TVORBA SW PRO APLIKACE V REÁLNÉM ČASE: NA BÁZI SPOJENÍ TECHNOLOGIÍ UML A SDL

Jan Šlechta
Ctirad Vrana

AGIT AB, Antala Staška 32, 140 00 Praha 4 - Krč

1. Abstrakt / Úvod

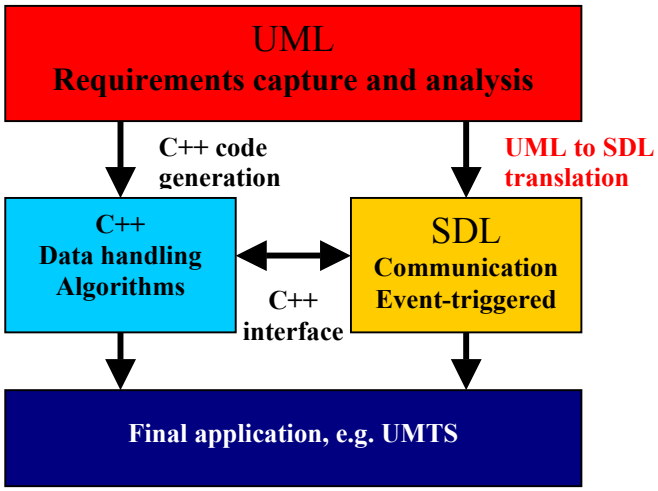
Tato stať je o **dvou specifikačních jazycích UML a SDL**. Jazyk **UML** není zapotřebí prezentovat. Existují také zkušenosti s jeho použitím pro aplikace ¨administrativního typu¨. Jazyk **SDL (Specification and Description Language)** je známý poněkud méně (i když s ním pracují tisíce softwerářů už mnoho let a např. koncept ¨Sequence Charts¨ pochází z SDL, apod.). Kombinace těchto dvou nabízí **možnosti přímo revoluční**.

¨Komunita UML¨ hledala cestu, jak vyplnit ¨**the Real-Time gap**¨. V UML chybí, jak známo, výrazové prostředky typu ¨action language¨, nutné při tvorbě v náročné oblasti aplikací reálného času (RT). V současné době pracuje **OMG** na definování standardu právě pro tyto potřeby – a ukazuje se, že SDL je nejvážnějším kandidátem. Mj. proto, že SDL je silný, standardizovaný jazyk pro aplikace v reálném čase, se kterým jsou mnohaleté pozitivní zkušenosti z oblasti telekomunikací.

Právě tak **ITU-T** (Telekomunikační Unie) učinila v poslední generaci standardu, **SDL-2000**, integrační kroky.

Na schématu je znázorněna možná **návaznost obou technologií**. Za předpokladu, že máme k dispozic integrované prostředí CASE podporující oba jazyky, se tímto mj. získává možnost:

- Efektivně modelovat struktury popisující spolupráci v RT
- Analyzovat např. výskyt ¨dead-locks¨ atd.
- Dynamicky simulovat (funkční chování, RT-kapacitu, apod.)
- Generovat cílový RT-kód, ev. s ohledem na konkrétní HW a OS



Naší snahou je na konkrétním příkladě přiblížit některé z těchto koncepčních ideí a ukázat jejich praktické dopady.

2. Příklad Bank Office System

A. Prekonceptualizace

Tato fáze vývoje software předznamenává fázi analytickou, a stejně tak jako špatná analýza kazí design, platí, že špatný koncept kazí analýzu. Naším konkrétním příkladem je Bank Office systém široce užívaný i v našich zeměpisných šířkách.

Customers (zákazníci) přicházejí do prostoru Office (kanceláře) a jsou obsluhováni Clerks (úředníky). ControlScheduleMachine (Plánovací automat) zajišťuje, aby každý nový zákazník obdržel Ticket (lístek) se svojí identifikací (maximálně 99 zákazníků v kanceláři). Zákazníci tvoří frontu (QueuedCustomers) stejně jako volní úředníci (IdleClerks). Je-li úředník volný, je mu automaticky přiřazen zákazník z fronty. Informace o přiřazení se objeví na Displayi.

B. Analýza

Diagramy 1.-5.

Při analýze použijeme např. MSC, HMSC, Class, State Chart a Deployment diagram. Pro různé typy příkladů – řešení použijeme různé typy diagramů.

C. Design

Diagramy 6.-8.

Designerská fáze je realizována modelováním v jazyce SDL. Modelováním proto, že existuje více návrhů řešení pro každý určitý typ příkladu.

D. Implementace

Diagramy 9. a 10.

Implementace při použití nástrojů, které dnes kolem jazyka SDL existují, je velmi rychlá. Generovaný kód po úspěšné syntaktické a semantické analýze získáme okamžitě a simulací odhalíme naše logická pochybení. Mnohdy i v analytické fázi. To však již souvisí s elaborací.

E. Elaborace

Při elaboraci dochází k iteraci analytické, designerské a implementační fáze vývoje software. Zřídka i k iteraci konceptuální, to už se většinou jedná o nový projekt.

Využitím jazyka SDL a nástrojů s ním spojených získáme právě ve fázi elaborace nejdramatičtější časovou úsporu. Logicky to implikuje vývojářský poznatek, že při dobré a přímé analýze a designu nezávisí tolik na volbě formálních prostředků, ale na počtu znovuvyužitelných komponent – bohatství a relevanci knihoven. Použitím jazyka SDL můžeme navíc tyto knihovny vytvářet i na designerské úrovni. Elaborace pak probíhá v těchto krocích úprav (editací):

1. UML (analytických) a
2. SDL (designerských) diagramů,
3. generování kódu
4. a simulace (nová implementace)
5. až do konečného řešení.

