DOBYVATEĽ MENOM UML

Kompromisom dosiahneš najviac.

Michal Cádrik

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
cadrikm[zavináč]gmail[.]com

Abstrakt. *Dokumentácia sa nachádza všade okolo nás a v každom procese ľudskej činnosti. Inak tomu nie je ani v procese tvorby softvéru. Je nevyhnutnosťou, ktorú si proces žiada vďaka svojej komplexnosti. Písaná dokumentácia bola v priebehu doplnená o grafické prvky v podobe UML diagramov, ktoré sú názornejšie a lepšie čitateľné. V budúcnosti by mohlo UML úplne nahradiť písanú dokumentáciu. Či je možné, aby sa tak stalo je otázne a diskutujeme o tom v tejto eseji. Silným argumentom pre UML je tiež jeho vykonávateľná súčasť xUML, ktorá tvorí základ pre modelom riadený vývoj a výrazne uľahčuje prácu vývojárom, keďže ten rozmýšľa len nad procesom vykonania a nie nad implementačnými detailami.*

Kľúčové slová: *UML diagram, OCL, xUML, MDD*

Úvod

Každý z nás to pozná. Dlhé čakanie v radoch na úradoch len preto, že si ide zobrať nejaké potvrdenie, dokument. Strašná byrokracia, ktorá vládne celému svetu a prináša prácu mnohým nezamestnaným. Nielen na úradoch, ale v každom odvetví, odbore, dokonca aj v softvérovom inžinierstve sa produkujú tony dokumentácie. V roku 1997 sa stal štandardom modelovací jazyk UML vďaka organizácií OMG (Object Management Group). Odvtedy sa začal používať vo všetkých projektových dokumentáciách. Je však možné, aby UML jazyk nahradil klasickú písanú dokumentáciu natoľko, že bude dokumentácia pozostávať iba z týchto diagramov? Je možné špecifikovať celý systém iba pomocou UML diagramov? Je to efektívne zapísaný typ dokumentácie? Dá sa z UML rýchlo všetko pochopiť? Aká je budúcnosť UML?

Načo nám je kopa slov, keď to nikto nečíta?

Každý si položme otázku: ku koľkým programom, ktoré používam som prečítal dokumentáciu? Určite to bude veľmi zanedbateľné percento. Ale koniec koncov je to dobre. Správny program má byť navrhnutý tak, aby sa používal intuitívne, aby sa nemusela čítať dokumentácia. To je používateľská stránka veci a používateľská dokumentácia, ale pri vývoji softvéru nejde len o používateľov. Ako je to s ľuďmi, ktorí softvér vyvíjajú alebo udržujú?

Dokumenty tvoria neodmysliteľnú zložku pri vývoji softvéru. Pre manažment tvoria doklad voči zákazníkovi o dohodnutých podmienkach a rozsahu projektu a pre vývojárov (analytikov, programátorov, …) tvoria základný tok informácií pre správnu koordináciu procesu tvorby a pre údržbu systému. Z hľadiska vývoja a údržby softvéru je dokumentácia veľmi potrebná.

Písanie dokumentácie - jediné riešenie?

Nie. Existujú dva druhy dokumentácie, čo sa týka druhu prenosu informácie. Prvým je textová dokumentácia a druhým grafická (UML diagramy). Môžeme teda dokumentáciu len „nakresliť“?

Všade je len text

Všetci dobre vieme, že pomocou textu sa dá vyjadriť všetko, lebo všetko vo svete má svoje pomenovanie. Dokonca aj viacznačné pojmy je možné bližšie špecifikovať ďalšími slovami.

Teda výhodou písanej dokumentácie je jej jednoznačnosť a široká škála použitia. Textom opíšeme všetky aspekty systému, jeho ohraničenia a špeciálne detaily systému.

Avšak textová dokumentácia má svoje nevýhody. Jednoduchú vec je potrebné opísať na mnoho riadkov, vzniká nám veľké množstvo textu, v ktorom sú nahustené všetky informácie a v konečnom dôsledku sa takýto dokument stáva neprehľadný. V dnešnej dobe sa softvérové firmy spájajú nielen v rámci jedného štátu, ale dokonca interkontinentálne. Tento trend spôsobuje problém v oblasti komunikácie, keďže sa stretávajú rôzne kultúry a vznikajú jazykové bariéry.

