Typologie uživatelského rozhraní

Martin Molhanec

ČVUT-FEL Praha, Katedra elektrotechnologie,Technická 2,166 27 PRAHA 6, Česká Republika

Abstrakt
Tématem tohoto příspěvku jsou úvahy o souvislostech mezi elementy grafického uživatelského rozhraní a jim odpovídajícím databázovým atributům. Souvislosti jsou osvětleny na několika jednoduchých příkladech.

Úvod

Problematika uživatelského rozhraní (GUI-Graphics User Interface) je jedním z aspektů, které v současné době rozhodují o komerční úspěšnosti informačních systémů. Prakticky standardem pro úspěch aplikace se stalo využití standardního uživatelského rozhraní typu Microsoft Windows. Přesto je problematice uživatelského rozhraní věnováno daleko méně odborných článků než například problematice datové či funkční analýzy. Přesto je nepochybné, že i návrh kvalitního uživatelského rozhraní si vyžaduje jeho analýzu, ale jaké jsou prostředky této analýzy?

Tento článek je nevelký příspěvek v oblasti analýzy uživatelského rozhraní. Jeho cílem je podat určitý pohled na základní vztahy mezi objekty datové analýzy a objekty uživatelského rozhraní.

Základní vztah mezi objektem GUI a ERD

Upřeme pozornost na vztah mezi jednoduchým prvkem GUI a objektem datového modelu - entitou. Tento vztah je znázorněn na obrázku č. 1. Jednoduchým prvkem GUI (Simple Field) je rozuměn grafický element GUI určený pro vstup nebo výstup jedné atomické hodnoty. Jeho obrazem v datovém modelu je atribut entity. Je zřejmé, že není žádných zábran, aby programátor realizoval situaci znázorněnou na obrázku č.1. Jednoduchý element A odpovídá atributu A entity E1 a jednoduchý element B odpovídá atributu B entity E1. Pro programátora nebude také žádným problémem sestrojit příslušný SQL příkaz, který oba jednoduché elementy GUI naplní hodnotami jim příslušných atributů entity E1.

Select A,B Into A:, B: From E1 Where <podmínka>;

Za předpokladu, že výsledkem podmínky jsou hodnoty atributů A a B jednoznačně určeny. Zjednodušeně řečeno; příkaz select nám vrátí jeden řádek databázové tabulky. SQL příkaz pro aktualizaci atributů A a B z jim odpovídajících jednoduchých elementů může být potom například:

Update E1 Set A = A:, B = B: Where <podmínka>;

Kde podmínka musí opět zajistit jednoznačnost atributů A a B, které budou aktualizovány. Je nutno dodat, že výše uvedené SQL příkazy nejsou jediným způsobem, jak aktualizovat atributy databáze z elementů formuláře a naopak.

Pohleďme nyní na obrázek č.2. Situace je podobná jako na obrázku č.1. Nicméně nyní jsou atributy obsaženy v různých entitách. Vztah mezi entitami je 1:1. Intuice nám říká, že opět není problém tuto situaci realizovat. Vztah mezi entitami je realizován prostřednictví primárního klíče PK, který je pro obě totožný. Uveďme si opět pro ilustraci SQL příkazy pro naplnění elementů z databáze a pro aktualizaci atributů z elementů formuláře.

Select A Into A: From E1 Where PK = <výraz>;
Select B Into B: From E2 Where PK = <výraz>;

Update E1 Set A = A: Where PK = <výraz>;
Update E2 Set B = B: Where PK = <výraz>;

Kde výraz musí opět zajistit jednoznačnost výběru hodnot atributů A a B a navíc musí mít pro oba výběry, respektive pro obě aktualizace stejnou hodnotu. Opět podotýkám, že výše uvedený způsob není ani jediný ani nejlepší z možných.

Zatím jsme se nesetkali se žádným problémem. Pokusme se situaci dále komplikovat. Obrázek č.3. ukazuje situaci, kdy je mezi entitami E1 a E2 vztah 1:M. Co se nám na naší modelové situaci změní ? Jedná se nějakou komplikaci ? Jaké problémy nám tato situace přináší ?

Představme si, že se jedná o klasický případ vztahu 1:M. Například půjde o vztah mezi *Fakturou* a její *Položkou*. Jedna *Faktura* může mít několik *Položek*, ale nejméně jednu a každá *Položka* přináleží právě jedné *Faktuře*. Pokud naší analýzu započneme od elementu B jež přináleží atributu *Položky* je zřejmé, že nám nic nebrání realizovat i element A, který přináleží atributu *Faktury*. Pro programátora není žádný problém realizovat program pro aktualizaci atributů entit z příslušných elementů GUI.

