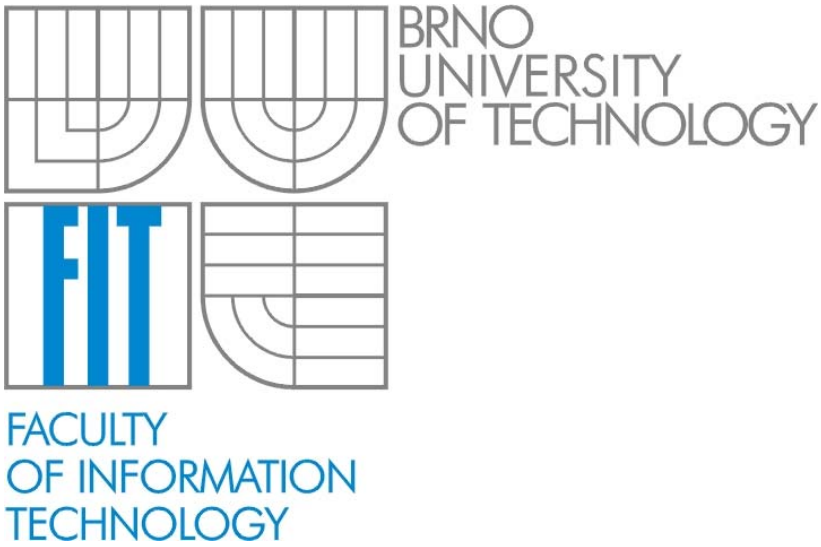


Síťová hra

Dáma

Projektová dokumentace

Zvolená platforma: OS Linux

Autoři a bodové rozdělení:

- Bendl Jaroslav xbendl00 27%
  Vypracoval: Server, skript
- Kopřiva Vít xkopri05 27%
  Vypracoval: XML modul, systém nápovědy
- Kryzhanovsky Maksym xkryzh00 27%
  Vypracoval: Klient
- Chmel Miloš xchmel15 19%
  Vypracoval: Dokumentace, testování