Z pohľadu softvérového vývojára je ťažšie čítať špecifikáciu v jazyku, ktorý nie je jeho materinským, a tým sa znižuje jeho efektivita práce. Nehovoriac o množstve textu, ktorý by musel prečítať, kým by sa dozvedel všetky potrebné informácie.

Textová dokumentácia určite nie je postačujúca pre dnešnú dobu z hľadiska efektívnosti, keďže softvér je čoraz komplexnejší a zložitejší, tak aj slov je viac a viac a je obtiažne sa vyznať v takom množstve textu aj napriek dobrému štruktúrovaniu.

Cesta k UML

Avšak písanie textu nie je jediný spôsob ako predať informáciu ďalej. Ako operačné systémy prechádzajú svojim vývojom a postupne sa stávajú viac používateľsky prívetivými vďaka prepracovanejšiemu grafickému používateľskému prostrediu, tak aj dokumentácia sa stala lepšie čitateľnou vďaka grafickým prvkom.

Preto v minulosti vznikali dokumentácie obohatené o grafické reprezentácie systému a jeho vlastností. Problémom ostal význam daných obrázkov, keďže každá firma alebo

vývojár mal vlastný spôsob zakreslenia tej istej vlastnosti systému. Zmena nastala až po štandardizácii UML v roku 1997. Pre dokumentovanie a dokonca aj vývoj systému to bol veľký krok vpred.

O slovo sa hlási UML

V súčasnosti je dostupný štandard UML 2.0, ktorý definuje rôzne typy diagramov potrebných pre tvorbu softvéru. Pomocou týchto diagramov sa dajú namodelovať rôzne aspekty systému, ktoré spolu tvoria obraz celého systému.

Z vlastnej skúsenosti viem, že je jednoduchšie sa pozrieť na diagram, v ktorom je väčšina vlastností (alebo všetky vlastnosti) danej funkcionality vykreslená, ako prečítať stranu textu ohľadom danej funkcionality.

Výhody UML sú jednoznačné už z jeho podstaty. Sú to diagramy, grafické reprezentácie celého systému alebo jeho jednotlivých funkcionalít. Pre každého čitateľa vie rovnaký diagram ponúknuť inú informačnú hodnotu. Softvérový architekt v diagrame vidí niečo iné, ako programátor, ktorý má niečo naprogramovať. Táto vlastnosť je veľmi výhodná, lebo nepotrebujeme pre každého zvlášť písať množstvo textu ako pri textovej dokumentácií.

Nespornou výhodou je taktiež schopnosť komunikovať pomocou diagramov internacionálne, keďže je to stanovený štandard, ktorý sa celosvetovo používa. Preto jazyk nehrá až takú veľkú úlohu ako v textovom type dokumentácie aj keď sa popisy v diagramoch nachádzajú. UML podporuje distribuovaný vývoj aplikácií v multikultúrnych prostrediach.

UML je rozšíriteľný jazyk. Táto vlastnosť je jeho silnou stránkou, pretože môžeme namodelovať ľubovoľnú vlastnosť systému alebo funkcionalitu. Vzniká tu problém, ktorý opisuje aj štúdia zo zdroja [3] a to mnohoznačnosť UML diagramov.

Je naozaj UML nejednoznačný jazyk?

V [3] bol opísaný kompletný proces vytvorenia štúdie, ktorá sa zameriavala na časť údržby softvéru a vykonanie zmien. Ak však chceme porozumieť celému systému korektne, tak je potrebné aj zodpovedajúce množstvo diagramov. Teda, na vykonanie potrebnej zmeny v systéme poskytol autor respondentom štúdie diagramy týkajúce sa danej funkcionality systému, ale neposkytol im celkový počet diagramov, ktoré by mohli viesť respondentov k správnym záverom.

Ďalší problém môže byť skutočnosť, že dokumentácia k systému nebola tvorená so zámerom vytvoriť vývojársku dokumentáciu založenú len na UML. V závere tvrdí, že UML nie je postačujúce preto, lebo je potrebná znalosť domény systému. Avšak v štúdii sa neuvádza, či bol popri UML použitý aj jazyk OCL (súčasťou UML podľa [2]), ktorý spresňuje význam UML diagramov tým, že definuje vlastnosti jednotlivých objektov v systéme, podmienok prechodov stavov, atď.