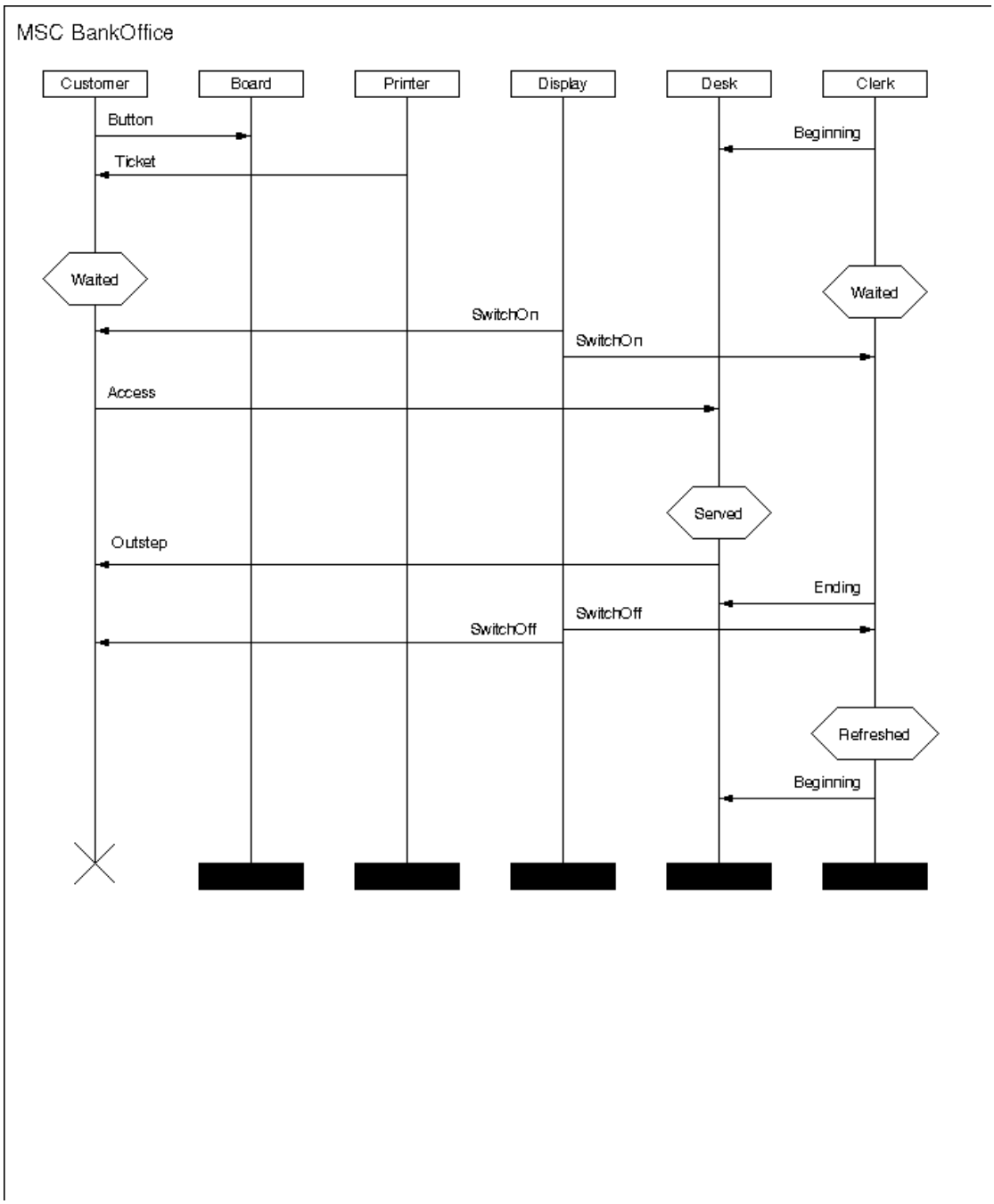


Diagram 1. Message Sequence Chart
Analýza
Systém Bank Office – Typická mise mezi pozorovatelnými aktéry

Diagramem MSC zobrazujeme scénáře (interakce) mezi aktéry misí. Aktéři jsou v zásadě jednající osoby (objekty personální reality – interface objects) a hardwarové komponenty (objekty automatu – entity objects).

Softwarové komponenty (řídící objekty – control objects) přibudou až ve fázi designu.

