVIČ, M., 2007: Podnikový informační systém a GUI jeho aplikačního software. In: Sborník prací z mezinárodní vědecké konference Agrární perspektivy XVI. Praha, Česká zemědělská univerzita v Praze, str. 781–792. ISBN 80-213-1531-8.

Adresa

prof. RNDr. Milan Mišovič, CSc., Vysoká škola polytechnická v Jihlavě, Tolstého 16, 586 01 Jihlava, Ústav informatiky, Mendelova zemědělská a lesnická univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika, e-mail: misovic@pef.mendelu.cz