Kapitola 1

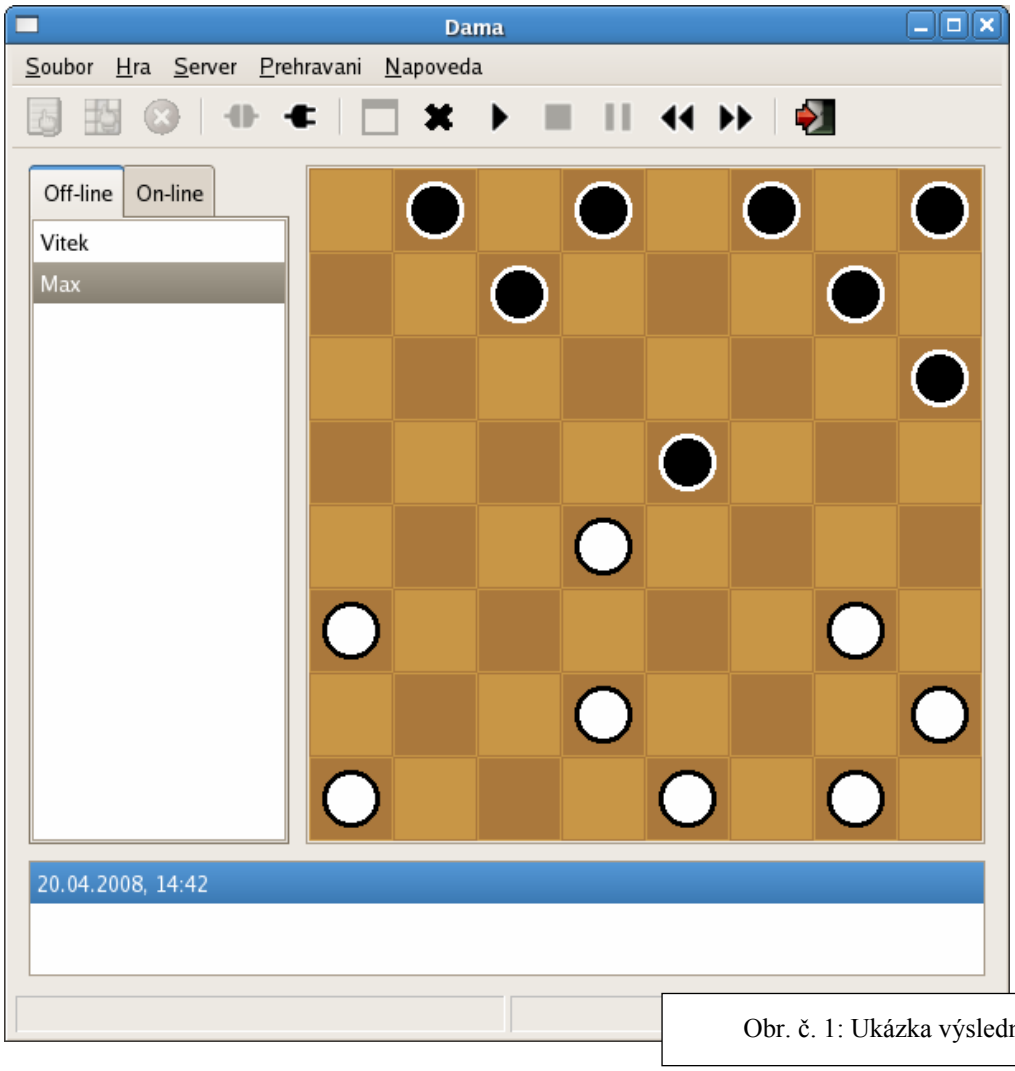
Úvod

Dokumentace popisuje způsob implementace síťové hry Dáma. Jedná se o klient-server aplikaci umožňující kromě samotné hry i archivaci rozehraných a dohraných partií s možností jejich opakovaného přehrávání.

V kapitole 2 se nachází popis realizace klientské části aplikace. Pozornost je věnována implementaci hracího pole a pohybu figurek. V rámci této části je také uveden způsob, jakým jsme pro reprezentaci jednotlivých figurek využili návrhového vzoru Flyweight.

Kapitola 3 popisuje realizaci serveru a komunikace s klientem. Komunikace je realizována pomocí soketů s využitím vlastního aplikačního protokolu. V rámci této kapitoly je také uveden způsob uchování informace o hrách pomocí XML dokumentů.

V kapitole 4 je zhodnoceno zvolené řešení.



Obr. č. 1: Ukázka výsledné aplikace

Kapitola 2

Hra

Dáma patří mezi nejoblíbenější strategické deskové hry pro dva hráče. Hra vznikla zjednodušením pravidel šachu. Základní princip spočívá v pohybu figurek po diagonálách šachovnice s cílem vyřadit přeskočením všechny soupeřovy kameny, případně je znehybnit. Naším úkolem bylo přenést tuto hru do virtuální podoby.

Existuje mnoho variant dámy, zejména v rámci jednotlivých států se standardy pravidel v detailech liší. My jsme si vybrali pravidla České dámy, podle kterých se řídí i náš program.

2.1 Šachovnice

Dáma se hraje se na stejném hracím poli jako šach, tedy střídajících se tmavých a světlých polích, po osmi na každé ose. Jelikož se kameny pohybují pouze na tmavých diagonálách, není nutné v rámci implementace celé šachovnice uvažovat světlá políčka. Tím se počet polí redukuje na polovinu. Celou šachovnici lze tedy popsat polem o 32 prvcích. Toto pole je naplněno odkazy na jednotlivé figurky. V české dámě rozlišujeme dva typy figurek – kameny a dámy. Ty jsou reprezentovány instancemi třídy `Checker` a `Dchecker`.