V realite však pri údržbe softvéru máte celkovú dokumentáciu, v ktorej sa vyskytujú všetky diagramy systému pokiaľ sa systém počas vývoja dokumentoval hlavne UML diagramami. Z týchto diagramov by sa doména systému mala dať určiť, pretože v nich musí byť popísaná funkcionalita každej časti systému. Prípadné vlastnosti systému po zmene sú definované používateľom a od neho vieme zistiť ostatné podrobnosti, keďže

zmenu robíme kvôli nejakému účelu. Takisto pomocou OCL súčasti jazyka UML je možné zachytiť aj doménové vlastnosti systému v diagramoch UML. Z daných poznatkov vyplýva, že UML jazyk je jednoznačný, ak sa použijú všetky jeho súčasti.

Postačuje nám UML na pochopenie systému?

Z vlastnej skúsenosti viem, že pri vývoji softvéru sú pre programátora UML diagramy s použitím OCL postačujúce. Ku každej funkcionalite existujú viaceré diagramy, ktoré spolu dotvoria celkový obraz o danej funkcionalite. Vďaka tomu výsledok štúdie uvedenej v zdroji [3] nemusí byť úplne správny.

Efektívnosť dokumentácie

Koniec koncov nie je dôležité, či použijeme jeden typ dokumentácie alebo druhý typ dokumentácie na vyjadrenie informácií. Ako sme zistili, tak oba typy sú na dokumentovanie systému vhodné. Vyššie sme uviedli jednotlivé výhody a nevýhody daných spôsobov.

Dôležitejším aspektom dokumentácie je teda v jej efektívnej pomoci vývojárovi pochopiť stavbu a funkčnosť systému, aby mohol danú funkcionalitu čo najrýchlejšie implementovať a pri údržbe mohol čo najlepšie posúdiť, ktorá zmena by bola pre chod systému najlepšia. Efektivita tvorí základ úspešnej firemnej stratégie ako aj úspešnosti projektov kvôli minimalizácií nákladov na vývoj softvéru.

Textová forma alebo grafická forma?

Čisto písaná forma dokumentácie určite nie je najefektívnejší spôsob dokumentácie softvérového produktu už len kvôli celkovej prehľadnosti. Na druhej strane ani systém popísaný množstvom diagramov nie je úplne najlepšia voľba, lebo vývojár musí preštudovať veľa diagramov a nemusí si pri tom všimnúť určité obmedzenia alebo špecifické vlastnosti systému.

Kombinácia prvých dvoch prístupov

Písaná forma dokumentácie určite nevymizne z procesu vývoja a údržby softvérového produktu, lebo je to univerzálna forma komunikácie v každej oblasti života. Určite je efektívnejšie, keď je každý diagram popísaný krátkym stručným textom, v ktorom sú uvedené iba špecifické časti systému alebo iné špecifiká, ktoré priamo nie sú viditeľné z UML diagramu alebo by ich bolo potrebné hľadať v iných diagramoch.

Takisto je to pre vývojára pohodlnejšie si hneď pozrieť krátky popis než študovať každý jeden diagram, aby pochopil malú časť systému, ktorú treba naprogramovať v prípade vývoja alebo zmeniť v prípade údržby softvéru.

Ako v každej oblasti ľudského života je potrebné robiť kompromisy, aj v oblasti dokumentácie je najvhodnejší. Svedčí o tom reálne používanie v praxi. Napriek tomu, že samotní vývojári nepotrebujú písaný text k vysvetleniu diagramov, predsa len je to efektívnejší spôsob tvorby dokumentácie vzhľadom na jej budúce použitie pri údržbe systému.

Budúcnosť je v UML

Vďaka neustálemu trendu zvyšovania komplexnosti a zložitosti systémov sa do popredia stále viac a viac dostáva modelom riadený vývoj. Nemalú zásluhu na tom má UML a jeho vykonávateľná časť xUML (executable UML), ktorý celý proces ešte zjednodušuje. Z xUML je možné vygenerovať vykonávateľný kód priamo podľa navrhnutého modelu. Veľkou výhodou je, že čo zmeníte podľa špecifikácie v modeli, tak sa to priamo zmení aj v kóde a nemusíte sa báť, že budete mať nekonzistentné tieto časti.

Najväčšou výhodou je, že popri tvorbe dokumentácie už máte hotový prototyp a viete si otestovať, či jednotlivé prvky systému fungujú správne. Všetky tieto výhody vplývajú pozitívne na efektivitu vývoja aplikácií. Samozrejme existuje množstvo CASE nástrojov na podporu vývoja riadeného modelom. Trendy smerujú k tomu, že programátor nebude musieť vedieť žiadny programovací jazyk, ale len UML, ktorým si potrebnú funkcionalitu namodeluje a generátory spravia potrebné transformácie za neho. Takisto je to aj bezpečnejšie, lebo programátori sú len ľudia a ošetrenie všetkých chýb nie je niekedy reálne (podľa prof. Ing. M. Bielikovej, PhD. neexistuje bezchybný softvér).

Ako každý istotne postrehol, v modelom riadenom vývoji je dokumentácia vo forme UML diagramov to jediné čo treba vykonať (ak máte správne CASE nástroje) a máte hotovú aplikáciu. Z tohto vyplýva, že aj údržba systému je rovnako pomerne jednoduchá (upravenie časti diagramu). Viac o výhodách MDD môžete nájsť v zdroji [1].

Záver

UML je silný jazyk, ktorý veľmi zefektívňuje proces tvorby softvéru tým, že sprehľadňuje dokumentáciu, čo takisto prispieva k efektívnejšej údržbe systému. Textová aj grafická forma dokumentácie pomocou UML diagramov je postačujúca a jednoznačná pre dokumentovanie celého systému, ale z pohľadu efektívnosti, ktorú má dokumentácia priniesť pre vývoj a údržbu systému nie je najefektívnejším riešením. UML veľmi podporuje vývoj riadený modelom, ktorý má množstvo výhod v dobe komplexných a distribuovane vyvíjaných systémov. V tomto spôsobe vývoja softvéru je základom model, ktorý je zároveň dokumentáciou a môže byť prekonvertovaný na základ aplikácie pre jej ďalší vývoj pomocou CASE nástrojov, alebo už finálnu aplikáciu. Vďaka tejto vlastnosti UML jazyka je podľa mňa možné, že za pár dekád rokov si dobyvateľ UML podrobí celý svet softvérového vývoja.

Použitá literatúra

1. Azoff, M.: *The Benefits of Model Driven Development*, Online: http://www.google.sk/url?sa=t&rct=j&q=pros%20of%20model%20driven%20development&source=web&cd=1&ved=0CCwQFjAA&url=http%3A%2F%2Fwww.ca.com%2F~%2Fmedia%2FFiles%2Fwhitepapers%2Fthe-benefits-of-model-driven-development.pdf&ei=J-THTuPtKpPZ4QTX_NRE&usg=AFQjCNG_5b-dlJfF8PI6aRZw99PU_i72kA&sig2=qWs49tMkUxct3csU3_-5DQ&cad=rja, Marec 2008, posledný prístup 19. 11. 2011

2. OMG: *Object Constraint Language,* Online: http://www.omg.org/spec/OCL/2.0/PDF/06-05-01.pdf, Máj 2006, posledný prístup 19. 11. 2011
3. Tilley, S., Huang, S.: *Qualitative Assessment of the Efficacy of UML Diagrams as a Form of Graphical Documentation in Aiding Program Understanding,* Proceedings. of the 21st annual international conference on Documentation. (SIGSOC '03), pp 184–191. ACM Press: New York, NY, 2003.

Annotation

UML the conqueror

Documentation can be found everywhere around us and in every process of human activity. The process of software making is not an exception. It is a necessity which is very important for the process because of its complexity Writed type of documentation was enriched by graphical elements in the way of UML diagrams which are more suitable for reading and understanding of program's processes. UML could be a replacement for the textual documentation. If it is possible is the point of this essay. The one pro is part of UML called executable UML (xUML), which is the basis of model driven architecture, is very strong argument for graphical documentation, because developer does not need think about implementation details of specific language.