Pokusme se nyní situaci obrátit. Pokud naší úvahu započneme od elementu A, který přináleží atributu *Faktury*, zjistíme, že je problém realizovat element B jako jednoduchý element GUI. Vyjma případu, kdy má *Faktura* právě jednu *Položku*. Naopak je zcela běžným případ, kdy jedna *Faktura* má *Položek* více! Je zřejmé, že pokud oba elementy budou jednoduché elementy (jednoduchý element umožňuje zobrazit nebo aktualizovat pouze jedinou atomickou hodnotu, která odpovídá jednomu atributu datového modelu), není situace nakreslená na obrázku č.3 reálná.

Náš rozbor přinesl nový poznatek, možnost realizovat naše schéma je odvislé od elementu, na kterém naší analýzu započneme! Nazvěme si tento element *Kořenovým* elementem (Root Element). Potenciální algoritmus analýzy může být následující. Určíme *Kořenový* element, jemuž odpovídá v datovém modelu právě jedna entita. Dále najdeme všechny okolní entity, tj. entity, které mají vztah (relationship) k této entitě, z nich vybereme ty entity, které mají ke *Kořenové* entitě (entita jež přináleží *Kořenovému* elementu) vztah 1:1 nebo 1:M.

Jestliže všechny entity, které mají vztah ke *Kořenové* entitě nazveme *Okolím entity prvního řádu*, potom všechny entity které mají ke *kořenové* entitě vztah 1:1 nebo 1:M nazveme *Okolím kořenové entity prvního řádu*. Tento postup budeme opakovat. Entity, které jsou vzdáleny přes dva vztahy, budou součástí okolí druhého řádu, atd. Sjednocením *Okolí* všech řádů (*Okolí řádu 0* bude entita sama) získáme *Okolí* entity. Je nasnadě, že pokud bude datový model spojitý, budou všechny entity navzájem ve svých *okolích*. Je to logické, pokud by existovala entita Y, která nebude v *Okolí* entity X, potom nebude ani v *Okolí* žádné entity z *Okolí* entity X. Potom bude entita Y v samostatném datovém modelu, který nemá k datovému modelu, ve kterém leží entita X, žádný vztah.

Jiný je ovšem význam pojmu *Okolí kořenové* entity. Pokud entita Y neleží v *Okolí kořenové* entity X, znamená to, že atributy z této entity nelze jako jednoduché elementy umístit společně s jednoduchými elementy odpovídajícími atributům *Kořenové* entity. Čili v našem jednoduchém případě, pokud je *Kořenová* entita *Faktura*, neleží entita *Položka* v *Okolí kořenové* entity *Faktura*, pokud je *Kořenová* entita *Položka*, leží entita *Faktura* v *Okolí kořenové* entity *Položka*.

Na obrázku č.4 je naznačena situace, kdy je *Kořenovým* elementem element B, *Kořenová* entita je potom entita E2, entita E1 je v *Okolí prvního řádu Kořenové* entity E2 a element A je element entity E1 z *Okolí prvního řády Kořenové* entity E2. Šipky určují postup kterým naše analýza probíhala. Od stanovení *Kořenového* elementu ke stanovení *Kořenové* entity, ke stanovení entit *Okolí prvního řádu kořenové* entity až ke stanovení elementů z entit *Okolí prvního řádu kořenové* entity. *Kořenový* element a entita jsou od ostatních entit odlišeny.

Ukažme si pro názornost o trošku složitější případy, abychom si ujasnili výše uvedené úvahy na stále ještě poměrně jednoduchých případech. Začneme obrázkem č. 5. *Kořenový* element je element A, *Kořenová* entita je entita E1. Entita E2 je v *Okolí kořenové* entity prvního řádu a entita E3 je v *Okolí kořenové* entity druhého řádu. Může se jednat například o *Položku* faktury, *Fakturu* a *Obchodní případ*. Je zřejmé, že při této volbě *Kořenového* elementu (respektive *Kořenové* entity) je možné bez problému naznačené GUI s příslušnými elementy realizovat. Na obrázku je také naznačeno *Okolí kořenové* entity prvního a druhého řádu.

Obrázek č.6 znázorňuje situaci, kde je vztah mezi entitou E3 a E2 obrácen oproti obrázku č. 5. Je zřejmé, že při zachování úvah o způsobu provádění analýzy elementů GUI vůči datovému modelu nebude entita E3 v žádném *Okolí kořenové* entity E1, a proto nebude možné do našeho formuláře umístit jednoduchý element C, korespondující s některým z atributů entity E3. Na obrázku je také naznačeno *Okolí kořenové* entity prvního řádu.

Pokud čtenáře při čtení výše uvedených úvah napadají skutečnosti, které mé definice a algoritmus komplikují má bezpochyby pravdu. K některým těmto skutečnostem se ještě dostaneme.