Figurku lze popsat barvou, typem a pozičním umístěním na hrací ploše. Celkovou konfiguraci šachovnice je tedy možné chápat jako sadu takto reprezentovaných objektů. Při podrobnější analýze tohoto návrhu se však ukázalo, že pozice figurky je jediným jedinečným atributem každé instance objektu, a ty ostatní jsou množinami duplikátů. Tato myšlenka byla rozhodující pro hledání vhodnějšího řešení. Z řady studovaných návrhových vzorů jsme nakonec využili podstaty vzoru Flyweight, která spočívá v oddělení vnitřních informací (barva, typ, …) od vnějších (pozice figurky). Pozice figurky je pak předávána jako parametr metodám popisující chování objektů. Výhoda spočívá v rapidním snížení počtu objektů – Není potřeba vytvářet skutečný počet instancí, ale stačí jen jediná pro všechny figurky stejné barvy a typu (tedy pro figurky se stejnými vnitřními informacemi).

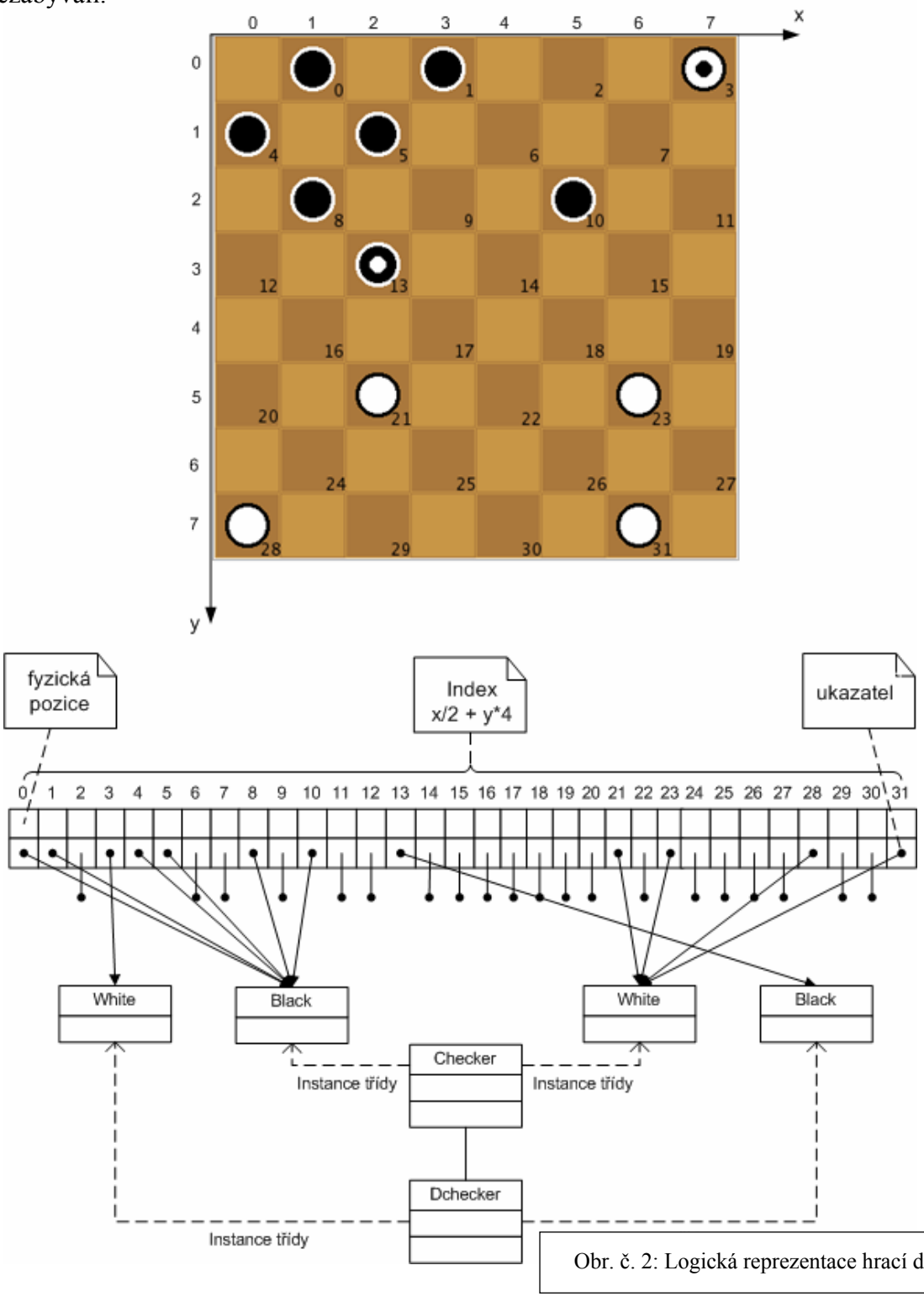
Celkový vzhled šachovnice a jeho provázání s vnitřní reprezentací je uvedeno na následujícím obrázku (obr. č. 2) na konkrétním příkladu konfigurace rozložení figurek. V horní části obrázku je zobrazena hrací deska, v dolní části obrázku je zobrazeno pole fyzických souřadnic umístění figurky (ukazatelů na objekt typu figurky). Pohyb figurek je realizován přemístěním ukazatelů. Prostřednictvím ukazatele je pak možné přistupovat k metodě implementující vykreslení figurky. Konkrétní pozice se dané metodě předává parametrem. Obdobného způsobu bylo využito i při kontrole pravidel.