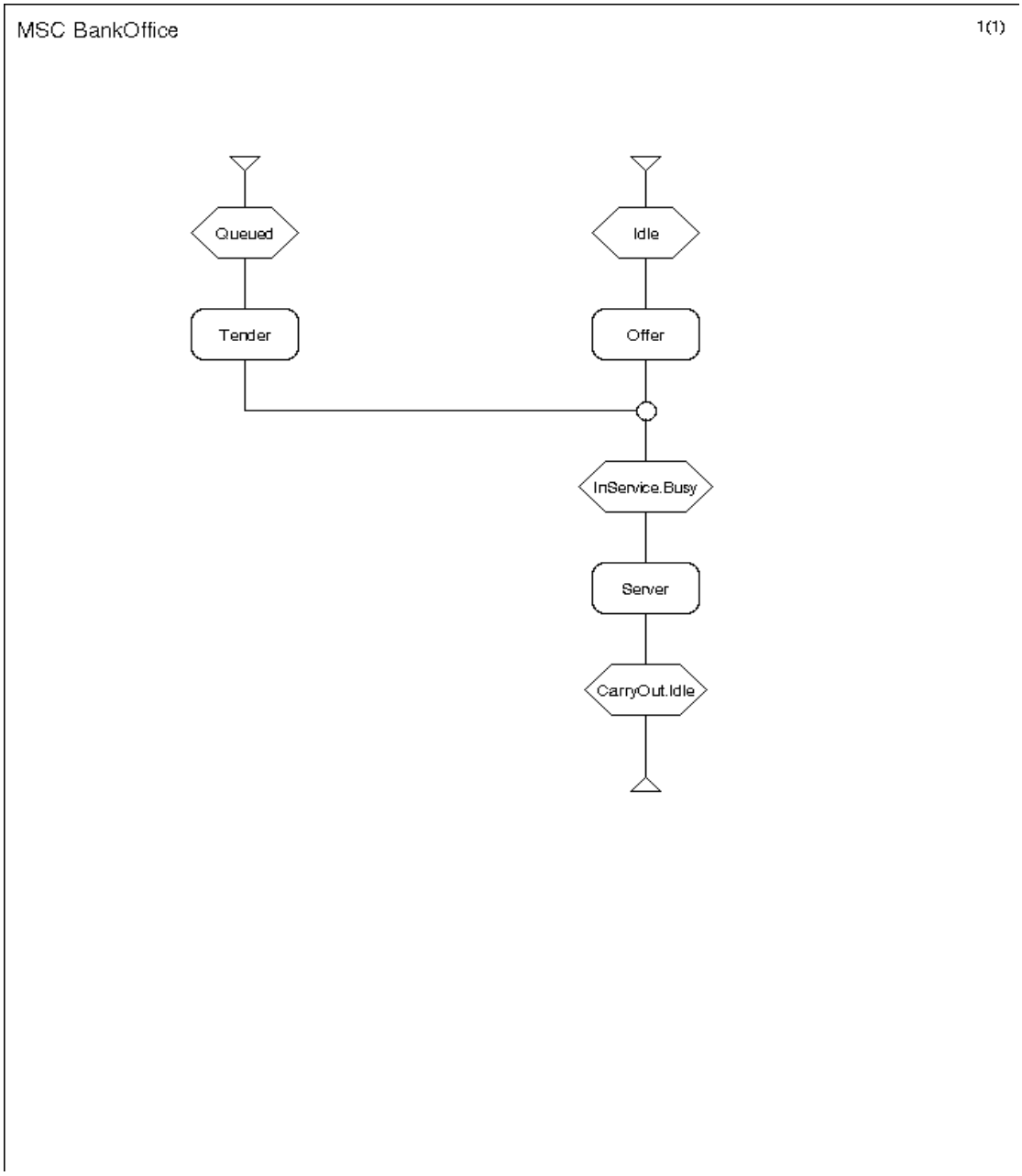


Diagram 2. High level Message Sequence Chart (Road Map)

Analýza

Systém Bank Office – Queue Flow Model

Diagramem HMSC vytipujeme hlavní mise celého systému. Jedná se o misi zákaznickou, která začíná vstupem zákazníka do kanceláře, a misi úřednickou, která začíná dokončením obsluhy jednoho a končí dokončením obsluhy dalšího zákazníka. Obě mise se navzájem spojují a tvoří uspořádané dvojice zákazník-úředník.

Objekty Tender, Offer a Server znázorňují Use Cases (Uživatelské případy) v interakci se systémem. Tyto Use Cases jsou zároveň Queues (fronty), které se v systému tvoří a proto jsou využity v designerské fázi jako softwarové (řídící) objekty (procesy), které tyto fronty spravují.

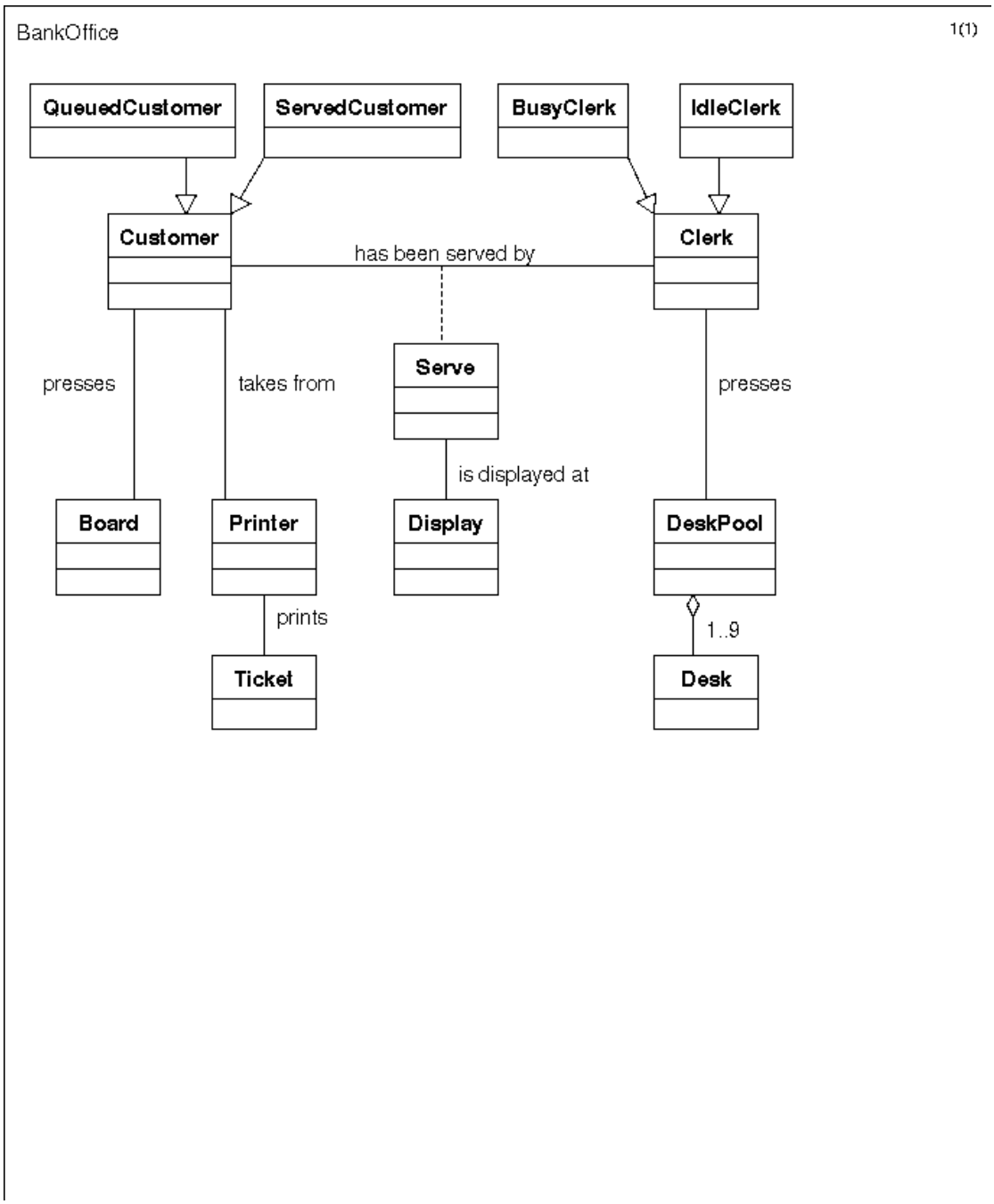


Diagram 3. Class diagram

Analýza

Systém Bank Office – Typy objektů vyskytující se v zamýšleném (pozorovaném) systému

Diagram tříd, a nebo, chcete-li, typů objektů taxonomicky (fenomenologicky) shrnuje vyskytující se entity a relace mezi nimi. Dynamika systému je na rozdíl od HMSC, MSC a SC (jednou z dimenzí je u těchto diagramů vždy čas) znázorněna pouze pomocí asociací. Ostatní prvky notace popisují statickou strukturu systému.

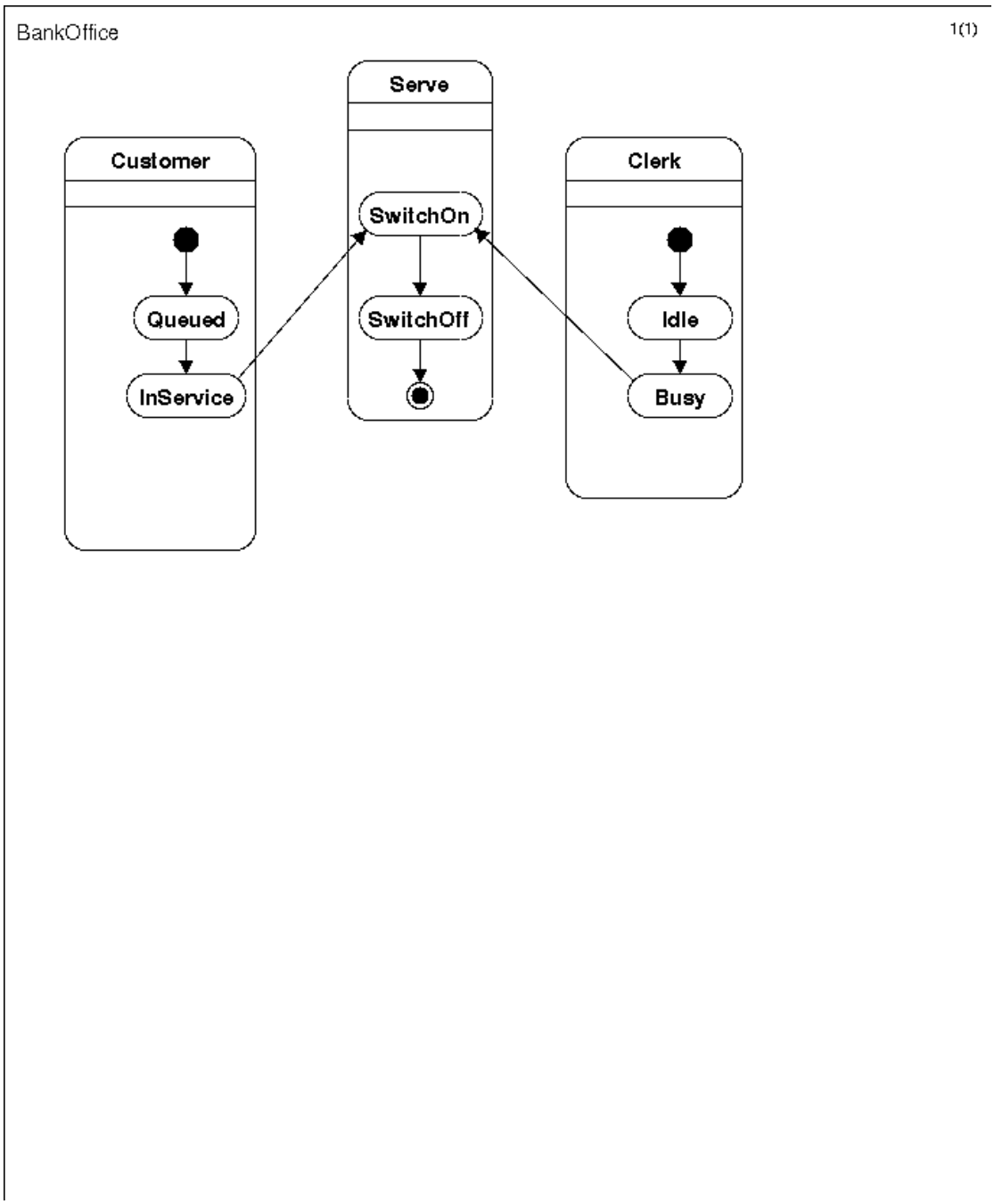


Diagram 4. State Chart diagram

Analýza

Systém Bank Office – Stavové diagramy objektů s více než 1 stavem a jejich propojení

U SC diagramu se explicitně zaměřujeme na vnitřní stavy systému a objektů, zatímco komunikace mezi aktéry je druhořadá a hůře postižitelná.

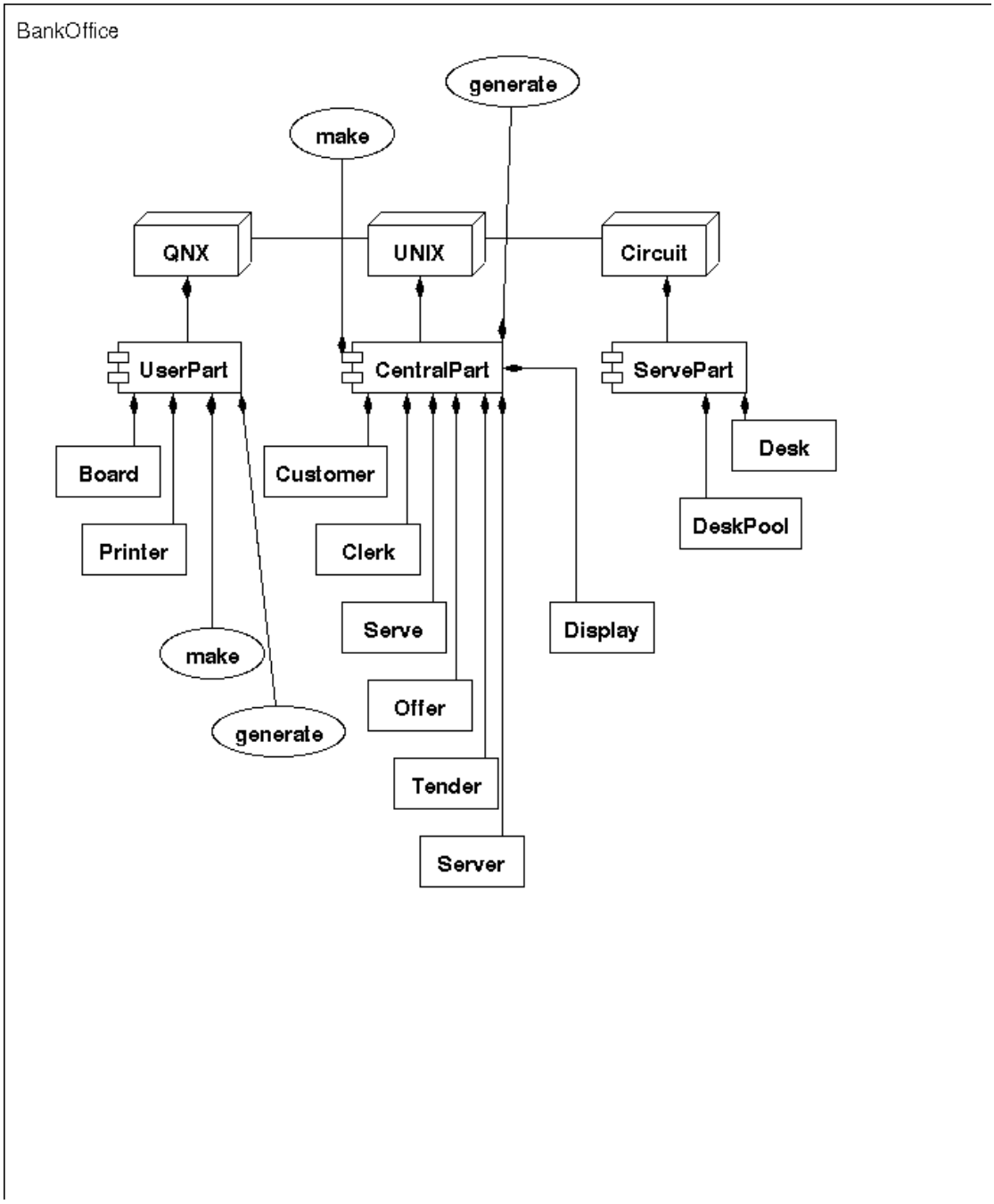


Diagram 5. Deployment diagram

Analýza

Systém Bank Office – Diagram komponent, programů a činností cílového systému

Deployment diagram je vlastně předpisem pro implementaci a svým způsobem ovlivňuje i design, neboť pro různou implementaci je často účelné použít různé designerské techniky.

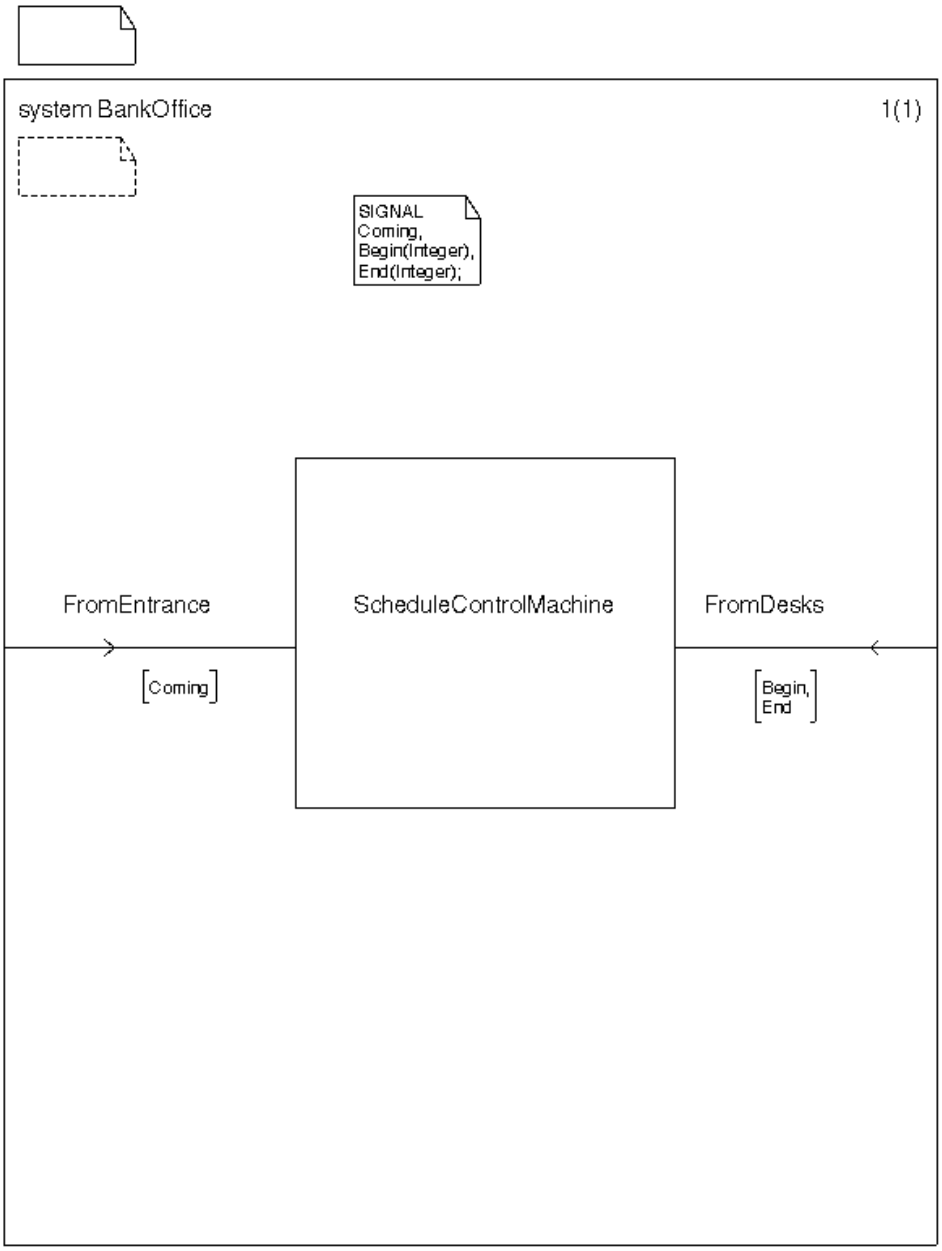


Diagram 6. System diagram
Design
Systém Bank Office – Systémový diagram v SDL jazyce

Na systémovém diagramu je „Plánovací automat“ vyobrazen jako černá schránka. Jiným designerskám přístupem bychom ho mohli rozčlenit na trojice bloků Reality(Interface)Objects, Software(Control)Objects a Hardware(Entity)Objects (tzv. Layer Model) resp. např. Offer(FrontEnd)Objects, Tender(FrontEnd)Objects a Server(BackEnd)Objects (tzv. Flow Model). Mezi bloky by potom byla patrná partikulární komunikace (mezivrstvová resp. časová).

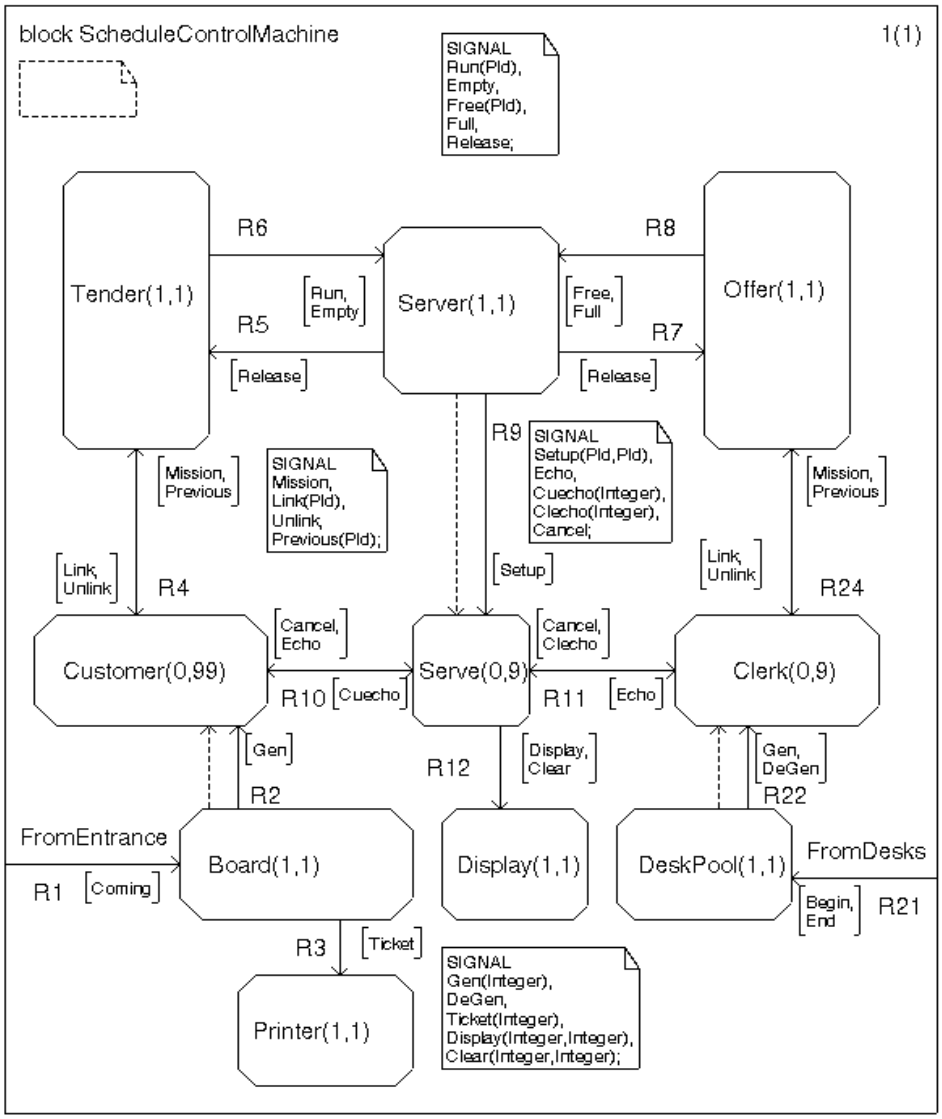


Diagram 7. Block diagram
Design
Blok ControlScheduleMachine – Blokový diagram v SDL jazyce

Blokový diagram zobrazuje asynchronní komunikaci SDL procesů a jako procesy mapuje prakticky všechny zjistitelné objekty analytické fáze (s výjimkou Ticket a Desk). Je skutečně na úvaze designéra jakým způsobem bude mapovat objekty. Mapování „co objekt – to proces“ je na první pohled elegantní, vzápětí však klade nároky na synchronizaci objektů – procesů. Naopak pokud bychom zredukovali celý konglomerát procesů (tyto navíc umožňují reusabilitu - znovuvyužitelnost) Tender, Server, Offer, Customer, Serve a Clerk na jediný objekt – proces Logic ev. Intelligence, vznikl by nám jediný avšak vnitřně složitý proces, který by byl sám o sobě náročný na synchronizaci vnitřních řídících struktur.

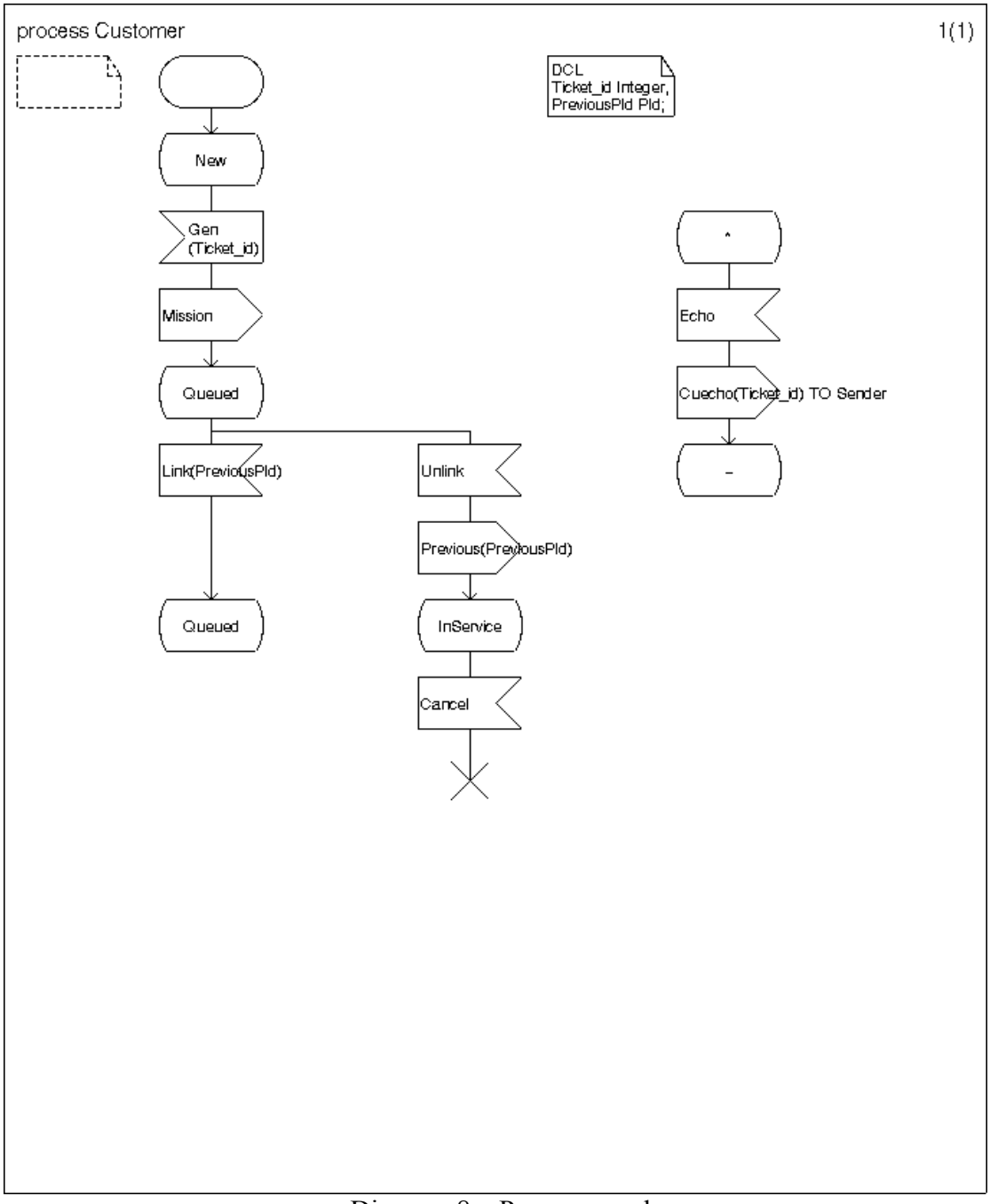


Diagram 8. Process graph

Design

Proces Customer – Procesový diagram v jazyce SDL

Procesový graf modelujeme jako rozšířený konečný automat. V tomto případě je to zároveň model chování reálného objektu Customer. Diagram zachovává stavy z analytického State Chart diagramu: Queued a InService. V rámci designu pak přibírá nutný stav New, ve kterém čeká na předání „generické informace“.

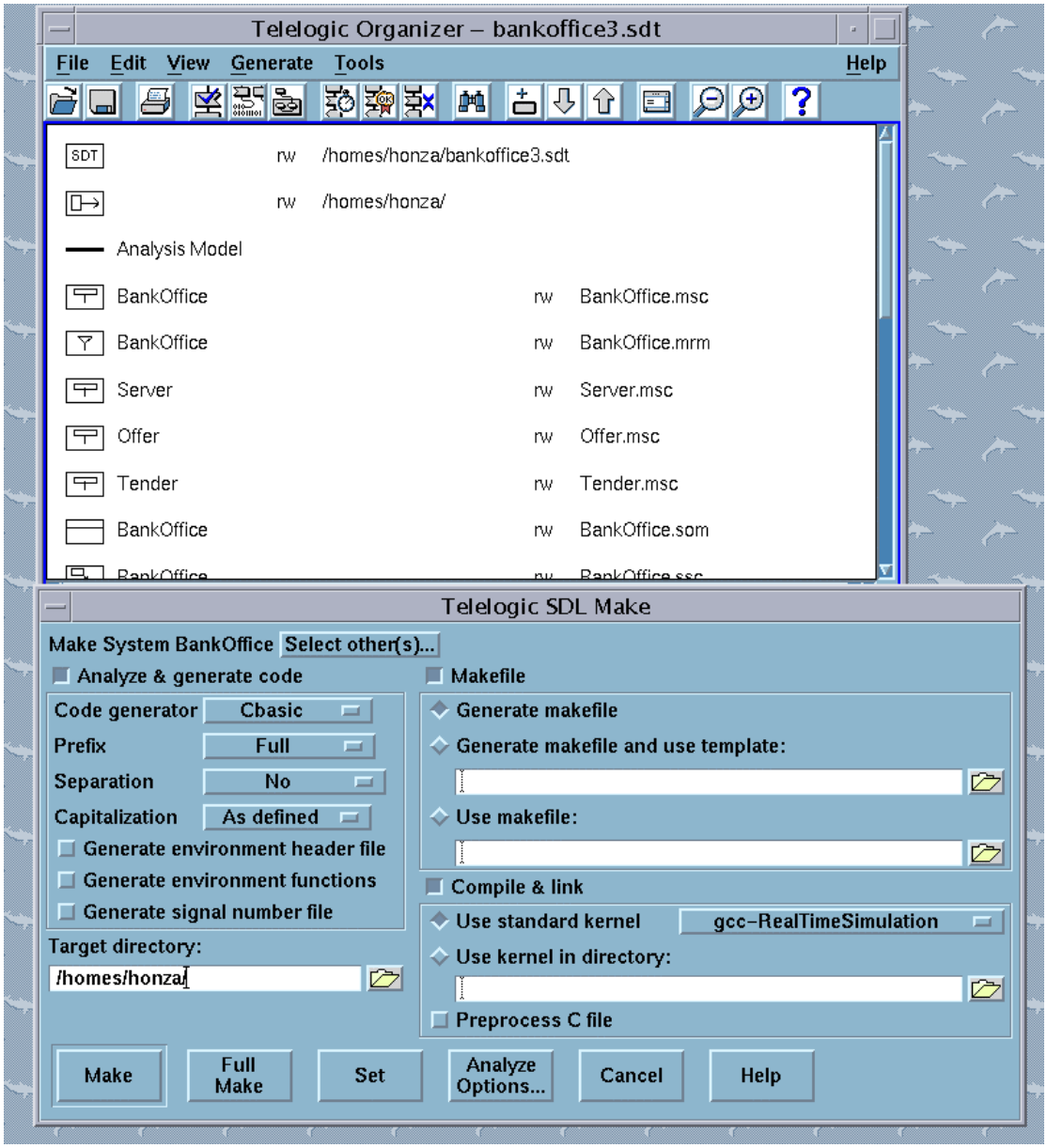


Diagram 9. Make dialogue

Implementace

Systém Bank Office – generování kódu aplikace

Po několika miniiteracích SDL systém versus analyzátor, kdy jsou odstraněny chyby syntaxe, statické a dynamické sémantiky jazyka SDL, jsme připraveni generovat kód v ANSI C/C++ s příslušným jádrem operačního systému reálného času.

Je-li systém separován do různých cílových prostředí, volíme příslušné jádro operačního systému.

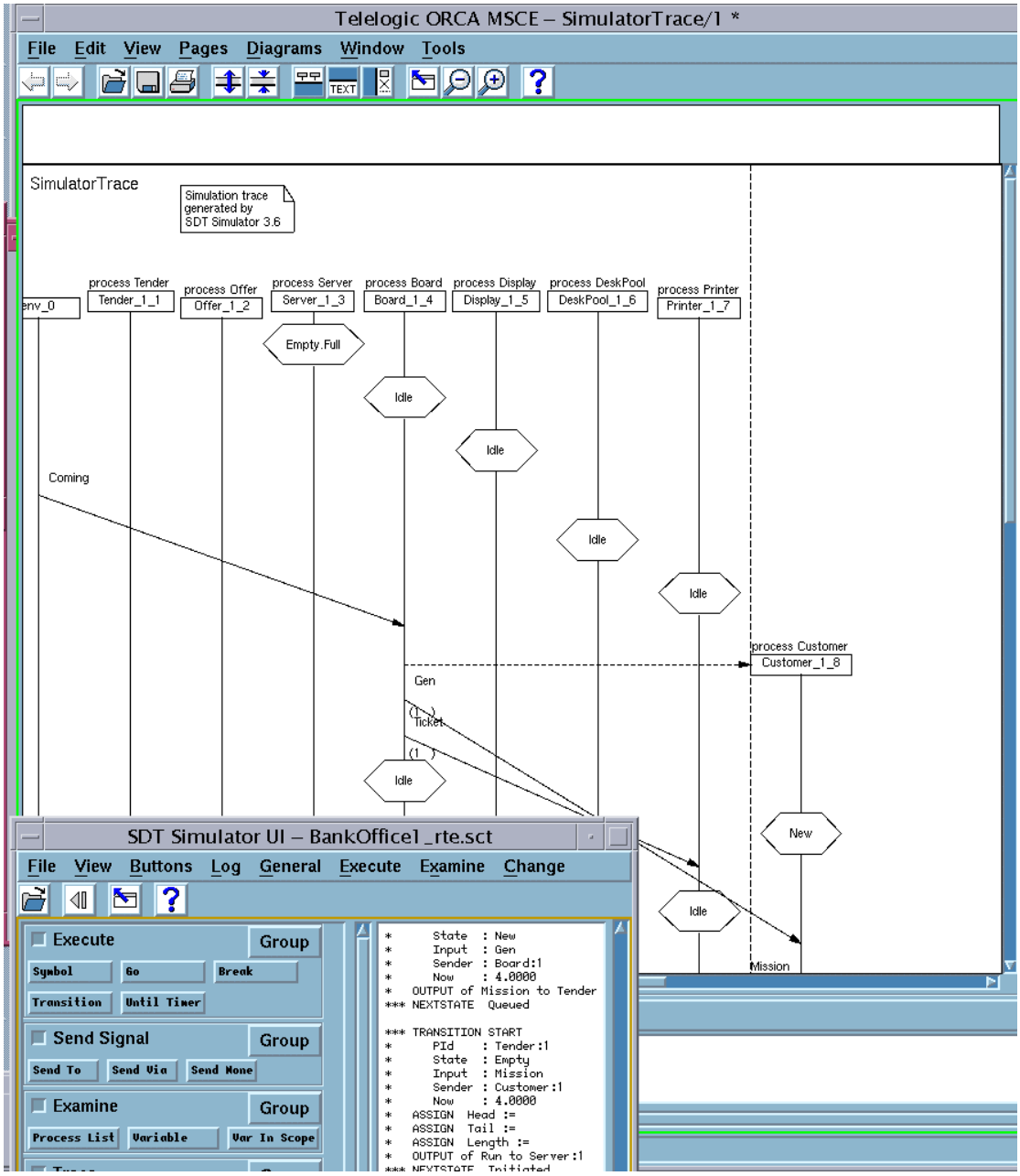


Diagram 10. Simulation dialogue

Implementace

Systém Bank Office – simulace aplikace

Po překladu aplikace jsme okamžitě schopni verifikovat simulační výstup na základě vstupů simulačního dialogu. Výstup je realizován interaktivním MSC, které zahrnuje instance všech procesů SDL systému.

V této fázi simulujeme nejen design v SDL (tzv. poloviční elaborace), ale i analytickou fázi v UML a můžeme dále dle potřeby a nutnosti iterovat.

3. Závěr

Pokud označíme písmenem kapitoly čas, který je potřebný pro realizaci 1 fáze vývoje RT-software dostáváme

T=A+B+C+D+E, (1)

kde zhruba

E=i*(b+c+d), (2)

kde malá písmena značí čas potřebný k **restrukturalizaci** a **editaci** výše uvedených fází a i počet **iterací**.

Protože čas potřebný k implementaci (při zavedeném operačním systému reálného času) je takřka zanedbatelný (limitně se blíží nule) a taktéž vyjdeme-li z relevantní prekonceptualizace na počátku projektu, potom T=B+C+ib+ic.

Značí to, že těžiště práce se přesouvá do **fáze analýzy** a **designu**. Rozhodující pro rychlost dokončení projektu se stává zkušenost s daným **typem systému** a jeho notací v jazyce UML a zkušenost s použitím **designerských vzorů** v jazyce SDL.

Využitím znovuvyužitelných analytických a designerských komponent i vzhledem k tomu, že používáme "**bezešvé**" spojení "**samodokumentačních**" formálních prostředků **UML** a **SDL**, můžeme dramaticky zkracovat **dobu vývoje RT-SW**.

Literatura

1. A Use case Driven Approach (Jacobson, Christenson, Jonnso, Övergaard), Addison-Wesley 1989
2. Use of SDL in European Telecommunications Standards, ETSI report, Sophia Antipolis 1994
3. SDL – Formal Object-oriented language for Communicating Systems (Elsberger, Hogfrefe, Sarma), Elsevier 1997
4. Z.100 – SDL, MSC, ITU-T Recommendation 1998
5. User manuals Telelogic Tau 3.6 – SOMT Methodology Guidelines, Telelogic AB 1999