Některé složitější případy vztahů elementů GUI a atributů ERM

Jednu z možných komplikací nám ukazuje obrázek č. 7. Na rozdíl od obrázku č.4 je zde vyznačeno, že ne pro každý výskyt entity E2 musí existovat výskyt entity E1. O jaký se jedná případ ? Například entita E1 bude *Automobil* a entita E2 bude *Kolo*. *Automobil* může mít několik *Kol*, dokonce většinou právě 4, ale *Kolo* v našem případě nebude muset být vždycky v *Automobilu*, může být například uskladněno ve skladě. Pokud obrátíme naše úvahy na to, jakou funkčnost bychom od našeho formuláře očekávali, je zřejmé, že v případě prohlížení informací o *Kolech* bychom očekávali, že se nám ve formuláři zobrazí i informace o *Automobilech*, ve kterých jsou *Kola* použita. Bude nutné ovšem ošetřit vznik situace, kdy bude *Kolo* uloženo ve skladě, a rozhodnout jakou hodnotu v elementu A našeho formuláře zobrazíme. Možností, jak tuto situaci řešit, je pochopitelně vícero. Jednou z nich je vytvoření fiktivního *Automobilu*, ve kterém jsou všechna *Kola* uložená ve skladě. Tímto způsobem převedeme námi studovaný případ na případ již známý z obrázku č. 4. Každé *Kolo* bude přináležet právě jednomu *Automobilu*.

Další z možných komplikací nám ukazuje obrázek č. 8. Je zcela jisté, že náš výše uvedený algoritmus na zobrazeném datovém schématu nebude úspěšný. Jsou zde pochopitelně dvě možnosti. První z nich je ta, že takové databázové schéma není reálné, a potom se jím nemusíme zabývat. Druhá, že se takové schéma může reálně vyskytovat, a potom je nutné si ujasnit jeho význam a náš algoritmus příslušným způsobem modifikovat, například tak, že každá entita může být právě v jednom *Okolí* daného řádu.

Další zajímavý případ nám představuje obrázek č.9, kde je znázorněn vícenásobný vztah mezi entitami. Je zřejmé, že požadavek na zobrazení formuláře je regulérní. Pokud bude *Kořenovou* entitou entita *Komise*, bude entita *Osoba* v *Okolí prvního řádu Kořenové* entity *Komise*. Z toho plyne, že pokud na formulář umístíme element z entity *Komise*, můžeme naň umístit také elementy z entity *Osoba* a budeme umět sestrojit příslušné SQL příkazy pro jejich naplnění z databáze a pro aktualizaci databáze z těchto elementů. Nicméně bude nutné poněkud upravit výše předložený algoritmus analýzy.

Poslední případ, který chci čtenáři předložit, je zobrazen na obrázku č. 10. Jedná se o zavedení nového pojmu - *Vícenásobný element (Multiply Field)*. Jeho vysvětlení je zřejmé z obrázku. *Vícenásobný element* dokáže zobrazit pro jeden výskyt entity E1 několik výskytů entity E2. Podobně, jako jsme studovali možnost existence *jednoduchých elementů*, máme nyní možnost studovat pravidla existence *vícenásobných elementů*.

