VÝZNAM PROSTŘEDKŮ CASE PRO ŘÍZENÍ JAKOSTI SOFTWARE

Dujka Jan

Doktorand ÚAI FSI VUT v Brně

1. Úvod

Lze konstatovat, že množství a kvalita testů prováděných při využívání CASE (Computer Aided Software Engineering) produktů, ať již na pozadí, či v přímé komunikaci s uživatelem, jsou určujícím faktorem chybovosti vyvíjeného software.

Jakost software představuje v současnosti velmi aktuální problém, jak se zvyšuje všeobecně tlak na jakost a roste počet programových produktů. [5] Můžeme určit následující skutečnosti, které jsou příčinou této zvýšené pozornosti, která je věnována jakosti software:

- Ochrana investic do SW, protože objem vložených finančních prostředků představuje celosvětově nesmírně vysoké částky
- SW se stal spotřebním zbožím a zákazníci chtějí být chráněni před nekvalitním SW
- Zvýšení bezpečnosti aplikací v oblasti řízení , kde chyba v SW může mít katastrofální následky (řídicí systémy jaderných elektráren, automatické řízení letadel a aut, programové řízení funkcí lékařských přístrojů, apod.)

Pro dosažení vysoké jakost software je nutno věnovat pozornost nejen správnému postupu tvorby programů, ale i používaným prostředkům, které podporují návrh softwarových produktů. Produkty CASE patří mezi základní prostředky, které umožňují významným způsobem snížit počet chyb při vývoji softwarových produktů.

2. Výhody produktů CASE

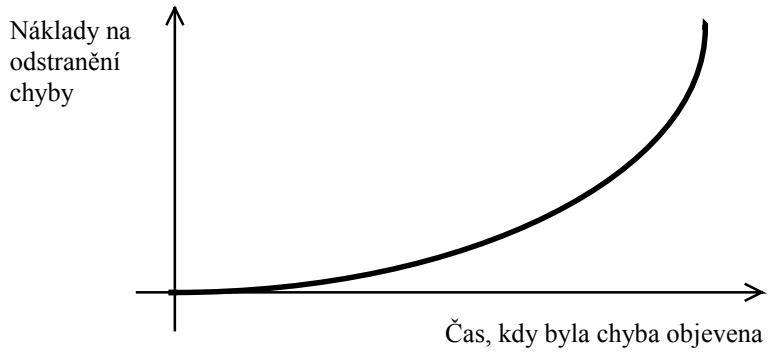
Připomeňme čtyři hlavní důvody užívání CASE produktů v jednotlivých fázích životního cyklu software.

Prvním důvodem je očekávání zkrácení doby jednotlivých etap životního cyklu, v nichž je daný CASE prostředek nasazen. Toto je dáno především tím, že CASE nástroj umožňuje prostřednictvím grafiky (různé typy diagramů) výrazně zlepšit komunikaci jak mezi vývojáři, tak i mezi vývojáři a budoucím uživatelem. Je naprosto zřejmé, že vypovídací schopnost diagramu je daleko dokonalejší než verbální popis. Navíc doba nakreslení diagramu je, ve srovnání s potřebou stejný problém popsat verbálně, výrazně nižší. Na zkracování vývoje aplikací se výraznou měrou podílí možnosti, které poskytují CASE produkty v oblasti generování prováděcích kódů vyvíjené aplikace.

Druhým důvodem používání CASE prostředků je pevná podpora vybrané metody. Toto je však pro softwarehouse velmi zavazující, pokud nechtějí i nadále vyvíjet ad-hoc a potřebují respektovat zásady norem ISO 9000:2000. Nezbývá, než se upnout buď na CASE nástroj (tento zakoupit a přizpůsobit se mu výběrem jím podporované metody) nebo se upnout na metodu (a následně zakoupit CASE prostředek, který ji podporuje).

Třetím důvodem je možnost využití dokumentačních nástrojů zvoleného CASE prostředku. Většina současných strukturovaných i objektově orientovaných CASE prostředků má v sobě integrované prostředky pro tvorbu všech typů dokumentace vznikající v různých fázích životního cyklu. Nejedná se zpravidla jen o pouhý tisk diagramů vyskytujících se na obrazovce, nýbrž o dokonale hierarchicky zpracovanou dokumentaci ve formě přehledových sestav, detailních popisů entit, atributů, relačních a dědičných vztahů a toto vše teprve doplněno jednotlivými diagramy. Jak již řekl Galileo Galilei: „O předmětu nic nevíme, pokud ho nejsme schopni doložit čísly nebo parametry“. Dokumentační nástroje CASE prostředků nám mohou proto být významným pomocníkem při dokumentování systému z důvodu certifikace ISO 9000.