2.2 Aplikace pravidel

Rozložení figurek bylo zvoleno tak, aby oba hráči měli své figurky přirozeně blíže k sobě. Při startu hry má každý ze soupeřů 12 kamenů rozmístěných dle obrázku č. 1. První na tahu je hráč s bílými kameny. Na základě přijatého tahu soupeře se provede funkce zjišťující kterými figurkami lze táhnout. Toto vyhodnocení probíhá na základě pravidel hry.

Pohyb kamenů po šachovnici je řešen pomocí principu Drag & Drop. V případě, že nelze danou figurkou táhnout, není povoleno ani její zvednutí. Pokud je figurka umísťovaná na nepovolené políčko, vrátí zpět do původní polohy. Je-li možné vykonat libovolnou figurkou na šachovnici skok přes soupeřovu figurku, hráč musí tento tah provést. Podobně je-li možné po skoku vykonat další skok, i ten musí být proveden, a to v rámci jediného hráčova tahu.

Hráč vyhrává, pokud soupeři sebere všechny jeho figurky, nebo je znehybní tak, že soupeři neumožní žádný povolený tah. Případem remízy jsme se v naší implementaci nezabývali.

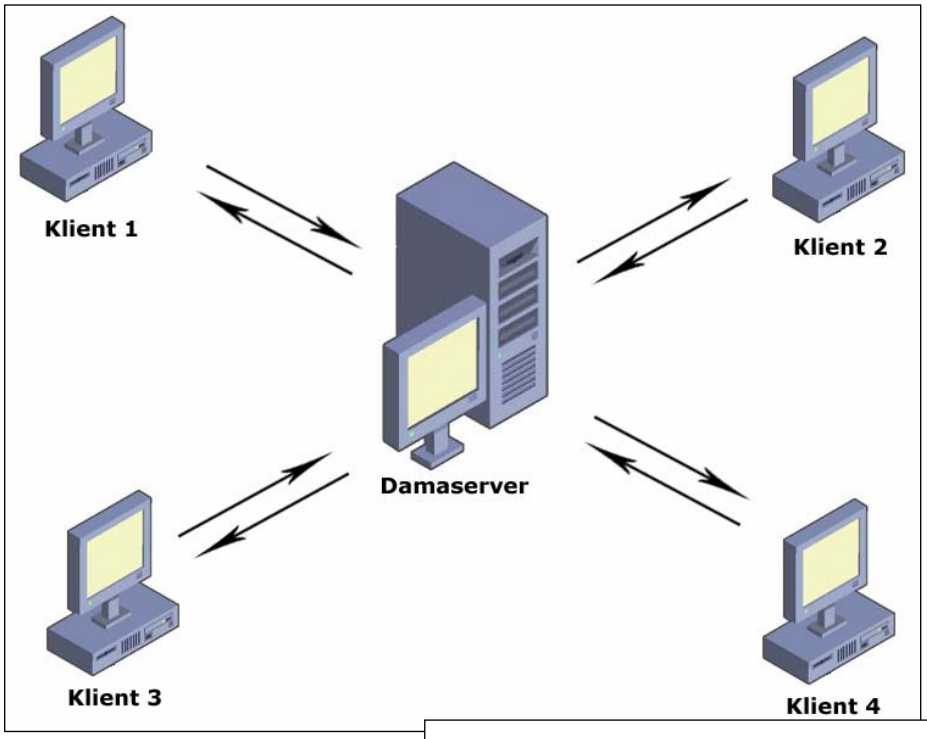


Obr. č. 2: Logická reprezentace hrací desky

Kapitola 3

Komunikace

V této kapitole se nachází popis vzájemné komunikace programu DamaServer2008 (dále jen „server“) a Dama2008 (dále jen „klient“). Komunikace je realizována pomocí soketů přes protokol TCP s využitím vlastního aplikačního protokolu.



Obr. č. 3: Schéma komunikace klientů se serverem

3.1 Komunikace z pohledu serveru

Činnost programu serveru spočívá v připojování klientů a obsluze jejich požadavků. Server musí být schopen komunikovat s více klienty zároveň, proto je nutné zvolit některý z modelů paralelního zpracování požadavků.

Nejjednodušším řešením je vytvořit pro každého klienta vlastní proces. Pokud je však připojených klientů více, informace k procesům zabírají značné množství paměti. Podobný problém vzniká i při použití vláken, které navíc výrazně zatěžují plánovač.

Z výše uvedených důvodů jsme se chtěli vyhnout situaci, kdy pro každého připojeného klienta budeme vytvářet nový proces / vlákno. Tento požadavek splňovala funkce `poll`, pomocí které byl nakonec server realizován. Princip této funkce spočívá v zablokování procesu, dokud se na některém z množiny soketů (hlavní soket serveru a sokety připojených klientů) neobjeví nepřečtená data. Tyto na hlavním soketu serveru znamenají požadavek nového klienta o připojení, zatímco u ostatních soketů jde o příchozí zprávu. Podle prvního

byte zprávy se určí o jaký typ požadavku půjde a vyvolá se příslušná obslužná funkce z modulu `ServerCom`.

Slabinou zvolené metody je její časová složitost v situaci, kdy má množina kontrolovaných soketů velké množství prvků a zároveň se data na soketech objevují v krátkých časových intervalech. Při každé změně na jediném soketu je totiž nutné procházet přes všechny připojené klienty.

3.2 Komunikace z pohledu klienta

Na straně klienta bylo pro komunikaci použito možností toolkitu wxWidgets. Zatímco zasílání a příjem zpráv pracuje podobně jako na serveru, samotné čekání na zprávu je řešeno jiným způsobem. Třída `wxSocketEvent` umožňuje využít vyvolávání událostí při změně na socketu a podle typu dané události (ztráta spojení, nová data, ...) je tak možné spustit příslušnou obslužnou funkci.