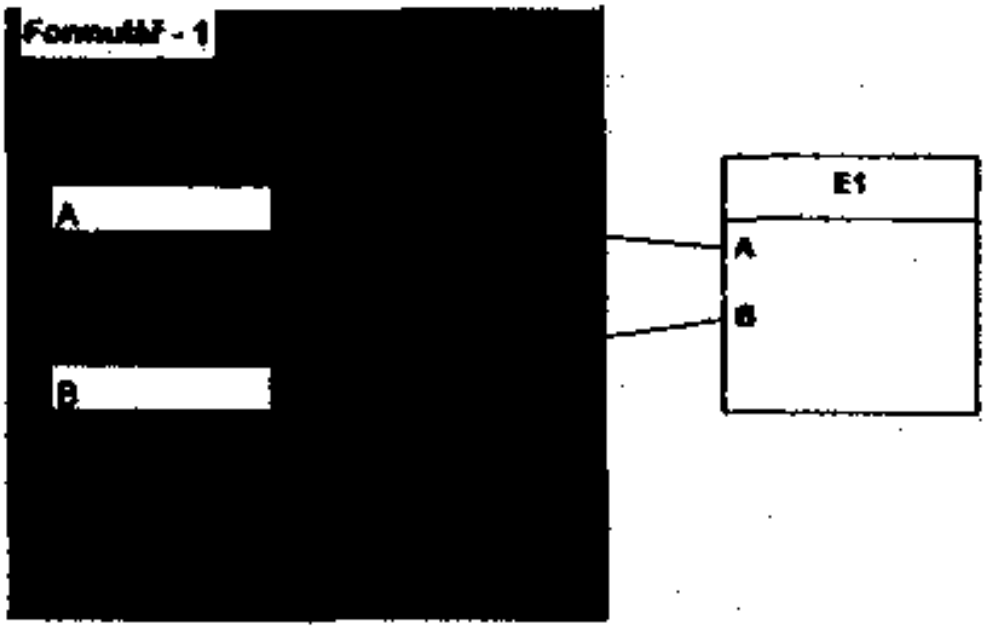
Závěr

Ve svém příspěvku jsem se pokusil naznačit některé souvislosti mezi elementy GUI a atributy ERM. Pochopitelně se mi nepodařilo zachytit tuto problematiku v celé její komplexní složitosti. Možná, že některý čtenář usoudí, že výše uvedené úvahy jsou zbytečné a že celou problematiku je možné řešit intuitivně. Nicméně se domnívám, že existují nejméně dva dobré důvody pro studium souvislostí mezi elementy GUI a databázovým schématem, kterému přináleží.

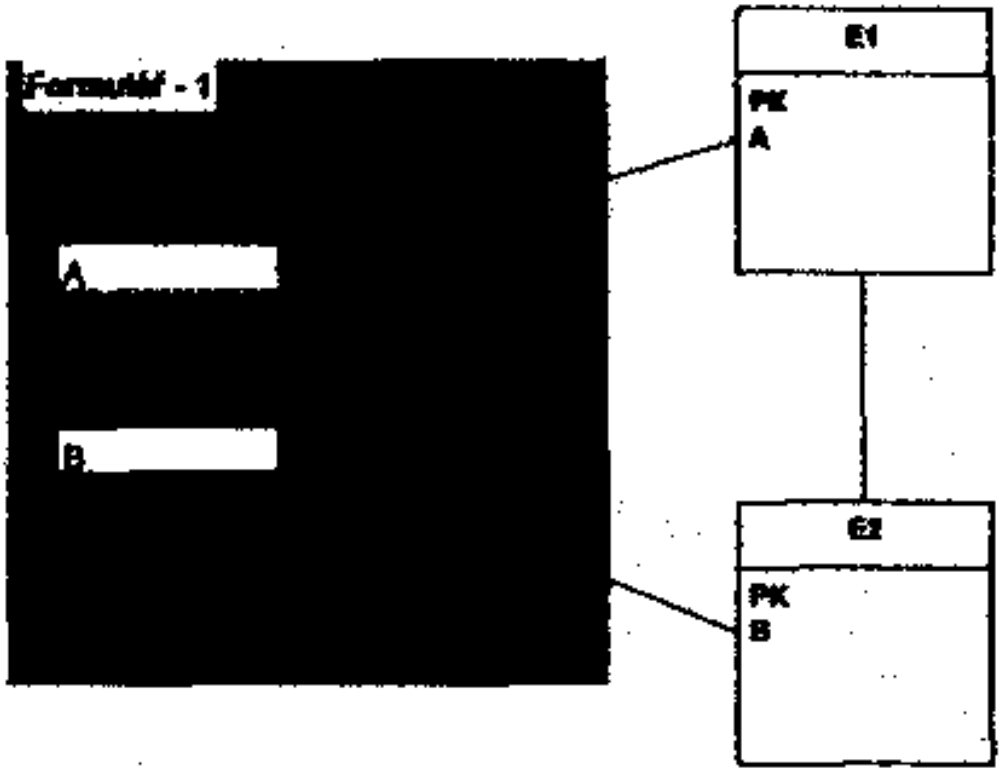
První z nich je možnost vytváření *inteligentnějších* systémů CASE či prostředků typu *Visual Wizard*, nebo *Visual Builder*. Větší inteligence těchto systémů bude spočívat v tom, že při tvorbě GUI budou nabízeny vzhledem k již navrženému GUI pouze určité entity a elementy, které s ohledem na studované souvislosti přicházejí v úvahu.

Druhý důvod je možnost automatické tvorby SQL příkazů pro naplnění formuláře daty z databáze nebo pro aktualizaci databáze hodnotami z formuláře.