Čtvrtým důvodem je požadavek na výrazné snížení chybovosti systému. Ze známé křivky si lze snadno uvědomit,



že čím dříve je v životním cyklu software detekována chyba, tím menší náklady jsou pak potřebné na její odstranění.

Výstupy z jedné etapy životního cyklu jsou zpravidla vstupem do etapy následující. V každé etapě životního cyklu vznikají chyby a našim cílem je zachytit je již v etapě, ve které vznikly, odstranit je a připustit jen minimální přenos chyb z jedné etapy do druhé.

Jak je možno tohoto cíle dosáhnout?

Předně důsledným dodržováním osvědčených metod vývoje software a jejich počítačovou podporou realizovanou prostřednictvím CASE.

CASE nástroje obsahují v sobě celou řadu integrovaných testů, kterými zaručují jednotlivým řešitelům více či méně pevné „mantinely“, které umožňují neodchýlit se od dané metodiky. Co se týče vztahu výše uvedených integrovaných testů a ceny vlastního CASE nástroje, platí známé rčení: „Za málo peněz, málo muziky“. Navíc lze konstatovat, že řada firem, snažících se dodat na trh integrovaný CASE produkt, opomíjí poněkud nezbytnost a striktnost integrovaných testů a tyto CASE nástroje se stávají tímto k vývojáři poněkud benevolentnější. Mám zde na mysli CASE nástroje, podporující jak datové, tak i funkční modelování, správu komplexní dokumentace, to vše s podporou řízení projektů.

Je na možno se domnívat, a praxe to dokazuje, že právě tyto nástroje nevedou k výraznému snížení chybovosti, jejich použití ke generování kódu ustupuje spíše do pozadí a využívají se pak jen k prezentačním a dokumentačním účelům.

Např. ve fázi analýzy je požadováno v praxi napřed sestrojení konceptuálního modelu (bez jakéhokoli zohlednění vlivu budoucí implementace), z konceptuálního modelu se následně vygeneruje fyzický model, který už v sobě zahrnuje implementační závislosti a z fyzického modelu se vygeneruje fyzická databáze. Aby takto vzniklá fyzická databáze měla praktickou využitelnost a aby byla jakostní, je nutné před každým generováním, provést řadu testů zajišťujících referenční integritu a bezpečný přechod do další fáze a případě detekovaných chyb je nutné je odstranit.

3. Hlášení nalezených chyb

Pro detailnější představu v následujícím příkladě výpis chybových hlášení po uskutečněném testu konzistence systému v case/4/0. Jsou-li detekovány chyby s ohledem na správu indexů a primárních klíčů, může být korespondující systém následně optimalizován a reorganizován a takto opět stabilizována integrita systému pro pozdější použití. Zobrazení chybových hlášení je na obr. 1.

Pro ukázku, jaké testy probíhají nad modelem vytvořeným v objektově orientovaném prosuktu POWER DESIGNERU uvádím následující výpis na obr.2.
Uveďme ukázku jak se generuje z konceptuálního modelu model fyzický. Ukázka je demonstrována v POWER DESIGNERU. Jsou zavedeny zkratky:
CDM ..... konceptuální model
PDM ..... fyzický model
Jen jako poznámku je nutno uvést, že cizí klíče (FK) při generování automaticky migrují z entity na straně 1 vztahu, kde jsou uvedeny jako klíče primární (PK), do entit na straně N. Vztah M:N v konceptuálním modelu je při generování automaticky nahrazen dvěma vazbami 1:N a jednou vazební entitou. Na závěr je nově vzniklý fyzický model ještě jednou otestován, jestli byly skutečně dodrženy pravidla referenční integrity. Viz obr.3.

4. Závěr

Z uvedených příkladů by mělo být patrné, že produkty CASE význačnou měrou umožňují zkvalitnit tvorbu software prostřednictvím zabudovaných automatických kontrol. Ve spojení s propracovanými metodami analýzy a návrhu informačních systémů [4] vytvářejí podmínky pro dosažení vysoké jakosti při tvorbě programů ve smyslu souboru norem ISO 9000:2000. Softwarové firmy, které chtějí získat certifikát jakosti podle těchto norem, by měly začlenit určitý produkt CASE mezi svoje pracovní nástroje a některou standardizovanou metodu tvorby software používat jako svůj základní pracovní postup.

Tři naše firmy, které se dodávají na náš trh produkty CASE (LBMS Praha. KOMIX Praha a UNICORN Praha) však jistě potvrdí, že poptávka po produktech CASE v ČR silně zaostává za jak za potřebou, tak za průměrem prodejnosti těchto produktů západních zemí stejně jako poptávka po produktech, které podporují provádění testů.

Literatura:

1. Smith,J.D.-Wood,B.,K.: Engineering quality sofrware. Elsevier Applied Science, London 1989 (Second Edition)
2. Software Cerification (ed. Neumann B.) . Elsevier Applied Science, London 1988
3. Patton,R.: Testování softwaru. Computer Press Praha 2002

4. Lacko,B.: Standardizované metody analýzy a návrhu informačních systémů, Softwarové noviny, roč.IX., (1998), č.1., str. 36- 38
5. Lacko,B.: Jakost a informační systémy. Sborník 6. mezinárodní konference JAKOST 97, Dům techniky Ostrava 1997, s.339-349

Obr.1.

case/4/0 Test report

Pfad:	E:\CASE41W\USERDAT2
System:	AZV_DOPR
Version:	4.x
Status:	The System is ok.

Meldungen:

ATTRIBUTE	Table is empty
BLOCK	Pkey Ok Index Ok Fkey Ok Text Ok
BLOCKPAGE	Table is empty
CALLBACK	Table is empty
CALLBACKTYPE	Pkey Ok Index Ok Fkey Ok Text Ok
COMBINATIONPART	Table is empty
COMPONENT	Table is empty
COND	Table is empty
CONDITIONPART	Table is empty
CONDITIONTERM	Table is empty
CONNECTIONPOINT	Pkey Ok Index Ok Fkey Ok Text Ok
CONNECTIONPOINTSET	Pkey Ok Index Ok Fkey Ok Text Ok
CONTROLACTION	Table is empty
CREATIONACTION	Table is empty
CURSOR	Table is empty
CURSORRELATION	Table is empty
CURSORVIEW	Table is empty
DATA	Table is empty
DATABASE	Table is empty
DATABASEFILE	Table is empty
DATABASERELATION	Table is empty
DATABASEVIEW	Table is empty
DATASTRUCTURE	Pkey Ok Index Ok Fkey Ok Text Ok
DATATYPE	Pkey Ok Index Ok Fkey Ok Text Ok
DATAVALUE	Table is empty
RESOURCEVALUE	Table is empty
RNODE	Table is empty
RSHIP	Table is empty
SCENARIO	Table is empty
USAGEDECL	Table is empty
USAGEFUNC	Table is empty

USAGERELATION	Table is empty
USERACTION	Table is empty
WIDGET	Table is empty
WIDGETATTRIBUTE	Table is empty
WIDGETPROPERTY	Table is empty
WIDGETVALUE	Table is empty

SYSTEM

Primary Key	Ok
Diagram Integrity	Ok

FUNCTION STRUCTURES

Element Position	Ok
Element Link	Ok
Page Reference	Ok
Function Integrity	Ok
Part Integrity	Ok
Recurrent Structure Integrity	Ok
Refinements	Ok

INFORMATION FLOWS

Element Position	Ok
Number of Elements	Ok
Element Link	Ok
Function Integrity	Ok
Port Integrity	Ok
External Interface Integrity	Ok

DATA STRUCTURES

Element Position	Ok
Element Link	Ok
Page Reference	Ok
Type Structure Reference	Ok

ER DIAGRAMS

Element Position	Ok
Number of Elements	Ok
Element Link	Ok
Entity Type Integrity	Ok
Attribute Integrity	Ok

TYPE STRUCTURES

Element Position	Ok
Element Link	Ok
Page Reference	Ok
Technical Name Integrity	Ok

RELATIONAL MODEL

Element Position	Ok
Number of Elements	Ok

Element Link	Ok
Relation Integrity	Ok
Attribute Integrity	Ok

MODULE STRUCTURES

Element Position	Ok
Element Link	Ok
Page Reference	Ok
Technical Name Integrity	Ok
Template Element Usage	Ok
Nested Program Usage	Ok

IMPLEMENTATION TREES

Element Position	Ok
Element Link	Ok
Page Reference	Ok

REPORT STRUCTURES

Element Position	Ok
Element Link	Ok
Page Reference	Ok

Obr.2

Checking the model "Sprava systemu" (SPRAVA_SYSTEMU)
Modification date: 20.01.2003 12:17

Checking Data Items...
Warning: The following data items are not used:
-> Data Item "VykId" (VYKID)
-> Data Item "VykNazev" (VYKNAZEV)
-> Data Item "VykNH" (VYKNH)
-> Data Item "VykPop" (VYKPOP)
-> Data Item "RC" (RC)
-> Data Item "PracId" (PRACID)
-> Data Item "EmiId" (EMIID)
-> Data Item "EmiNazev" (EMINAZEV)
-> Data Item "VecId" (VECID)
-> Data Item "VecNazev" (VECNAZEV)
-> Data Item "DanId" (DANID)
-> Data Item "DanNazev" (DANNAZEV)
-> Data Item "DanSazba" (DANSAZBA)
-> Data Item "DovId" (DOVID)
-> Data Item "DovNazev" (DOVNAZEV)
-> Data Item "DovPopis" (DOVPOPIS)

Checking Entities...
Warning: The following entities do not have a relationship:

-> Entity "S_ChybyDB" (S_CHYBYDB)
-> Entity "S_Poznamky" (S_POZNAMKY)
-> Entity "Dočasné tabulky" (S_TEMPTABLE)

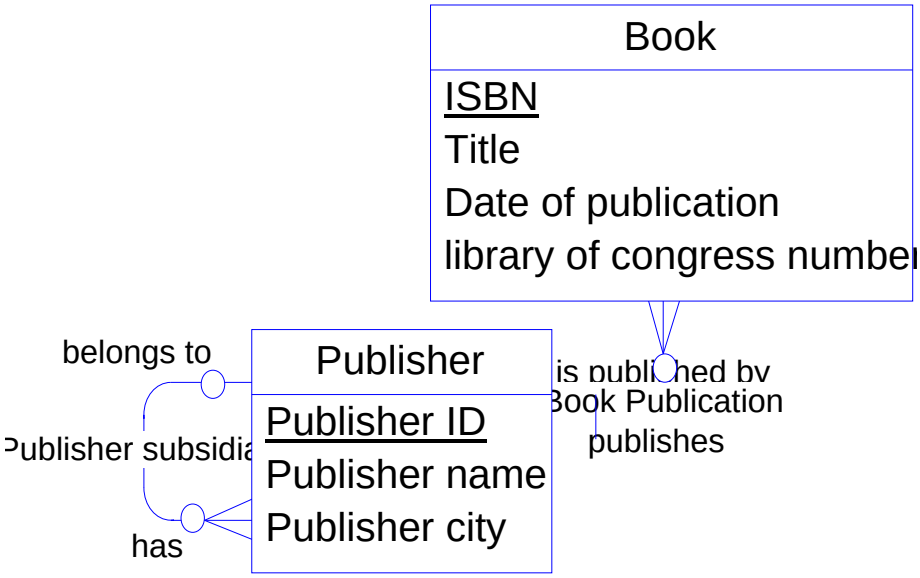
Checking Relationships...

Checking Inheritances...

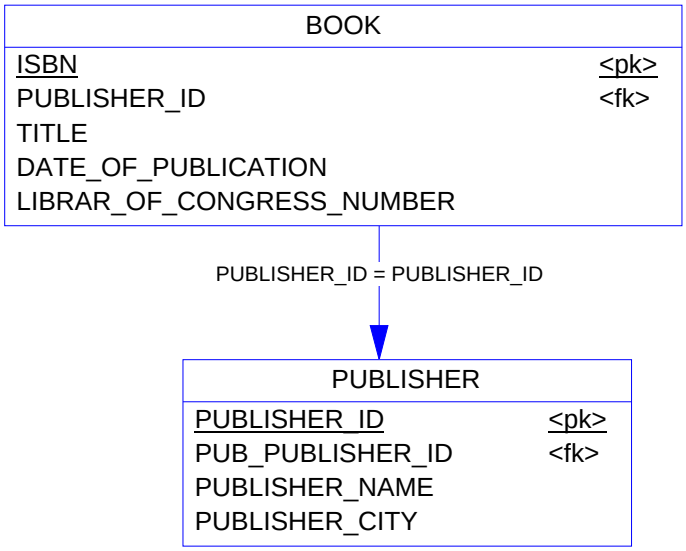
Result: 0 error(s), 19 warning(s).
The model is correct, no errors were found.

Obr.3

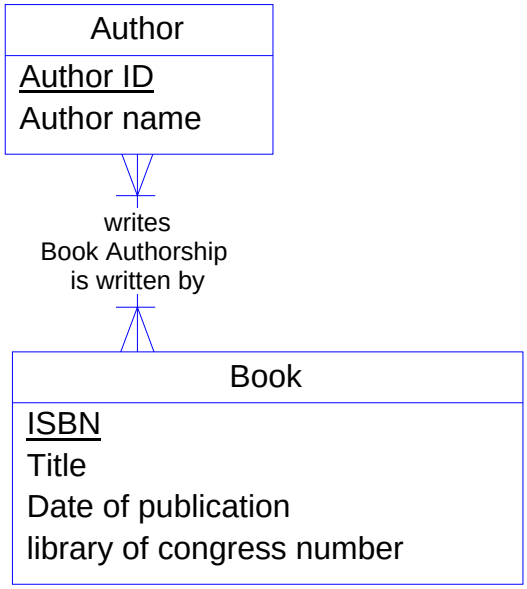
CDM pro vazbu entit Book a Publisher



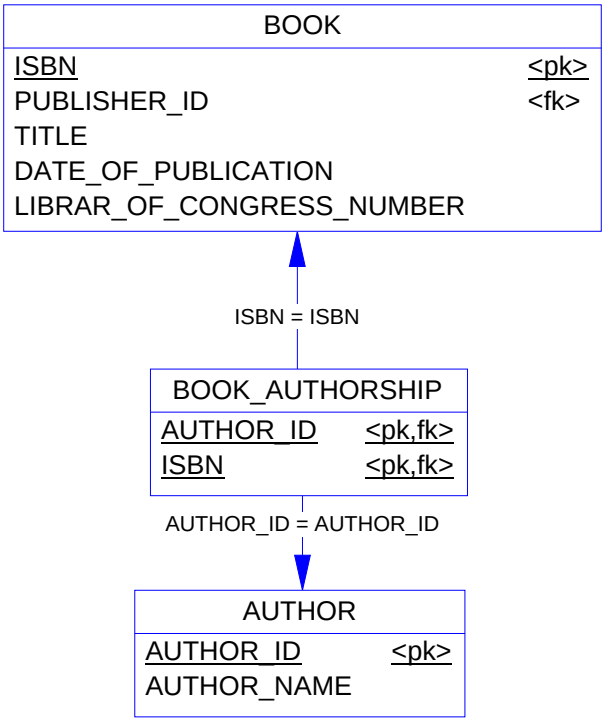
PDM pro vazbu entit Book a Publisher


BOOK
ISBN <pk>
PUBLISHER_ID <fk>
TITLE
DATE_OF_PUBLICATION
LIBRAR_OF_CONGRESS_NUMBER
PUBLISHER_ID = PUBLISHER_ID
PUBLISHER
PUBLISHER_ID <pk>
PUB_PUBLISHER_ID <fk>
PUBLISHER_NAME
PUBLISHER_CITY

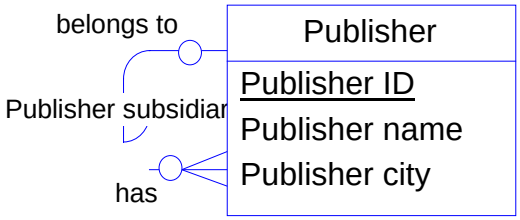
CDM pro vazbu entit Autor a Book



PDM pro vazbu entit Autor a Book



CDM pro vazbu entity Publisher



PDM pro vazbu entity Publisher