3.3 Aplikační protokol

Pro komunikaci klienta se serverem jsme si vytvořili vlastní aplikační protokol. Ten definuje množinu zpráv, které obě strany rozumí. První byte u všech zpráv určuje jejich typ, dále následuje samotný obsah. Úplný výčet zpráv s podrobným popisem se nachází v programové dokumentaci, v níže uvedeném příkladu je uvedena ukázka několika z nich:

CLI_SIGN **Žádost o přihlášení na server**

0x10|NAME Klient -> Server

Parametry:

NAME	char[PLAYER_LEN]	Jméno přihlašovaného hráče

CLI_ASK_GAME **Žádost protihráče o dohrání rozehrané hry**

0x20|OPPONENT|GAME_ID|DATE|ONSTEP Klient_1 -> Server

Parametry:

OPPONENT	char[PLAYER_LEN]	Jméno protihráče
GAME_ID	int	Identifikátor rozehrané hry
DATE	time_t	Čas započetí hry
ONSTEP	unsigned char	Příznak určující, jestli bude Klient_2 na tahu

---

CLI_GAME_LIST **Žádost o zaslání seznamu rozehraných her s daným protihráčem**

0x60|OPPONENT Klient -> Server

Parametry:

OPPONENT	char[PLAYER_LEN]	Jméno protihráče

3.4 Záznamy her

Pro uchování informace o hrách jsme využili toolkitu TinyXML poskytující rozhraní pro práci s XML dokumenty. Standardně se hry ukládají do souboru xml/games_client.xml (na straně klienta) a xml/games_server.xml (na straně serveru), které se vytvoří automaticky při prvním spuštění programu. Dále je možné načíst libovolný jiný XML soubor pomocí volby z menu Soubor-Archív.

Na straně klienta je zaznamenávání her možné zapínat či vypínat přímo při startu programu (příslušný ovládací prvek je umístěn v rámci přihlašovacího okna) nebo z menu Soubor-Konfigurace. Na straně serveru je zaznamenávání zapnuté neustále. Jakmile je však partie dohrána, záznam se ze serveru smaže. Funkce serveru tedy nespočívá v archivaci her, nýbrž v uložení rozehraných partií v podobě umožňující dokončení hry z libovolné klientské aplikace.

V níže uvedeném příkladu je ukázka dokumentu games_server.xml po vložení jedné hry, v rámci které byly provedeny dva tahy.

```
<GAMES>
    <GAME>
        <GAME_ID>1</GAME_ID>
        <PLAYER_1>Jarda</PLAYER_1>
        <PLAYER_2>Vitek</PLAYER_2>
        <DATE>1208360353</DATE>
        <ON_STEP>Jarda</ON_STEP>
        <FINISHED>0</FINISHED>
        <GAMESTEPS last_step_id="2">
            <STEP step_id="1">
                <DATA>11</DATA>
                <DATA>15</DATA>
            </STEP>
            <STEP step_id="2">
                <DATA>9</DATA>
                <DATA>13</DATA>
            </STEP>
        </GAMESTEPS>
    </GAME>
</GAMES>
```

Kapitola 5

Závěr

V rámci řešení tohoto projektu se nám podařilo vytvořit síťovou hru Dáma. Jedná se o klient-server aplikaci umožňující kromě samotné hry i archivaci rozehraných a dohraných partií s možností jejich opakovaného přehrávání.

Při návrhu klientské aplikace s grafickým uživatelským rozhraním byl kladen důraz na jednoduchost ovládání a příjemný vzhled. Jelikož funkčnost klientské aplikace je zcela závislá na spolehlivě fungující komunikaci, na straně severu byla velká pozornost věnována ošetření možných chybových stavů. Součástí odevzdaného řešení je i systém nápovědy, který je možné spustit přímo z klientské aplikace. Program tedy splňuje požadavky stanovené v zadání, ačkoliv při provozním nasazení by jej bylo vhodné rozšířit o autorizaci uživatelů na základě hesla. Autoři si uvědomují, že použití čekání na více událostí na straně serveru pomocí systémového volání `poll` není zcela optimální a při větším počtu připojených klientů má server pomalejší odezvu.

Řešení je plně funkční pod platformou OS Linux.