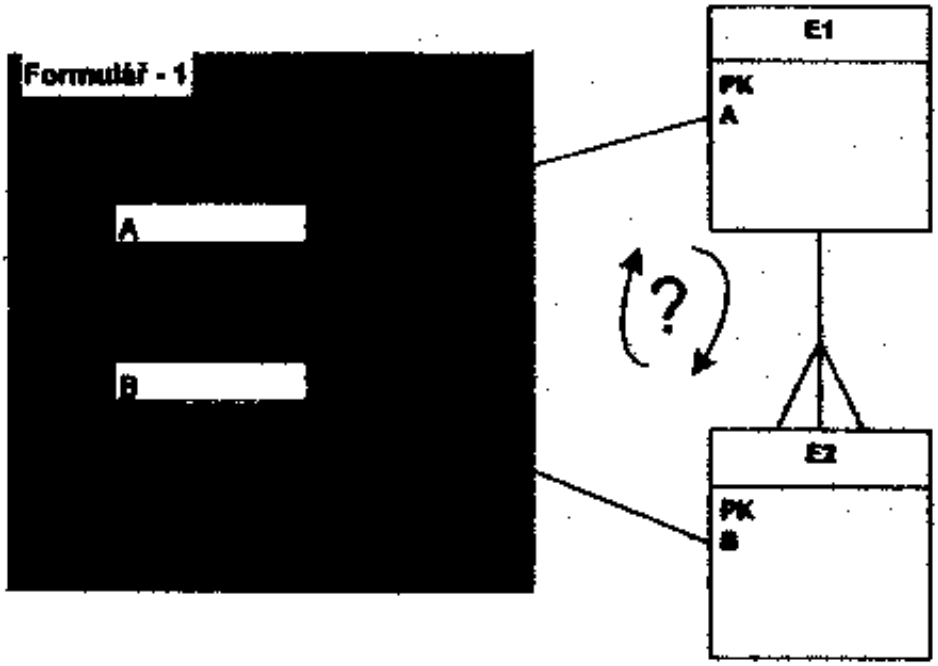
Informace uvedené v tomto příspěvku pochopitelně nestačí pro splnění dvou výše uvedených cílů, jsou však určitým nástinem úvah, které je nutné pro jejich zdárné vyřešení provést. Jejich podrobnější rozvedení může být obsahem některého budoucího příspěvku na této konferenci.



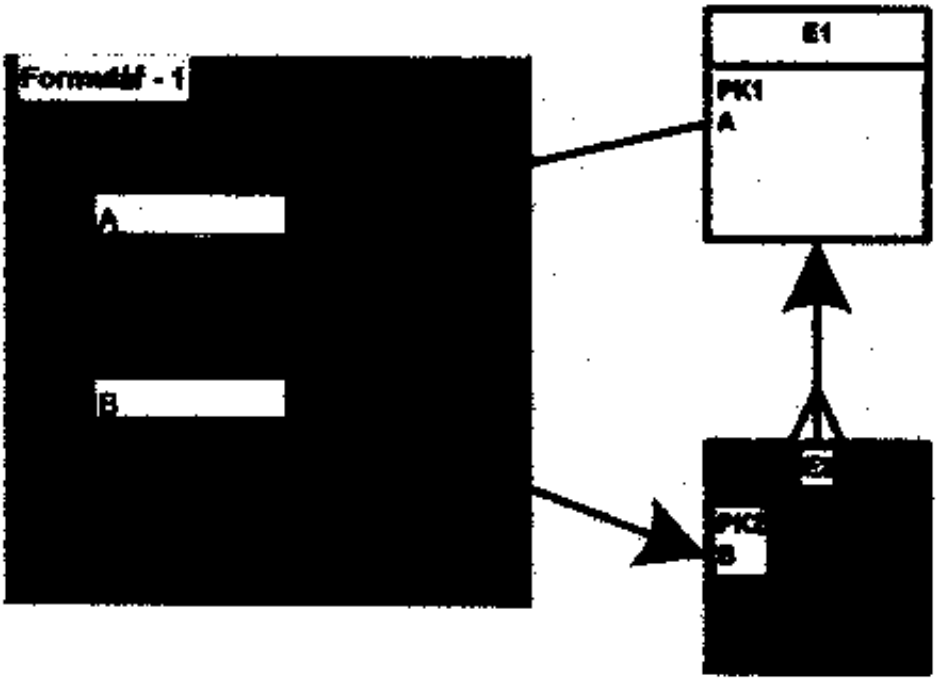
Obrázek 1:



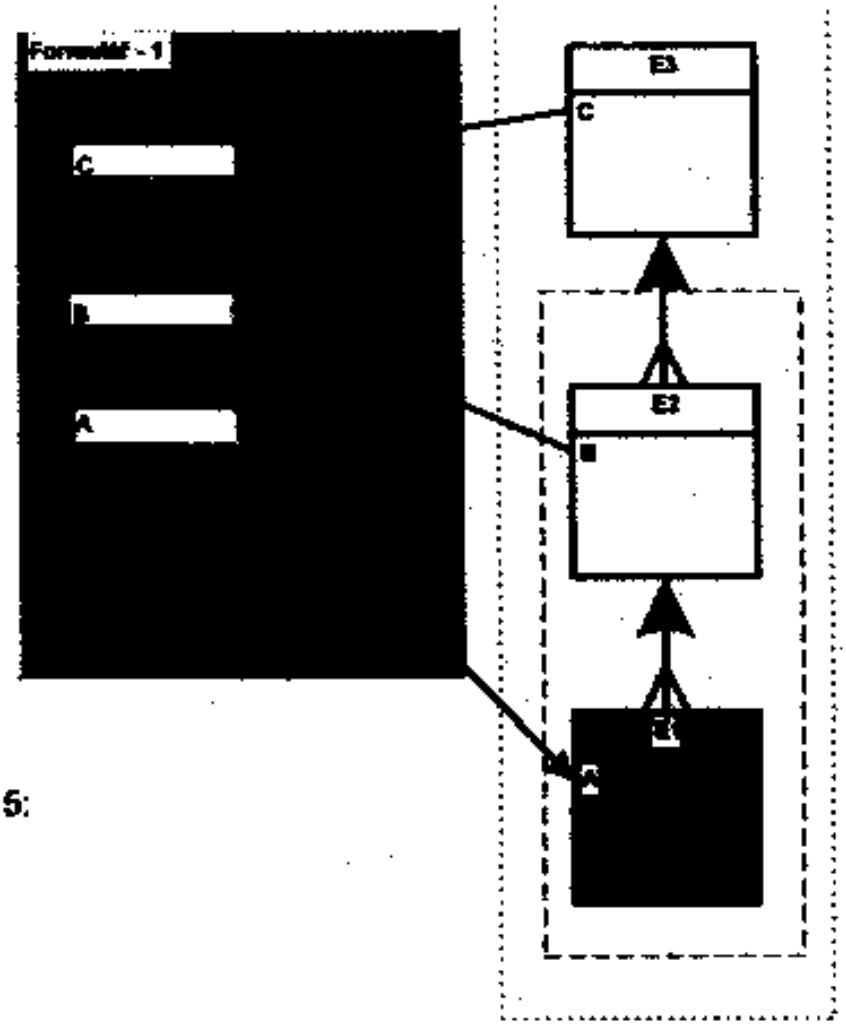
Obrázek 2:



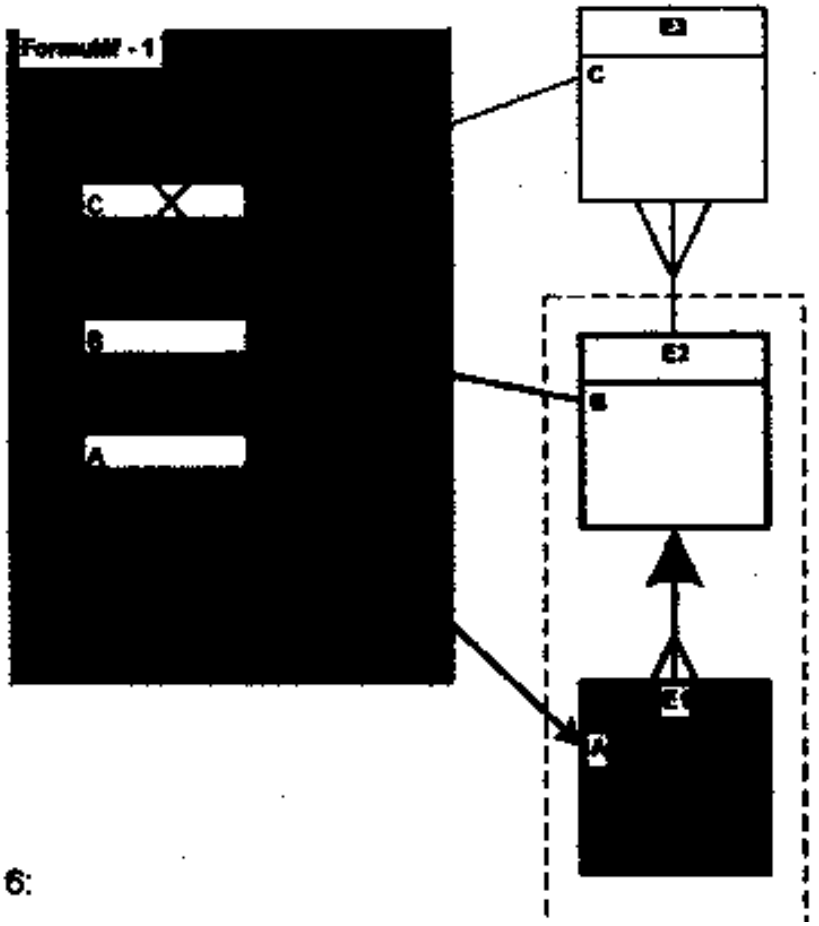
Obrázek 3:



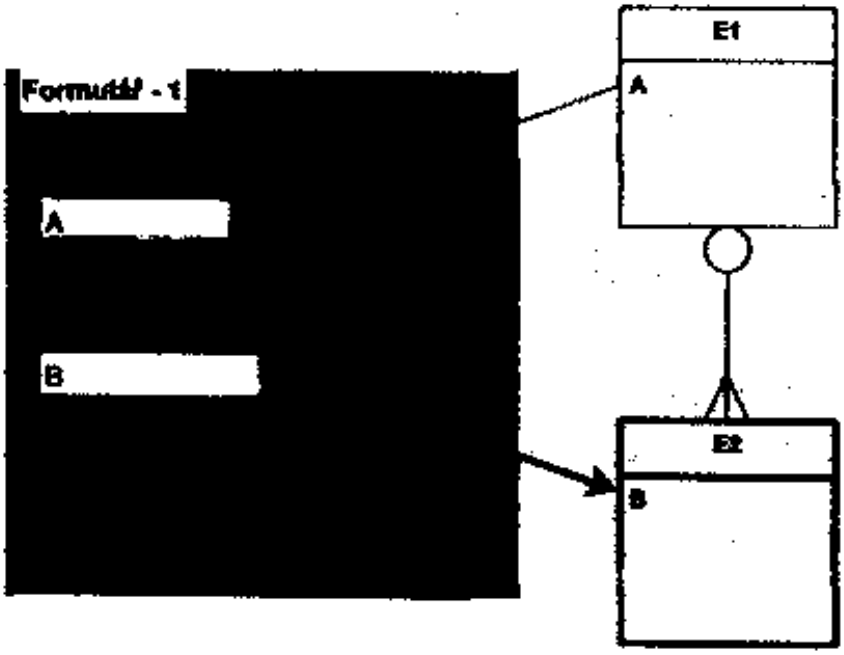
Obrázek 4:



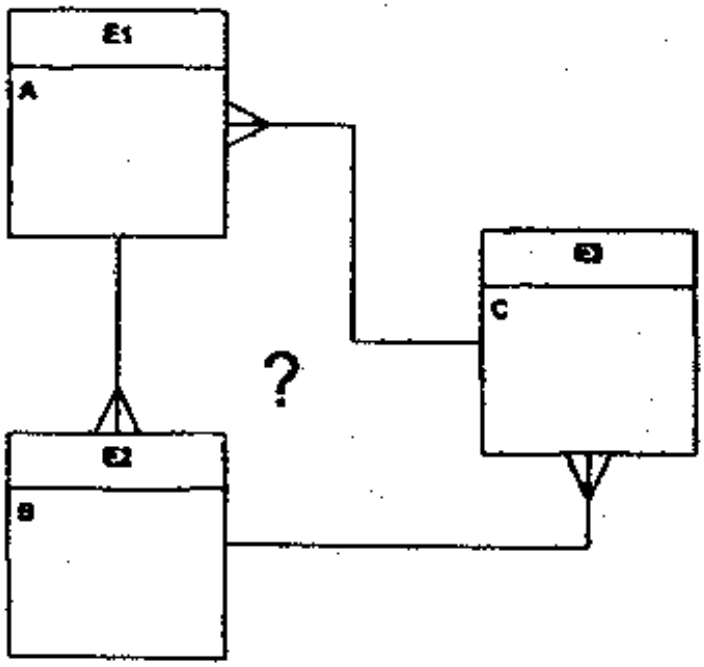
Obrázek 5:



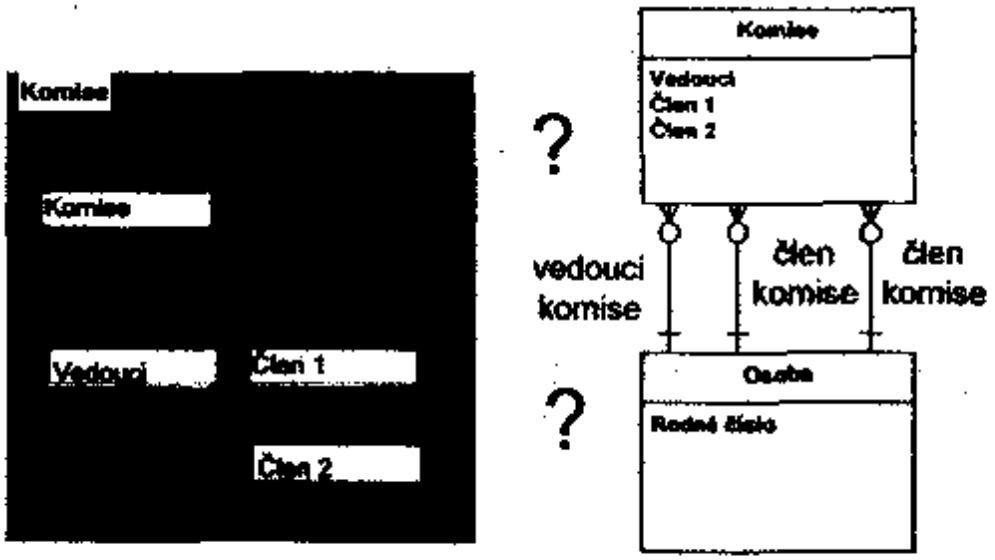
Obrázek 6:



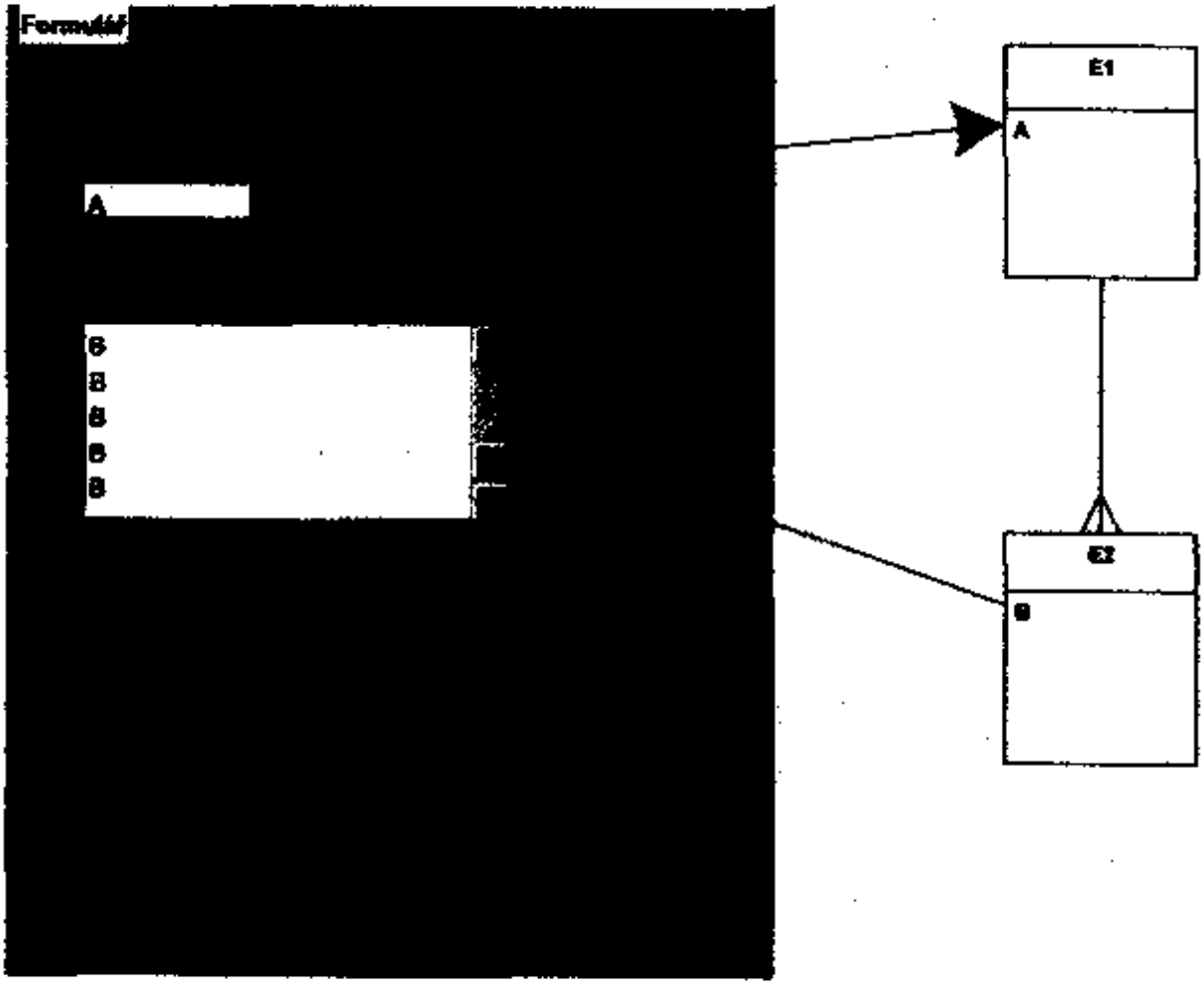
Obrázek 7:



Obrázek 8:



Obrázek 9:



Obrázek 10: