VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

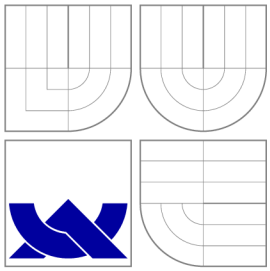
SYSTÉM PRO ZÁZNAM A OPAKOVÁNÍ UDÁLOSTÍ PRO ZVUKOVÉ SYSTÉMY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

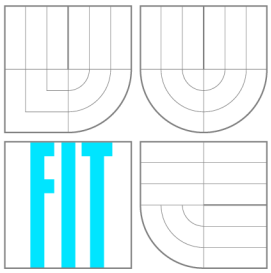
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KLOBÁSA

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYSTÉM PRO ZÁZNAM A OPAKOVÁNÍ UDÁLOSTÍ PRO ZVUKOVÉ SYSTÉMY

SYSTEM FOR RECORDING AND REPEATING OF EVENTS FOR SOUND SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KLOBÁSA

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2015

Abstrakt

Práce se zabývá návrhem a realizací systému pro záznam a opakování událostí pro zvukové systémy. Úvodní část práce je věnována studiu zvukových systémů, jejich uživatelskému rozhraní a komunikačním protokolům, které jsou zvukovými systémy využívány. Dále je prezentována základní charakteristika platformy Mac OS X, její uživatelské rozhraní a principy vývoje na této platformě. Jádro práce tvoří návrh systému pro záznam a opakování událostí a konceptu událostí, který specifikuje reprezentaci událostí v systému a jejich následné zpracování. V závěru práce je představena implementace navrženého systému pro Mac OS X s ohledem na možnosti rozšíření systému a jeho vlastnosti.

Abstract

This thesis deals with the design and implementation of system for recording and repeating of events for audio systems. The introductory part is devoted to the study of sound systems, their user interfaces and communication protocols that are used by audio systems. Hereafter the description of essential features of Mac OS X, as well as the characteristic of its user interface and principles of development on this platform, are given. The core of the thesis consists of the design of system for recording and repeating of events and the concept of events which specifies the reprezentation of events in the system and their subsequent processing. In the end of the thesis is presented the implementation of the designed system for Mac OS X with respect to possible extensions of the system and its features.

Klíčová slova

uživatelské rozhraní, Mac OS, zvuk, zvukový systém, záznam událostí, opakování událostí, MIDI, OSC, Cocoa

Keywords

user interface, Mac OS, sound, audio system, event recording, event repeating, MIDI, OSC, Cocoa

Citace

Jiří Klobása: Systém pro záznam a opakování událostí pro zvukové systémy, diplomová práce, Brno, FIT VUT v Brně, 2015

Systém pro záznam a opakování událostí pro zvukové systémy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Klobása
25. května 2015

Poděkování

Rád bych poděkoval vedoucímu práce panu prof. Dr. Ing. Pavlu Zemčíkovi za odborné vedení, ochotu a čas, který mi věnoval při konzultacích. Dále bych také rád poděkoval firmě DISK Multimedia s.r.o. za odborné konzultace a vybavení pro realizaci této práce.

Obsah

Kapitola 1

Úvod

V poslední době ceny počítačů a počítačových systémů znatelně klesají a současně roste i jejich výkonnost. Důsledkem toho je také snaha o jejich maximální využití. Tato situace se promítá i do oblasti systémů pracujících se zvukem prostřednictvím rozvoje digitálního zpracování zvuku. Zvuk, jakožto jedna z možností předávání informace, je dnes součástí celé řady systémů, které jsou svým zaměřením velmi různorodé.

Mezi těmito systémy lze nalézt specializované systémy pro určitou úlohu a také systémy, které využívají zvuk pouze jako jeden z možných komunikačních kanálů. Takové systémy jako kolekce elektronických zařízení se stále více vyskytují i v oblastech, ve kterých dříve nebyly využívány. V dnešní době jsou k nalezení prakticky v každém divadle, kině, konferenčních místnostech, inteligentních domácnostech, školách nebo jako součást větších multimediálních zařízení.

Ke konfiguraci takového systému jsou využívány nejen hardwarové ovládací prvky, kterými jsou tlačítka, přepínače nebo například otočné potenciometry, ale také jejich odpovídající reprezentace v softwarové podobě grafického uživatelského rozhraní. Tato podoba poskytuje několik výhod, a to především úsporu finančních prostředků spojených s výrobou řídících zařízení a nutné kabeláže, větší flexibilitu změny uživatelského rozhraní a také snazší automatizaci procesu konfigurace jednotlivých zařízení.

Jejich řízení obvykle probíhá formou příkazů, které jsou obecně nazývány událostmi. Na tyto události systém reaguje určitým chováním, změnou své aktuální konfigurace nebo provedením patřičné akce. Tyto události je rovněž možné zaznamenávat, opakovat a také plánovat, čímž je možno systém automatizovaně řídit. Plánování událostí za účelem ovládání zvukového systému přináší nejen usnadnění procesu řízení pro uživatele, ale také eliminaci možných chyb způsobených lidským faktorem.

Během své činnosti tyto systémy také produkují události, které popisují výsledky jejich aktivity nebo pouze informují o změnách konfigurace daného zařízení. Příkladem může být hra na elektronický hudební nástroj, který produkuje řadu událostí reprezentujících jeho aktivitu, automatická detekce událostí ve zvuku prováděná specializovaným systémem pro zpracování signálů, událost informující o změně konfigurace zařízení v závislosti na aktuální situaci a mnoho dalších.

Hlavní motivací pro tvorbu tohoto systému byla kontrola činnosti více různých zvukových zařízení a zpracování události generovaných těmito zařízeními v závislosti na jejich aktuálním stavu. Zaznamenané události je systém schopen reprodukovat, plánovat jejich vykonání a případně je ve stanovených časových intervalech opakovat. Mimo záznam těchto událostí umožňuje systém také vytvářet nové události, kterými jsou zvukové systémy řízeny. Kromě těchto vlastností systém také vykazuje jistou schopnost adaptivního chování

při zpracování událostí. Na vstupní události systém může reagovat, měnit své chování, a tak předcházet nežádoucím stavům u řízených systémů. Vzhledem k velkému množství potenciálně řízených zvukových systémů si tato práce kladla za cíl navrhnout systém tak, aby pracoval s libovolným zvukovým systémem a byl snadno rozšiřitelný o nové události a komunikační protokoly.

Následující kapitola 2 seznamuje čtenáře se základními vlastnostmi zvuku a jeho digitální reprezentací. Dále jsou popsány funkce zvolených zvukových systémů a jejich komunikační protokoly. V kapitole 3 je shrnuto uživatelské rozhraní typické pro zvukové systémy a možnosti implementace uživatelského rozhraní pomocí multiplatformních knihoven. Závěr kapitoly 3 prezentuje možný popis struktury zvukového systému. Další kapitola 4 nabízí bližší seznámení se základními rysy, principy vývoje a charakteristikou uživatelského rozhraní operačního systému Mac OS. Kapitola 5 analyzuje současný stav zvukových systémů a stanovuje požadavky na výsledný systém. Popis návrhu systému, jeho stavba a rozdělení do funkčních bloků uvádí kapitola 6. V rámci návrhu systému je představen koncept událostí v systému a jejich možné zpracování. Je zde také diskutován návrh uživatelského rozhraní systému. V kapitole 7 je prezentována realizace systému na platformě Mac OS a mapování návrhu systému na implementované funkční bloky. Poslední kapitola shrnuje postup při řešení této práce, cíl práce a stav jejího naplnění. Závěr kapitoly 8 popisuje další plánované pokračování této práce.

Kapitola 2

Zvuk, zvukové systémy a jejich komunikační protokoly

V této kapitole jsou popsány základy fyzikální charakteristiky zvuku a jeho převod do digitální reprezentace. Přestože tato práce není přímo zaměřena na signálové zpracování zvuku, je vhodné se seznámit s principy, se kterými zvukové systémy pracují. Zvukové systémy a jejich funkce úzce souvisejí s událostmi, které realizovaný systém zpracovává. Proto je součástí této kapitoly také popis vybraných zvukových systémů, které provádí záznam, zpracování nebo reprodukci zvuku. Závěrem této kapitoly jsou prezentovány zvolené komunikační protokoly, které představují koncepty využívané při komunikaci se zvukovými systémy. Oblast zvukových systémů je poměrně velmi rozsáhlá co do počtu různých zařízení a jejich zaměření. Z tohoto důvodu jsou zde prezentovány pouze některé zvukové systémy a komunikační protokoly nutné pro orientaci v následujících kapitolách.

2.1 Zvuk a jeho digitalizace

Zvuk lze definovat jako kmitavý pohyb částic v látkovém prostředí, při kterém dochází k přenosu energie mezi částicemi [20]. Díky této energii se zvuk šíří v prostoru v podobě mechanického vlnění od zdroje zvuku k posluchači, kde vyvolává zvukový vjem.

Zdrojem zvuku je každé kmitající těleso jako například kytarová struna nebo membrána reproduktoru. Kvalita vlnění v okolí zdroje zvuku je určena nejen jeho kmitáním, ale také jeho geometrickým tvarem.

Frekvenční rozsah slyšitelného zvuku je individuální a obvykle se s rostoucím věkem snižuje. V odborné literatuře jsou nejčastěji uváděny hodnoty od 16 Hz do 20 kHz. Zvukové frekvence nižší než 16 Hz nazýváme infrazvukem a frekvence vyšší než 20 kHz ultrazvukem.

Zvuky lze rozdělit na tóny a hluky [20]. Tóny vznikají při pravidelném, v čase periodicky probíhajícím pohybu, kmitání. Zdrojem hudebních tónů mohou být například lidské hlasivky či hudební nástroje. Jako hluky označujeme nepravidelné vlnění vznikající jako složité nepravidelné kmitání těles nebo krátké nepravidelné rozruchy jako je např. srážka dvou těles nebo výstřel. Periodické kmitání lze popsat rovnicí:

$$a = A\ sin(2\pi ft + \varphi) \tag{2.1}$$

kde a je okamžitá výchylka v čase t, A je maximální amplituda, φ fázový posun a f frekvence označující počet kmitů za jednotku času. Doba trvání jednoho kmitu je nazývána periodou T. Pro vztah periody signálu T a frekvence f platí $f = 1/T$.

Šíření zvuku ve volném prostředí probíhá všemi směry ve formě tzv. *vlnoploch* [20]. Rychlost šíření c závisí na látkovém prostředí, jeho teplotě a vlnové délce λ. Rychlost zvuku ve vzduchu při teplotě 13,6 °C je standardně 340 m/s. V reálném akustickém prostředí zvukové vlny dopadají na překážky, čímž dochází k odrazům, pohlcení nebo ohybu zvuku okolo překážky. Množství odražené akustické energie závisí na tvaru, velikosti a materiálu překážek. Úhel odrazu je vždy roven úhlu dopadu zvukové vlny.

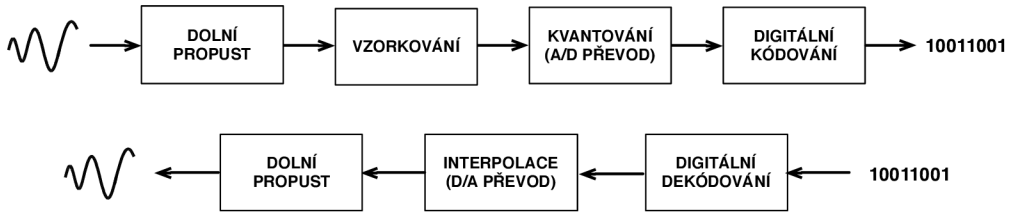
2.1.1 Digitalizace zvuku

Zpracování zvukového signálu může probíhat dvěma přístupy v závislosti na jeho reprezentaci. Analogový zvukový signál má spojitý charakter, jeho hodnota je definována v každém časovém okamžiku. Naopak digitální zvukový signál je diskrétní, tedy jeho hodnota je definována pouze v určitých časových okamžicích. Převod analogového signálu na digitální se skládá ze dvou základních kroků.

Prvním krokem je proces *vzorkování* [20], při kterém je vstupní analogový signál transformován na posloupnost periodicky zaznamenaných diskrétních vzorků. Frekvenci záznamu jednotlivých vzorků udává vzorkovací frekvence f_{vz}. Pro zpětnou rekonstrukci analogového signálu bez ztráty jeho vlastností je nutné dodržet tzv. *Shannonův teorém* [20]. Tento teorém udává minimální hodnotu vzorkovací frekvence f_{vz}, a to jako:

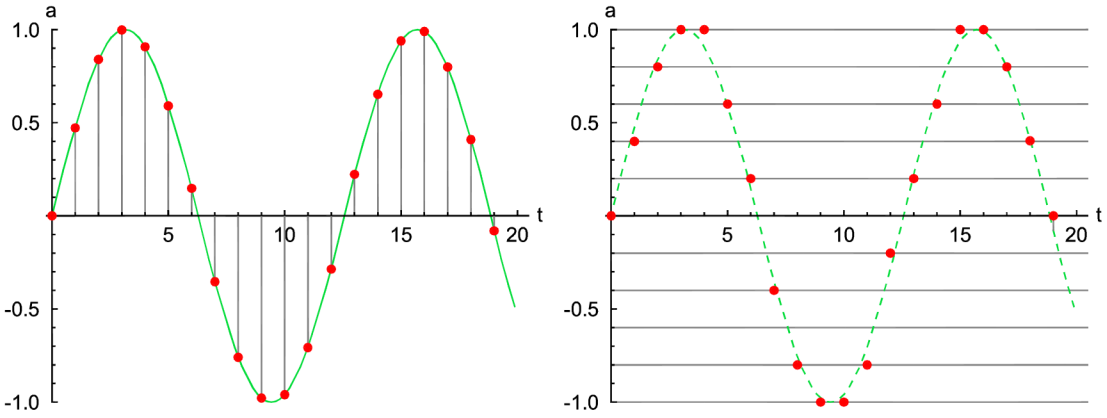
$$f_{vz} \geq 2f_{max} \tag{2.2}$$

kde f_{max} je nejvyšší frekvence obsažená ve vzorkovaném signálu. Vzhledem k faktu, že lidské ucho je schopné vnímat zvuky s maximální frekvencí 20 kHz, je standardně využívána vzorkovací frekvence 44,1 kHz pro většinu zařízení a záznamových disků. Porušením *Shannonova teorému* dojde k nenávratnému zkreslení signálu, tedy jevu nazývanému *aliasing*. Vliv *aliasingu* na výslednou reprezentaci analogového signálu lze omezit tzv. *antialiasingovým filtrem*, což je vysokofrekvenční filtr omezující vstupní analogový signál před jeho samotným vzorkováním.



Obrázek 2.1: Převod mezi analogovým a digitálním signálem [20]

Druhým krokem převodu je proces *kvantování* [20]. *Kvantování* přiřazuje získaným vzorkům vstupního signálu hodnotu jedné z kvantovacích hladin a kóduje tuto hodnotu do binární podoby. Počet bitů využitých pro uložení jednoho vzorku je označován jako *bitová hloubka*. *Bitová hloubka* udává konečnou množinu hodnot, kterých zakódovaný signál může nabývat. Získaný jednoduchý binární kód reprezentující digitální signál je pak případně nahrazen kódem, který je vhodnější pro další zpracování.



Obrázek 2.2: Vzorkování a kvantování signálu [20]

2.2 Zvukové systémy

Jako zvukový systém lze označit systém, který pracuje se zvukem, provádí jeho záznam, zpracování nebo reprodukci. Následující text popisuje základní charakteristiku zvukových systémů, které lze přesněji zařadit do kategorie *DAW*[1]. Tyto systémy nepokrývají celou oblast zvukových systémů, prezentují však rozsah těchto systémů a jejich funkce.

2.2.1 Reason

Reason [8] je hudební software orientovaný na elektronickou produkci hudby vyvinutý společností *Propellerhead*. Tento systém je jedním z virtuálních softwarových nosičů, které kombinují možnosti celé řady integrovaných komponent. Základní části tohoto systému tvoří 2 softwarové syntezátory, digitální samplery, mixery, sekvencery a mnoho efektových jednotek. Všechny tyto komponenty mohou být řízeny vestavěným *MIDI* sekvencerem nebo případně jiným systémem, který lze propojit pomocí protokolu *ReWire*. Tento softwarový protokol byl vyvinut společnostmi *Propellerhead* a *Steinberg* za účelem vzdáleného řízení a přenosu dat mezi systémy zpracovávajícími zvuk.

Základ systému je tvořen 64kanálovým *MIDI* převodníkem, 64kanálovým audio převodníkem a speciálním vstupem *ReBirth* pro připojení jiných zařízení výrobce tohoto systému [8]. Do základního panelu uvnitř systému lze pomocí virtuálních konektorů připojit libovolný počet komponent zpracovávajících zvuk, které jsou v systému dostupné. Jediným omezením je výkon počítače, na kterém tento systém běží. Pro výměnu dat, zvukových vzorků, virtuálních nástrojů a sestavení jednotlivých komponent pracuje tento systém s konfiguračním souborem.

Uživatelské rozhraní tohoto systému je navrženo tak, aby všechny komponenty systému co nejpřesněji kopírovaly vzhled skutečných zařízení. Stejně je tomu i u jejich ovládacích prvků, které fungují stejně jako u hardwarových zařízení. Některé z integrovaných komponent jsou zachyceny na obrázku 2.3.

[1] Digital Audio Workstation - elektronické zařízení nebo softwarová aplikace pro nahrávání, úpravu a produkci hudebních skladeb

Obrázek 2.3: Reason - uživatelské rozhraní integrovaných komponent [8]

2.2.2 Renoise

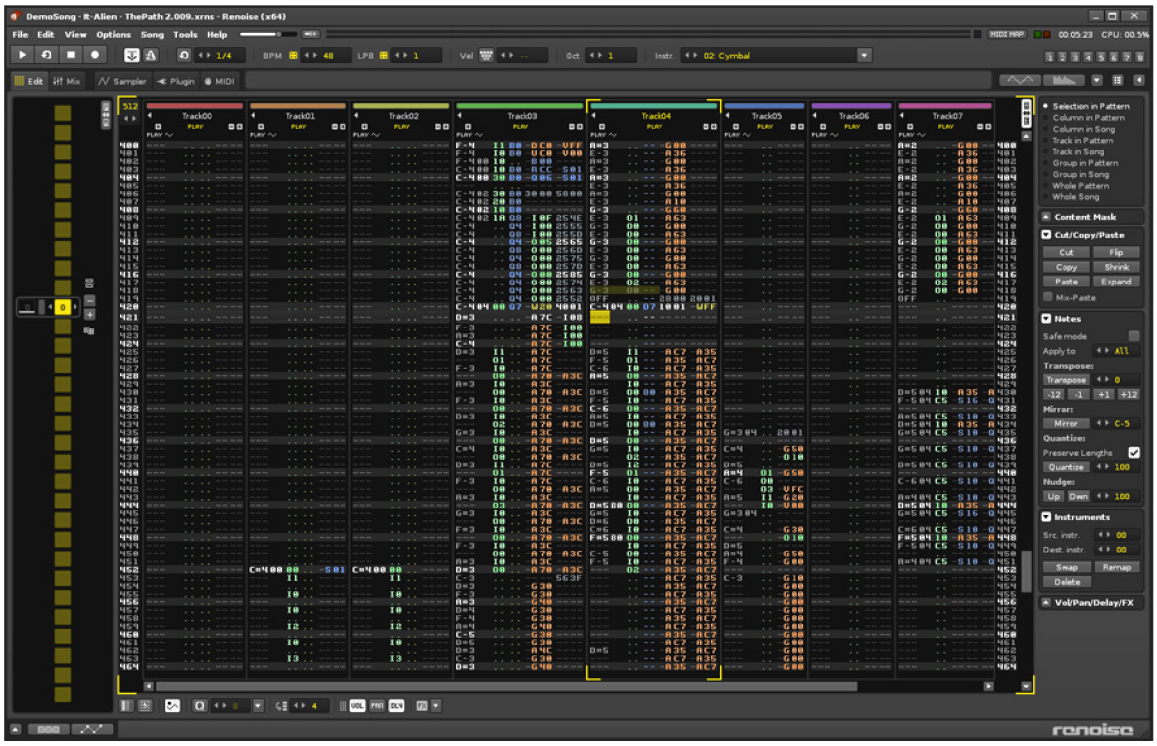
Renoise [17] je jedním z profesionálních systémů pro tvorbu, záznam a úpravu hudby. Tento zvukový systém byl vyvinut na základech hudebního trackeru[2] *NoiseTrekker*.

Jeho primární využití je orientováno na kompozici hudby prostřednictvím zvukových vzorků, syntezátorů a efektových zasuvných modulů. Hlavní charakteristikou tohoto softwaru je jeho vertikální časová osa, která vychází z hudebního trackeru. Tato časová osa je doprovázena mřížkou, která reprezentuje zvukové kanály jako sloupce a základní časové intervaly jako řádky. Vizuální podobu zvukového kanálu tvoří tzv. *track*, který obsahuje noty a jejich parametry, instrumenty[3] a další efekty. Úprava a tvorba skladby probíhá pomocí této mřížky, na které jsou zaznamenávány noty jednotlivých instrumentů.

Renoise také obsahuje mnoho zvukových procesorů pro zpracování zvukového signálu, virtuálních nástrojů a podporu pro efektové moduly mnoha formátů jako je *VST*, *AU* nebo *LADSPA*[4] [17]. Součástí tohoto systému je také možnost mixování skladeb pomocí mixážního pultu a také přímá úprava zvukových vzorků nebo frekvenční analýza hudebních skladeb.

Systém lze propojit s libovolným zařízením komunikujícím pomocí protokolu *MIDI*. Další možností komunikace je využití protokolu *OSC*. *Renoise* nabízí poměrně bohaté možnosti nastavení, které umožnují dálkově řídit jeho funkce a zpracování zvuku pomocí protokolu *OSC*. Tento systém je vyvíjen jako multiplatformní pro Linux, Windows a Mac OS X.

[2] Softwarový hudební program používaný pro tvorbu hudebních skladeb
[3] Sada zvukových vzorků, které jsou v trackeru organizovány jako jeden logický celek
[4] Virtual Studio Technology, Audio Units, Linux Audio Developer's Simple Plugin API



Obrázek 2.4: Renoise - uživatelské rozhraní pro tvorbu hudební skladby [17]

2.2.3 Logic Pro

Logic Pro [5] je sofwarovou aplikací pro záznam, úpravu a tvorbu hudby na platformě *Mac OS X*. Tento systém byl původně vyvíjen multiplatformně, a to jako *MIDI* sekvencer[5]. Tento sekvencer byl doplněn o možnost zápisu hudební notace a v roce 2002 se jeho vývoji začala věnovat společnost *Apple*.

Logic Pro poskytuje celou řadu softwarových hudebních nástrojů, zvukových efektů a nahrávací zařízení pro hudební syntézu. Pro tuto syntézu lze také využít tzv. Apple Loops, které představují krátké zvukové vzorky vytvořené za účelem jejich opakování.

Jádrem tohoto systému je také robustní mixážní konzole [5]. Umožňuje zpracovávat vícekanálový prostorový zvuk a při dostatečném výkonu počítače až 255 hudebních stop současně. Všechny prováděné akce je také možné automatizovat. Uvnitř systému je k dispozici nástroj pro záznam aktivity uživatele a umožňuje tak reprodukovat prováděné činnosti. Prostřednictvím protokolu *MIDI* je schopen komunikovat s libovolným zařízením. Při záznamu *MIDI* událostí umožňuje aplikovat nedestruktivní modifikátory na definované oblasti záznamu, a to v reálném čase.

Tento systém je také vybaven podporou pro distribuované zpracování. Při spojení s aplikací *Logic Node*, která musí být dostupná v lokální síti, umožňuje distribuovat časové náročné výpočty na více zařízení. Díky tomu je možné výrazně urychlit syntézu a zpracování efektů při produkci hudební skladby.

[5] Zařízení nebo aplikační software, který umožňuje provádět záznam, úpravu a reprodukci *MIDI* událostí
[6] Převzato z http://i1-mac.softpedia-static.com/screenshots/Logic-Pro_2.jpg

Obrázek 2.5: Logic Pro - mixážní konzole[6]

2.3 Komunikační protokoly zvukových systémů

Řízení a obecně komunikace se zvukovými systémy může probíhat v mnoha různých formách. Příkladem může být nízkoúrovňové zpracování zvukových signálů hardwarovými zařízeními za účelem syntézy zvuku, řízení elektronických hudebních nástrojů a jejich kompozice nebo také kontrola algoritmů pro zpracování zvuku v softwarové podobě.

Pro přenos informací mezi zvukovými systémy je využíváno různých komunikačních protokolů. Některé z nich lze označit téměř za standard vzhledem k jejich rozšířenosti a univerzalitě, jiné jsou naopak určeny pro popis specializované činnosti systému. V této kapitole jsou popsány pouze zvolené alternativy pro prezentaci základních mechanismů komunikace zvukových systémů.

2.3.1 MIDI

MIDI [13] je standardem v hudebním průmyslu, který popisuje protokol řízení a komunikace mezi zvukovými systémy a dalšími zařízeními. Tato technologie byla zavedena v roce 1982 a dodnes je spravována organizací *MIDI Manufacturers Association*. V průběhu 90. let se standard rozšířil i na osobní počítače, které poskytovaly jednoduché programovatelné prostředí. *MIDI* prošlo řadou aktualizací, z nichž za zmínku stojí zavedení souborových formátů pro uložení *MIDI* dat jako např. *SMF*[7], *DLS*[8] a *XMF*[9]. Dalším zlepšením bylo také zavedení standardu *General MIDI*, který zaručoval zvukovou kvalitu přehrávaných souborů a jednotnou interpretaci parametrů a řídících příkazů, které nebyly v původní definici specifikovány.

[7]Standard MIDI File
[8]Downloadable Sounds
[9]Extensible Music Format

Hlavní výhodou zavedení této technologie byla především redukce nutné kabeláže mezi jednotlivými zařízeními a s tím související menší náročnost na různé další převodníky, které zajišťovaly kompatibilitu mezi zařízeními od různých výrobců. Další výhodou je také poměrně vysoká univerzalita použití.

MIDI rozhraní je sériové a umožňuje využít až 16 kanálů pro izolování přenášených informací. Zařízení tak může reagovat pouze na zprávy na určitém kanálu a ostatní mohou být ignorovány. Komunikační zpráva je složena ze série bytů, kde první bit každého bytu určuje, zda se jedná o stavový byte identifikující typ zpráv nebo datový byte obsahující samotná data zprávy [13]. Zprávy zasílané přes toto rozhraní lze rozdělit do 2 základních kategorií.

První z nich zahrnuje systémové zprávy [13]. Tyto zprávy jsou určeny pro nastavení hodnot společných pro všechny kanály. Příkladem těchto zpráv může být volba kanálu, požadavek na naladění zařízení, výběr skladby nebo nastavení pozice ve skladbě. Součástí této kategorie jsou také tzv. *SysEx* zprávy neboli *System Exclusive* zprávy. *SysEx* zprávy představují informace, které slouží pro nastavení parametrů u specifického zařízení. Tyto zprávy nejsou nijak standardizovány, obvykle jsou definovány výrobcem daného zařízení. Tento typ zpráv se výrazně podílí na flexibilitě standardu *MIDI*. Každý výrobce zařízení, která podporují tento standard, má přidělené jednoznačné identifikační číslo, které je součástí každé *SysEx* zprávy. Na základě této identifikace pak na zprávu reagují pouze zařízení, kterým je určena.

Druhá kategorie zpráv zahrnuje kanálové zprávy [13]. Tyto zprávy přenáší informace o přehrávání not, zastavení přehrávání, změně výšky nebo ohýbání not, stisku kláves a obecně změnách hodnot parametrů daného kánalu. Kanálová zpráva má velikost 3 byty. První byte, který je nazýván stavovým, určuje typ požadavku a zvolený kanál. Zbylé 2 byty jsou datové, jejich obsah je specifický pro daný typ zprávy. Pro zprávu nesoucí informace o přehrání noty tyto byty udávají číslo noty a její sílu.

Podrobný popis jednotlivých zpráv lze nalézt v literatuře [13]. Příklad kanálové zprávy zachycuje obrázek 2.6. Jedná se o zprávu popisující zahrání noty C s maximální hlasitostí na prvním kanále.

Obrázek 2.6: Kanálová zpráva MIDI - Note On

2.3.2 OSC

OSC neboli *Open Sound Control* [22] je jedním z protokolů vytvořených výzkumným centrem *CNMAT* pro komunikaci mezi multimediálními zařízeními. Původně byl určen ke sdílení dat popisujících hudební vystoupení mezi elektronickými hudebními nástroji a dalšími zařízeními.

OSC svým návrhem realizuje architekturu klient/server [22]. Data jsou zasílána v datových jednotkách nazývaných pakety. Jakékoliv zařízení, které odesílá *OSC* pakety, je považováno za klienta. Naopak zařízení, která přijímají data, představuji servery. *OSC* je vysokoúrovňovým protokolem aplikační vrstvy, nespecifikuje tedy mechanismus přenosu

OSC paketů. Nejčastěji jsou data přenášena transportním protokolem *UDP/IP*.

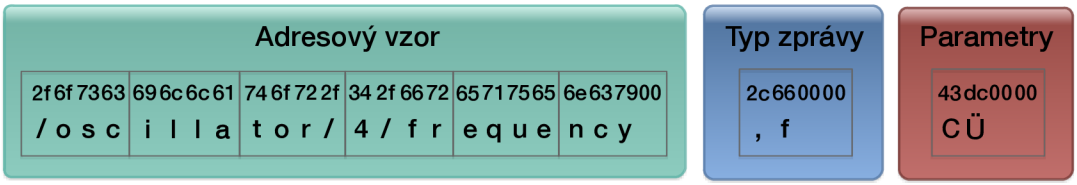
Struktura komunikačních zpráv tohoto protokolu je blízká některému ze serializačních jazyků. Každá komunikační zpráva je složena ze specifikace adresového vzoru, typu zprávy a parametrů zprávy [22].

Adresový vzor představuje adresu objektu v rámci adresového prostoru příjemce zprávy, která však může volitelně obsahovat zástupné znaky specifikující volitelnost nebo všechny objekty na dané úrovni. Každý *OSC* server definuje svůj adresový prostor jako stromovou hierarchii objektů, kterým lze zasílat zprávy. Adresa objektu je pak utvořena jako cesta od kořene stromu adresového prostoru do uzlu reprezentujícího objekt. Komponenty adresy jsou odděleny pomocí lomítka, stejně jako je tomu např. při specifikaci adresy v souborovém systému.

Typ zprávy označuje především datové typy jejich parametrů. Definice typu začíná znakem čárky následovaným řetězcem, kde každý znak určuje datový typ jednoho z parametrů komunikační zprávy. Parametry zprávy pak následují v binární podobě bezprostředně za sebou, tak jak byly specifikovány jejich typy.

Kromě komunikační zprávy umožňuje protokol také zaslat balík zpráv. Balík zpráv je identifikován klíčovým slovem „#bundle“, za kterým následuje časové razítko specifikující čas vykonání obsahu balíku a sekvence elementů balíku. Elementem může být komunikační zpráva nebo další vnořený balík zpráv. Při zpracování obsahu balíku *OSC* serverem musí být zprávy zpracovány atomicky, tedy tak jako kdyby je server obdržel v jednom okamžiku.

Obrázek 2.7 zobrazuje příklad komunikační zprávy adresované objektu *frequency* na třetí úrovni stromové hierarchie objektů s jedním parametrem s plovoucí desetinnou čárkou o hodnotě 440. Podrobnější specifikaci *OSC* lze nalézt v dokumentaci [22].



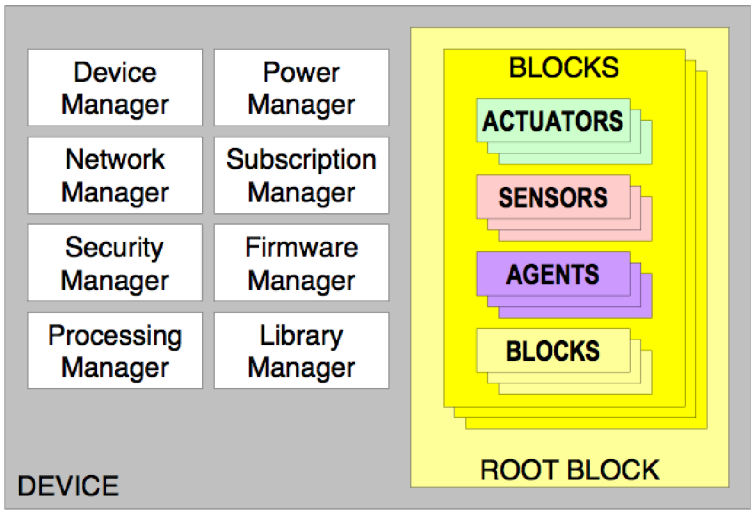
Obrázek 2.7: Komunikační zpráva OSC

2.3.3 OCA

Open Control Architecture [11] je architektura vytvořená za účelem kontroly a monitorování mediálních zařízení po síti. Neposkytuje přímo mechanismy pro transport jednotlivých signálů, ale je navržena tak, aby spolupracovala se současnými a budoucími transportními protokoly. Struktura této architektury je navržena pro podporu sítí, které kombinují zařízení od různých výrobců. Hlavním zaměřením jsou profesionální aplikace orientované na zvukové systémy. V rámci definice *OCA* lze rozlišit její 3 části.

První část *OCF* neboli *Open Control Framework* představuje architekturu balíků nástrojů a aplikačních rozhraní, které definují strukturu a mechanismy *OCA*. Součástí *OCF* je také definice tzv. modelu zařízení [11]. Tento model představuje kanonický popis kontrolního rozhraní, které zařízení komunikující pomocí *OCA* poskytuje. Součástí tohoto modelu je také objektově orientovaný popis požadovaných a volitelných objektů, které musí být zařízením v rámci rozhraní implementovány. Řídící zařízení pak následně může přistupovat k atributům objektů implementovaných v rámci tohoto rozhraní a provádět tak řízení nebo monitoring zařízení.

Druhou část tvoří hierarchická struktura tříd, které mohou zařízení instanciovat jako tzv. *OCA objekty* [11]. Tyto třídy lze dále rozdělit na tzv. *pracovníky*, kteří prezentují aplikační funkce zařízení jako například spínače, ekvalizéry, měřiče úrovně apod., *agenty*, kteří upravují a pomáhají zajišťovat kontrolní funkce, a *manažery*, kteří představují různé globální stavy zařízení. Mezi *pracovníky* se dále řadí aktuátory, senzory a bloky. Bloky mohou dále reprezentovat sdružení jednotlivých tříd do samostatných celků. Strukturu modelu zařízení v rámci architektury *OCA* zobrazuje obrázek 2.8.



Obrázek 2.8: Model zařízení architektury *OCA* [11]

Poslední část *OCA* architektury reprezentuje sada definic protokolu *OCP* [11]. Každá definice protokolu popisuje implementaci *OCA* pro konkrétní typ sítě. V současné době existuje pouze verze *OCP.1*, která se zaměřuje na implementaci pro standardní *TCP/IP* sítě. Budoucí verze protokolu plánují rozšíření např. pro propojení pomocí *USB* sítě a textovou verzi protokolu ve formátu *JSON*.

Každý *OCP* protokol definuje 3 typy komunikačních zpráv. Prvním typem jsou řídící příkazy zasílané objektům v zařízeních za účelem získání hodnoty parametru nebo provedení nějaké akce. Jako odpověď na řídící příkaz zasílá objekt zařízení informace o výsledku provedení požadované akce nebo hodnotu parametru, která byla v předchozím příkazu požadována. Posledním typem komunikační zprávy jsou upozornění, která jsou automaticky generována objekty při výskytu nějaké události nebo při periodické aktualizaci informace o hodnotě parametru.

2.3.4 Protokol DMP

DMatrixx Protocol [15] je komunikační protokol, který byl vyvinut ve spolupráci se společností DISK Multimedia, s.r.o., pro komunikaci mezi zvukovými zařízeními a především pro řízení systémů zpracování zvukových signálů v reálném čase. Původně byl navržen pro systém *Dmatrixx*, proto je označován jako *DMP*.

Základní myšlenkou tohoto protokolu je, že všechny jeho parametry jednoznačně popisují každý proces prováděný při číslicovém zpracování signálu. Každý přenášený parametr je tak specifikován identifikátory *Channel type*, *Channel group*, *Channel number*, *Parameter block*, *Parameter block index*, *Parameter type* a *Parameter type index* [15].

První 3 identifikátory se vztahují ke kanálu, na kterém probíhá zpracování signálu. Jednoznačně tedy určují typ kanálu, jeho zařazení do skupiny a číslo kanálu, které udává pořadí kanálu ve skupině. Zbývající identifikátory popisují přenášené parametry. *Parameter block* a *Parameter block index* určují blok parametrů a index bloku parametrů v případě existence více bloků stejného typu. *Parameter type* a *Parameter type index* udávají typ parametru a případně jeho index stejně jako u bloku parametrů. Příkladem bloku parametrů může být ekvalizér, kompresor nebo šumová brána, typem parametru je např. úroveň signálu, kmitočet nebo stav parametru.

Ve verzi 2.0 protokol definuje devět skupin zpráv, které reprezentují jeho možnosti [15]. Zprávy jsou děleny na komunikační a konfigurační, zprávy parametrů, zprávy mapy metrů, zprávy scén, zprávy parametrů skupin, zprávy masky a ochrany parametrů, zprávy popisů a zprávy přenosu binárních dat.

První skupina komunikačních zpráv zahrnuje všechny zprávy pro ustanovení a správu komunikace mezi serverem a klientem, zprávy informující o chybách a také pro přenos textu a obecných dat. Druhá skupina, konfigurační zprávy, obsahuje zprávy pro přenos konfigurace systému. Řízený systém poskytuje tato data o kanálech, skupinách, blocích a jejich propojení řídícímu systému, který je může využít pro identifikaci parametrů. Třetí skupina slouží pro přenos parametrů. K dispozici jsou zprávy pro různé typy přenosů parametrů v blocích nebo pro přenos více parametrů se společnou určitou částí specifikace.

Čtvrtou skupinou zpráv metrů jsou přenášeny tzv. *mapy metrů* [15]. *Mapa metrů* představuje datovou strukturu obsahující identifikátory *bodů metrů*, tj. bodů ve virtuální cestě zvukového signálu, kde je úroveň signálu měřena a vysílána řídícímu systému. Další skupina zpráv scén je využita pro ukládání, vyvolávání a správu okamžitých nastavení parametrů uložených v paměti systému. Zprávy skupin parametrů plní analogickou funkci jako zprávy parametrů, avšak jsou navrženy pro přenos informací o uživatelem definovaných skupinách.

Skupina zpráv masky a ochrany parametrů zahrnuje zprávy pro nastavení ochrany parametru proti uživatelským změnám a nastavení masek parametrů, které umožňují filtrovat změny parametrů. Pomocí skupiny zpráv popisů lze přenášet textové popisy kanálů, bloků nebo parametrů. Poslední skupina zpráv přenosu binárních dat slouží k přenosu kódů algoritmů do cílového systému, a to např. za účelem přeprogramování firmwaru přímo v řízeném systému.

Podrobnější specifikace všech typů zpráv a jejich formátu je k dispozici v [15]. Uvedené základní typy zpráv mohou být dále rozšiřovány o nové typy, které se úzce vztahují ke specializovaným zařízením.

Při zakódování parametru v protokolu *DMP* je celková délka 7 až 9 bytů podle typu dat [15]. První byte obsahuje identifikaci typu kanálu a 3 bity určující počet bitů v následujícím 16bitovém slově, ve kterém je uloženo číslo skupiny kanálů. Následuje identifikace bloku parametru a jeho indexu, typu parametru a v sedmém bytu je specifikován kromě indexu typu parametru i formát a rozsah dat.

Součástí každého datového bloku zprávy je také hlavička, která identifikuje verzi použitého protokolu, časové razítko zprávy, velikost zprávy a možné nastavení priority jednotlivých zpráv. Tato hlavička slouží především k opravě možných chyb vzniklých při přenosu, spojení rozdělených bloků zprávy, synchronizaci pořadí zpráv nebo např. doručení zpráv na základě priority v případě doručení více zpráv ve stejném okamžiku.

DMP je také transparentní z hlediska přenosu přes protokol *MIDI*. V rámci tzv. *System Exclusive* zpráv protokolu *MIDI* [13] lze přenést libovolnou zprávu protokolu *DMP*, která je přizpůsobena na 7bitový přenos.

Kapitola 3

Uživatelské rozhraní zvukových systémů

Uživatelské rozhraní zvukových systémů se promítá i do realizovaného systému, který lze považovat také za zvukový systém. Pro úpravu parametrů událostí je vhodné volit ovládací prvky, které nejlépe vystihují charakter daného parametru a poskytují uživateli nejvíce informací o zvoleném nastavení. V následujícím textu je popsáno uživatelské rozhraní zvukového systému jako softwaru, který však svým uživatelským rozhraním do značné míry odpovídá adekvátnímu hardwaru. Důraz je kladen na analýzu ovládacích prvků charakteristických pro zvukový systém a jejich možnou multiplatformní implementaci. Součástí této kapitoly je také možný popis stavu a struktury zvukového systému jako celku.

3.1 Popis uživatelského rozhraní zvukových systémů

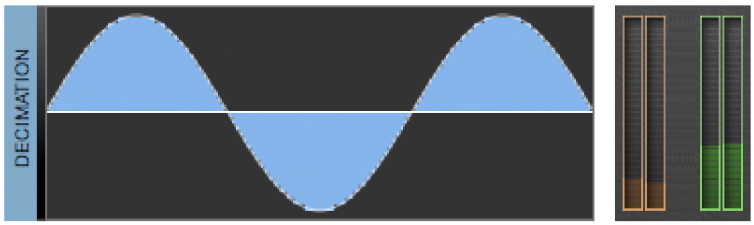
Základem aplikace pro práci se zvukem by mělo být intuitivní uživatelské rozhraní, které respektuje zvyklosti zvukařů, hudebníků a dalších, kteří obsluhují skutečná elektroakustická zařízení. Velký důraz je proto kladen na uživatelské rozhraní, které by mělo velmi přesně simulovat vzhled a funkci ovládacích prvků na skutečných zařízeních.

Systémy pro zpracování zvuku [16] se vyznačují vysokou mírou parametrizace a nutností přehledně zobrazovat mnoho stavových indikátorů. Prakticky mají i poměrně jednoduché algoritmy řádově desítky až stovky parametrů, které je nutné při provozu zařízení ovládat. Kromě standardních ovládacích prvků pro skalární veličiny je nutné zvolit prvky i pro vektorové veličiny nebo veličiny s nutností speciálního zobrazení. Povaha zobrazované veličiny, jako například indikátoru stavu, mnohdy může vyžadovat návrh speciálních ovládacích prvků.

Prvky uživatelského rozhraní lze rozdělit na pasivní a aktivní [16]. Pasivní slouží primárně k zobrazení aktuálních hodnot parametrů, aktivní naopak k úpravě hodnot parametrů. Aktivní ovládací prvky lze dále rozdělit podle toho, jakým způsobem pracují, a to na skokové a spojité. Spojité prvky mají lineární nebo např. logaritmický průběh.

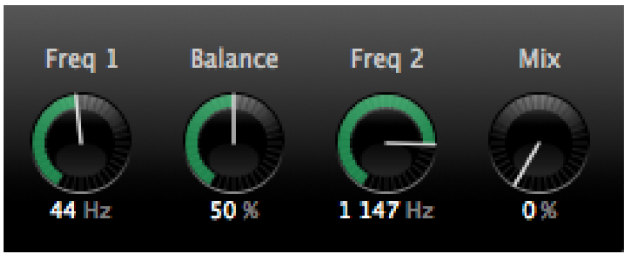
Mezi aktivní prvky lze zařadit tlačítko, přepínač, posuvník a otočný potenciometr. Z těchto aktivních prvků pracuje posuvník a otočný potenciometr spojitě. Tlačítko a přepínač pracují skokově. V případě pasivních prvků jsou nejčastěji využívány LED, graf, výpis hodnoty a sloupcový indikátor, který zpravidla zachycuje úroveň signálu.

V praxi se potenciometrem [16] ovládají parametry typu zesílení, frekvence ekvalizéru, poměr modulovaného a nemodulovaného množství signálu apod. Potenciometr je nejčastěji



Obrázek 3.1: Ukázka pasivních prvků [16]

ovládán buď fyzickými hardwarovými zařízeními, nebo standardními vstupními zařízeními jako je klávesnice a myš. Ne vždy je ovládání posuvníku umožněno i pomocí klávesnice. Z tohoto důvodu je vhodné, aby byl ovládaný potenciometr zvýrazněn a informoval tak uživatele o aktuálně prováděné akci. Řada zvukových aplikací také rozlišuje při ovládání myší mezi lineárním a kruhovým režimem. Změna hodnoty v lineárním režimu je pak prováděna kliknutím myši na prvek a pohybem myši. Velikost výchylky potenciometru je určena délkou dráhy, kterou ukazatel myši urazil v souřadnicích x a y zároveň. V případě kruhového režimu uživatel nastavuje výchylku potenciometru přímo ukazatelem myši. Mnohdy lze potenciometr ovládat i kolečkem myši.



Obrázek 3.2: Ukázka potenciometrů (Audiffex inTone Mixer 2)

Posuvník [16] je na rozdíl od potenciometru obvykle běžnou součástí systémových ovládacích prvků. Systémové ovládací prvky však lze velmi obtížně přizpůsobit těm hardwarovým, které například mohou zobrazovat velikost úrovně signálu. V rámci hardwaru je posuvník využíván například u mixážního pultu nebo grafického ekvalizéru, kde je ovládáno zesílení jednotlivých frekvenčních pásem. U softwarových aplikací nejčastěji nalézá uplatnění u přehrávačů, kde je pomocí horizontálního posuvníku signalizován ukazatel po-

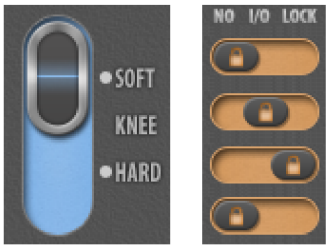
Obrázek 3.3: Ukázka posuvníků (Audified SceneFlow)

zice v přehrávaném audiu. Posuvník bývá horizontální nebo vertikální. Ovládání hlasitosti je zpravidla realizováno vertikálním posuvníkem, kdežto zmiňované přehrávání horizontálním.

Tlačítko [16] je stejně jako posuvník jedním ze standardních ovládacích prvků. Tlačítka jsou nejčastěji využita například v mixážních pultech pro utlumení kanálu nebo naopak utlumení ostatních kanálů. U přehrávačů aktivují funkce přehrávání, pauzy, zastavení nebo přetočení. Tlačítko může pracovat v režimu, kdy po stisknutí setrvá stisknuté. Pro jeho uvolnění je třeba jej znovu stisknout.

Obrázek 3.4: Ukázka tlačítek (Audified SceneFlow)

Přepínač [16] slouží k přepínaní mezi konečným počtem stavů. Může se jednat o stavy zapnuto a vypnuto, ale i jiné jako například typ zvoleného efektu. Ovládání probíhá opět pomocí myši nebo případně šipkami na klávesnici.



Obrázek 3.5: Ukázka přepínačů (Audified AutoMix, Audiffex inTone 2)

3.2 Možnosti implementace uživatelského rozhraní

Pro implementaci grafického uživatelského rozhraní existuje řada softwarových balíků. Tyto balíky lze rozdělit do 2 základních skupin.

První z nich obsahuje ovládací prvky poskytované samotným výrobcem operačního systému. Jedná se například o *MFC*[1], *WPF*[2], *Windows Forms* pro operační systém Microsoft Windows nebo již zmíněný vývoj pomocí vrstvy *Cocoa* na platforně Mac OS X [2]. Tyto balíky ovládacích prvků jsou určeny především pro běžné aplikace bez speciálních požadavků na nestandardní ovládací prvky. Jejich využití v aplikacích pracujících se zvukem je tudíž poněkud obtížnější. U výše zmíněných alternativ jsou rovněž nevýhodná jejich omezení na danou platformu.

Druhá skupina zahrnuje multiplatformní balíky pro tvorbu uživatelského rozhraní, které jsou převážně vyvíjeny třetími stranami. Mezi nejrozšířenější lze zařadit technologii *Qt* [10],

[1] Microsoft Foundation Class Library
[2] Windows Presentation Foundation

GTK+ [14], *wxWidgets* [21] a *VSTGUI* [9]. *Qt* není zaměřeno pouze na uživatelské rozhraní, ale podporuje také vývoj celých aplikací včetně síťových rohraní, 3D grafiky nebo práce s databází. Knihovna *VSTGUI* je pak více specializována pro využití v audio aplikacích, především svou množinou ovládacích prvků. *wxWidgets* zase výchází ze systémových komponent, což poskytuje snazší přiblížení vzhledu aplikace zvyklostem dané implementační platformy.

V následujícím textu budou podrobněji popsány výše zmíněné knihovny ve vztahu k implementaci uživatelského rozhraní zvukového systému zejména pro jejich možné multiplatformní využití.

3.2.1 Knihovna Qt

Knihovna *Qt* [10] je jednou z multiplatformních knihoven, které obsahují podporu nejen pro tvorbu uživatelského rozhraní, ale také celých aplikací. Knihovna je složena z několika modulů, které přinášejí různou funkčnost. Většina ovládacích prvků této knihovny vychází ze systémového vzhledu na dané platformě, který lze libovolně přizpůsobit na základě motivů definujících změny vzhledu.

V oblasti zvukových systémů jsou požadavky na vzhled ovládacích prvků mnohdy zcela specifické [16]. Z tohoto důvodu jsou ovládací prvky knihovny *Qt* často využity jako základ pro vytvoření specifických ovládacích prvků na míru pro účely cílové aplikace.

Jako nadstavba knihovny *Qt* existuje knihovna *Qwt* [19], která je specializována na uživatelské rozhraní pro technické aplikace. Řada ovládacích prvků knihovny *Qwt* je široce použitelná i pro zvukové systémy. Z aktivních ovládacích prvků je zde k dispozici potenciometr, posuvník a přepínač. Z pasivních lze zde najít především grafy, které podporují různé způsoby vykreslení. Limitujícím faktorem u knihovny *Qwt* je možnost stylování vzhledu ovládacích prvků. Pro technické aplikace je strohý vzhled dostačující, u zvukových systémů, kde je nutné vzhled do značné míry přizpůsobit reálným zařízením, může být tato vlastnost limitující.

Qt umožňuje distribuci pod duálním licencováním, a to ve formě otevřených zdrojových kódů a pro komerční využití. Díky licenci *GNU/LGPL* ji lze za jistých podmínek použít i v komerčních aplikacích. Podmínkou distribuce pro výrobce je poskytování úprav ve zdrojových kódech knihovny k dalšímu využití. Dále musí být zmíněna informace o použití knihovny a o její licenci. V neposlední řadě cílová aplikace musí používat *Qt* a jeho moduly jako dynamické knihovny. Knihovna *Qwt* je distribuována také pod licencí *GNU/LGPL*.

3.2.2 Knihovna VSTGUI

Knihovnu *VSTGUI* [9], která byla vyvinuta společností *Steinberg*, lze považovat téměř za standard mezi ovládacími prvky pro zvukové aplikace. Tato knihovna obsahuje mnoho ovládacích prvků, které mají v aplikacích pracujících se zvukem široké uplatnění. Knihovna je vytvořena v jazyce C++, umožňuje tudíž i libovolné rozšíření dostupných ovládacích prvků pomocí objektového paradigmatu. Většinu ovládacích prvků obsahuje knihovna ve více variantách.

Z aktivních ovládacích prvků využívaných ve zvukových aplikacích je to například třída *CKnob* pro klasický potenciometr, *CAnimKnob* pro potenciometr vykreslovaný pomocí sledu několika snímků, *CSlider*, *CHorizontalSlider* a *CVerticalSlider* pro posuvník, *CHorizontalSwitch*, *CVerticalSwitch* a *CRockerSwitch* pro přepínač a *COnOffButton* nebo

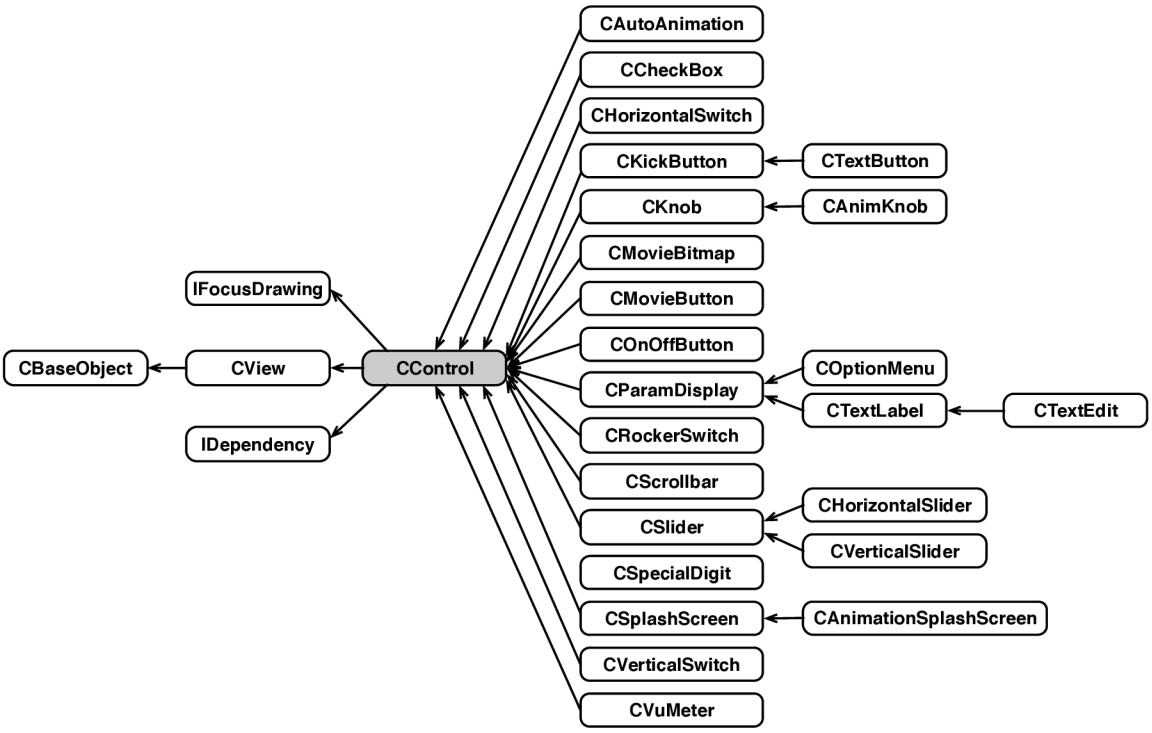
CMovieButton pro klasické tlačítko.

Z pasivních ovládacích prvků lze zmínit třídu *CVuMeter* jako indikátor úrovně signálu. Knihovna *VSTGUI* však neobsahuje ovládací prvky pro graf nebo 2D ovládací prvek.

Vzhled zmíněných ovládacích prvků lze rovněž stylovat pomocí obrázků. Příslušný ovládací prvek pak již zajistí vykreslení obrázku. Poměrně snadno lze tedy přizpůsobit výsledný vzhled cílové aplikaci. Mimo prvky uživatelského rozhraní knihovna také podporuje integraci *VST*[3], která je široce využívána pro tvorbu efektových pluginů jako zásuvných modulů zvukového systému.

Knihovna je distribuována pod vlastní licencí, která povoluje distribuci zdrojových kódů nebo případně binární formy v případě splnění jistých podmínek [9].

Podrobnější pohled na hierarchii tříd knihovny *VSTGUI* poskytuje obrázek 3.7.



Obrázek 3.6: Diagram dědičnosti tříd knihovny *VSTGUI* [9]

3.2.3 Knihovna wxWidgets

Další multiplatformní alternativou pro implementaci uživatelského rozhraní je knihovna *wxWidgets* [21]. *wxWidgets* je nadstavbou nad jinými systémovými knihovnami, která je implementována v jazyce C++. Knihovna je také přenesena do jiných jazyků, kvalita implementace již však není zaručena ze strany původního vývojáře.

Výhodou této knihovny je především úzká vazba s platformou, na které je knihovna využívána. Komponenty této knihovny přímo využívají systémové komponenty, což poskytuje standardní ovládání a vzhled specifický pro danou platfomu bez nutnosti markantního přizpůsobení vzhledu. Knihovna je orientována na prvky grafického uživatelského rozhraní,

[3]Virtual Studio Technology

obsahuje i řadu tříd a modulů pro podporu vláknového zpracování, komunikace po síti, zpracování HTML dokumentů, práce se souborovým systémem nebo také pro přístup k databázové vrstvě aplikace.

Pro vykreslení specifického vzhledu ovládacích prvků využívá přímé integrace *OpenGL*. Mimo standardní ovládací prvky, jako např. různé typy tlačítek, posuvníků a přepínačů, jsou zde také k dispozici komponenty pro práci s grafy, které lze snadno přizpůsobit požadovanému vzhledu.

Knihovna je distribuována pod licenci *GNU/LGPL*, což umožňuje její libovolné využití i v komerčních aplikacích.

3.2.4 Knihovna GTK+

GTK+ [14] neboli *The Gimp Toolkit* je dalším z multiplatformních balíků nástrojů pro implementaci grafického uživatelského rozhraní. Tento balík byl původně vytvořen pro rastrový grafický editor *GIMP*.

Knihovna byla původně vytvořena v jazyce C, dnes je však možné využít mnoho jiných programovacích jazyků pro tvorbu aplikací jako např. C++, Perl, Ruby, Java nebo také jazyk PHP. *GTK+* byla vystavěna na základě čtyř základních knihoven [14], kterými jsou knihovna *Glib*, *Pango*, *Cairo* a *ATK*.

Knihovna *Glib* tvoří základ *GTK+*. Zajišťuje především základní potřeby jako je obsluha datových struktur, rozhraní pro zpracovávání událostí, práce s vlákny, dynamické načítání nebo správa objektového systému.

Zpracování textů a zejména jejich vykreslování je zajištěno knihovnou *Pango*. Součástí této knihovny je také podpora pro vícejazyčnost.

Knihovna *Cairo* je orientována na vykreslování 2D grafických elementů. Díky této knihovně je zajištěna konzistence grafického výstupu bez ohledu na různá výstupní zařízení a využití hardwarové akcelerace.

Poslední ze základních knihoven, *ATK*, obsahuje sadu nástrojů, které usnadňují přístup k datům aplikace. Součástí těchto nástrojů je např. zvětšovací lupa nebo možnost čtení textu z obrazovky.

V rámci *GTK+* je k dispozici velká sada ovládacích prvků, ze kterých lze tvořit uživatelské rozhraní aplikace. Tyto prvky mají jednotný vzhled, čímž je sjednocen i celkový styl aplikací vyvíjených pomocí *GTK+*. Vzhled aplikace je možné snadno přizpůsobit pomocí sady nastavení a emulovat tak vzhled charakteristický pro libovolnou platformu. Knihovna ve standardní verzi neobsahuje podporu pro tvorbu grafických prvků pomocí knihovny *OpenGL*. Tento nedostatek lze však vyřešit pomocí knihoven rozšíření *GtkGLArea* a *GtkGLExt*.

GTK+ je šířena pod licencí *GNU/LGPL*, což umožňuje vývoj i komerčních aplikací.

3.3 Popis struktury a stavu zvukového systému

Pro popis zvukového systému jako celku je nutné uvažovat jistou míru abstrakce především nad množstvím jeho možných parametrů a zpracováním multimediálních signálů [16]. Díky této abstrakci je pak možné aplikovat tuto formu popisu na libovolný systém bez znalosti jeho vnitřní struktury. Na základě zvolené formy popisu musí být možné jednoznačně a úplně popsat jakýkoliv dílčí proces v systému, např. při zpracování zvukového signálu.

Na základě těchto požadavků se nabízí využít některý ze serializačních jazyků, které nejen umožní jednoznačně definovat strukturu a stav libovolného zvukového systému, ale také přenést tuto formu popisu přes libovolné komunikační rozhraní bez nutnosti dalších transformací.

V následujícím textu bude uveden příklad popisu struktury jednoho z profesionálních zvukových systémů *Audiffex inTone 2 Mixer*, jehož autorem je společnost Disk Multimedia, s.r.o. Tento software využívá pro popis stavu systému jazyk *XML* [18], přesněji se jedná o formu popisu prostřednictvím souboru *Property List*, který vychází z popisu konfigurace aplikací na platformě Mac OS X.

3.3.1 Popis pomocí XML

XML [18] je jedním z obecných značkovacích jazyků, který byl vyvinut a standardizován konsorciem *W3C*[4]. Tento jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů, u kterých popisuje strukturu z hlediska věcného obsahu jednotlivých částí a nezabývá se jejich vzhledem. Jedná se tedy o otevřenou technologii, která neslouží jen pro popis omezené množiny v současnosti používaných součástí zvukových systémů, ale naopak umožňuje jednoznačně popsat stav a strukturu libovolného objektu včetně jeho možných parametrů a atributů.

Pro zajištění jednoznačnosti z hlediska možného obsahu *XML* dokumentu a jeho struktury je rovněž využíváno jazyka *XSD* [12]. *XSD* neboli *XML Schema Definition* umožňuje formálně definovat povinné a volitelné elementy dokumentu včetně jejich struktury, datových typů, atributů a výchozích hodnot. Tento popis usnadňuje výměnu dokumentů mezi aplikacemi a také tvorbu nových dokumentů odpovídajících dané specifikaci.

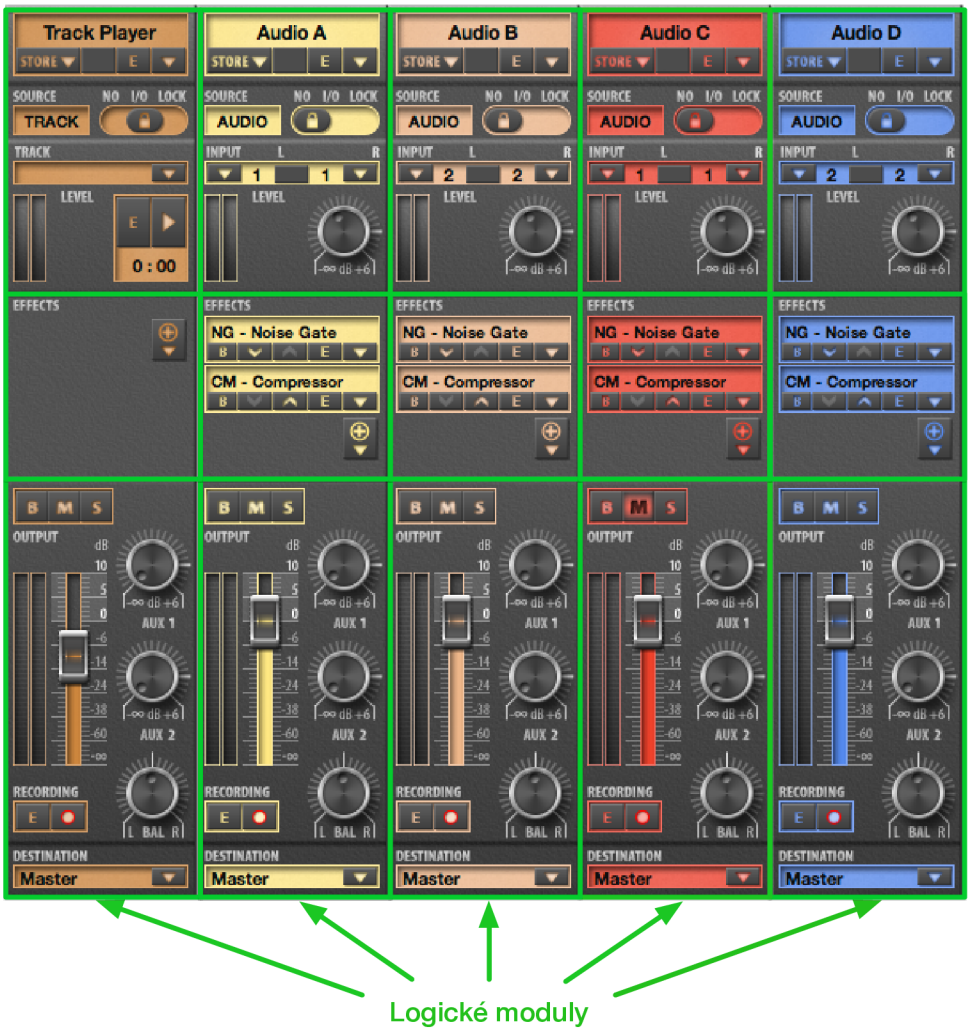
Pro dosažení co největší flexibility je popisovaný systém rozdělen na logické moduly [16]. Každý modul je reprezentován jedním uzlem v *XML* dokumentu, který je popsán unikátní adresou s kompletním popisem stavu tohoto modulu. Tyto logické moduly lze dále hierarchicky dělit na základě struktury a složitosti popisovaného systému.

Unikatní adresa každého *XML* uzlu může být tvořena více způsoby. První alternativa využívá unikátního číselného identifikátoru nebo například unikátního názvu uzlu v rámci celého systému. Tato alternativa však při přístupu k uzlu vyžaduje průchod celou hierarchií systému pro nalezení uzlu, což není příliš efektivní. Lepší variantou je vytvoření unikátního identifikátoru složením unikátních identifikátorů objektů na každé úrovni hierarchie. Uzel pak může být například identifikován pořadovými čísly modulů, ve kterých se nachází, a tímto je možné provést přímý přístup k jeho obsahu.

V rámci popisu modulu je zcela abstrahována možná implementace uživatelského rozhraní. Uživatelské rozhraní lze tedy vytvořit zcela nezávisle pro potřeby uživatele nebo jej například generovat automaticky na základě *XML* popisu logického modulu.

Pro komunikaci mezi moduly je zvolen univerzální přístup [16], který nerozlišuje mezi

[4] World Wide Web Consortium - mezinárodní konsorcium, jehož členové vyvíjejí webové standardy



Obrázek 3.7: Zvukový systém a jeho členění na moduly (Audiffex inTone Mixer 2)

místem fyzické existence modulů. Lze tedy adresovat modul, který je fyzicky součástí zvukového systému v jiné síti, stejně jako modul zvukového systému v lokální síti. Vzhledem k tomu, že stav modulu je plně popsán *XML* uzlem, je možné popsat všechny změny ve stavu modulu pomocí změn v tomto *XML* uzlu. Některé změny se týkají pouze jednoho klíče v *XML* uzlu, jiné zase vyššího počtu. Při změně konfigurace některého z logických modulů jsou okolní moduly rovněž informovány, aby mohly na změny reagovat.

Každá úroveň v popisu struktury systému je identifikována klíčem, za kterým následuje obsah dané úrovně [16]. Tímto obsahem mohou být přímo hodnoty různých datových typů jako textový řetězec, celočíselná hodnota, reálná hodnota, pole hodnot nebo také seznam klíčů vnořených úrovní společně s jejich obsahem.

Ukázka popisu struktury jednoho z logických modulů zvukového systému je zobrazena v následujícím úseku konfiguračního souboru. Jedná se pouze o zvolenou část modulu na nejnižší úrovni hierarchie. Jsou zde zobrazeny základní atributy modulu a jeden z jeho parametrů. Tento parametr může v rámci ovládacích prvků zvukového systému reprezentovat např. otočný potenciometr. V praxi je popis několikanásobně rozsáhlejší především vzhledem k velikosti popisovaného systému a množství informací, které je nutno reprezentovat.

```
<dict> <!-- Logický modul na nejnižší úrovni -->
    <key>ID</key> <!-- Atributy modulu -->
    <integer>12</integer>
    <key>Name</key>
    <string>Audio Input</string>
    <key>Parameters</key> <!-- Parametry modulu -->
    <array>
        <dict> <!-- Parametr modulu -->
            <key>Default Value</key> <!-- Výchozí hodnota -->
            <real>0.5</real>
            <key>ID</key> <!-- Identifikace -->
            <integer>0</integer>
            <key>Max Value</key> <!-- Maximum -->
            <real>1</real>
            <key>Min Value</key> <!-- Minimum -->
            <real>0</real>
            <key>Name</key> <!-- Název -->
            <string>Gain Level</string>
            <key>Step Value</key> <!-- Krok -->
            <real>0.01</real>
            <key>Units</key> <!-- Jednotka -->
            <array>
                <string>dB</string>
            </array>
            <key>Value</key> <!-- Aktuální hodnota -->
            <real>0.2505936099351</real>
        </dict>
    </array>
</dict>
```

Ukázka popisu 3.8: Úsek konfiguračního souboru popisující logický modul

Adresace *XML* uzlů jakožto logických modulů nebo jejich částí poskytuje snadný a elegantní přístup k jednotlivým ovládacím prvkům a jejich parametrům [16]. Tento způsob adresace je však prakticky využitelný pouze u softwarových alternativ systémů. U hardwarových zařízení je nutné počítat s celou řadou limitujících faktorů jako je např. omezená paměť, výpočetní výkon nebo také dostupné prostředky pro zpracování konfiguračního souboru. Pro komunikaci s hardwarovými zařízeními je tudíž nutné logickou strukturu transformovat, a to např. na jediný logický modul reprezentující celý zvukový systém, nebo provést jinou transformaci, která umožní zpracovat popis zvukového systému i v těchto zařízeních.

Kapitola 4

Operační systém Mac OS

Tato kapitola je věnována operačnímu systému Mac OS. Tento systém byl zvolen jako implementační pro realizovaný systém, protože většina zvukových systémů je realizována s ohledem na tuto platformu. Nejdříve jsou prezentovány základní vlastnosti tohoto systému a jeho rozdělení do funkčních vrstev. Dále je uvedena problematika vývoje aplikací na této platformě a charakteristické rysy jejího uživatelského rozhraní. Cílem této kapitoly je prezentovat čtenáři základní informace nutné pro realizaci systému na této platformě. Podrobnou specifikaci lze nalézt v literatuře, a to zejména v manuálech [2] a [3].

4.1 Charakteristika systému Mac OS

Mac OS je operační systém vytvořený společností *Apple*. Tento systém byl původně využíván pro počítače *Macintosh* a postupně byl nahrazován novějšími verzemi až po verzi 10, která je označována jako Mac OS X[1]. Tato verze byla vytvořena roku 2001 na bázi hybridního unixového jádra typu *XNU* společně s množstvím *BSD* a *GNU* nástrojů. Nad jádrem systému, které je souhrnně nazýváno *Darwin*, je vytvořena množina knihoven, služeb a technologií, které jsou z velké části přejaty z operačního systému *NeXTSTEP*. Vývoj Mac OS X dále pokračoval až do aktuální verze 10.10, která nese název *Yosemite*. V roce 2012 byl název systému zkrácen pouze na OS X.

OS X lze rozdělit na 5 základních vrstev [2], které zajišťují funkčnost systému a na něm běžících aplikací. Tyto vrstvy poskytují vývojářům prostředky a rozhraní pro ovládání všech funkcí zařízení. Jsou jimi vrstva *Cocoa*, *Media*, *Core Services*, *Core OS* a nejnižší vrstva jádra společně s ovladači zařízení. Jejich hierarchické uspořádání společně s hlavními balíky nástrojů, které obsahují, lze najít na obrázku 4.1.

Nejvyšší vrstva *Cocoa* [2], označována také jako aplikační vrstva, je nejvyužívanější vrstvou především kvůli vysoké míře abstrakce a jednoduchému použití, které poskytuje. Tato vrstva obsahuje nástroje pro implementaci grafického uživatelského rozhraní, interakci s uživatelem, zpracování gest a také celou řadu systémových služeb, jako například centrum upozornění, herní centrum, automatické ukládání stavu aplikací, podporu pro celoobrazovkový režim, *AppleScript*[2], *Spotlight*[3] nebo zpřístupnění systému pro hendikepované. Součástí této vrstvy je také *AppKit*, klíčový balík nástrojů pro vytváření aplikací.

Vrstva *Media* [2] poskytuje prostředky pro vytváření 2D i 3D grafických prvků, pře-

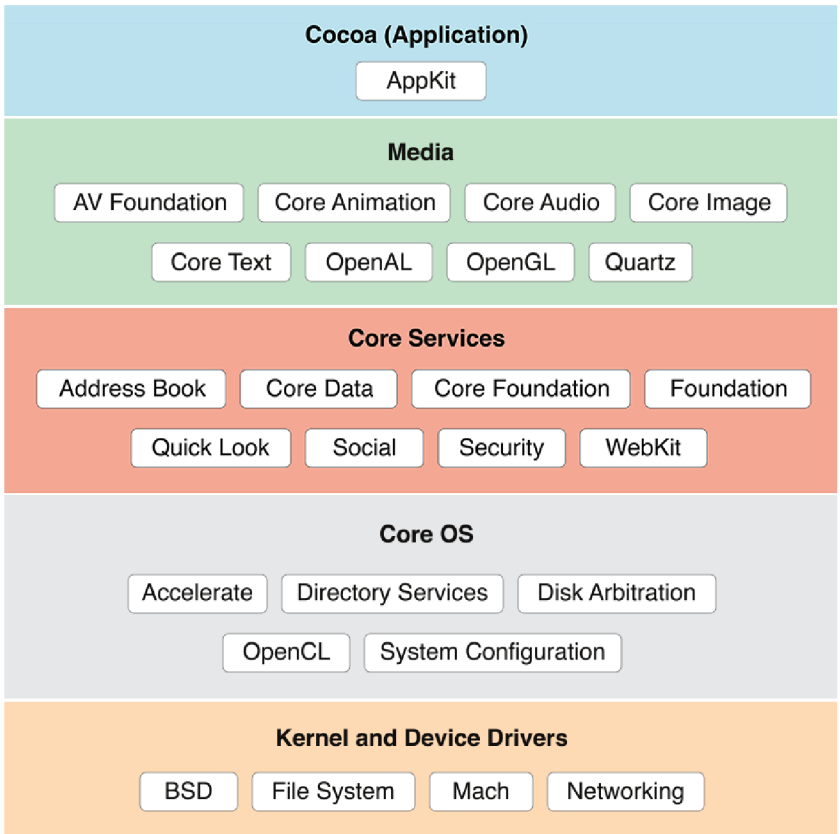
[1] http://en.wikipedia.org/wiki/History_of_OS_X
[2] Skriptovací jazyk navržený s ohledem na komunikaci s grafickým uživatelským rozhraním
[3] Vyhledávací systém na platformě Mac OS X

hrávání animací, videí nebo zvuků. Grafické prvky lze tvořit za pomocí balíku nástrojů *Quartz 2D* nebo standardu *OpenGL*. Práce se zvukem je zastoupena třemi hlavními balíky nástrojů a to *AV Foundation*, *OpenAL* a *Core Audio*.

Vrstva *Core Services* [2] nese svůj název především na základě toho, že poskytuje základní služby pro aplikace, které ale přímo neovlivňují uživatelské rozhraní. Umožňuje přístup k uživatelským účtům, napojení na sociální sítě, využití serverového úložiště, které je na platformě OS X nazýváno *iCloud*, lokalizačním a mapovým službám, vláknovému programování, nákupům v aplikaci, databázi *SQLite* pro uložení dat a nástrojům ke zjištění aktuální konfigurace zařízení nebo také zálohování pomocí *Apple Time Machine*. Neméně důležité jsou bezpečnostní služby jako autentizace uživatelů, správa certifikátů, jejich ověřování a transport. Řada těchto služeb je přímo závislá na nástrojích a technologiích nižších vrstev.

Přímá komunikace s jádrem je zajištěna pomocí vrstvy *Core OS* [2]. Tato vrstva poskytuje především nízkoúrovňové služby, které vycházejí z přímého napojení na jádro a ovladače zařízení. Služby této vrstvy jsou orientovány na bezpečnost aplikací, jako je blokování aplikací od neidentifikovatelných vývojářů, režim pískoviště aplikace nebo certifikace aplikací pro ověření jejich původu.

Poslední, a také nejnižší vrstvou, je vrstva samotného jádra. Součástí této vrstvy je řada ovladačů komunikujících s hardwarem a firmwarem. Jádro obsahuje *Mach*, který poskytuje mnoho kritických funkcí operačního systému, části *BSD*, jako model procesů a jejich identifikace, vláknové zpracování nebo komunikace po síti, a celkovou podporu a správu souborových systémů a síťové komunikace.



Obrázek 4.1: Vrstvy platformy OS X [2]

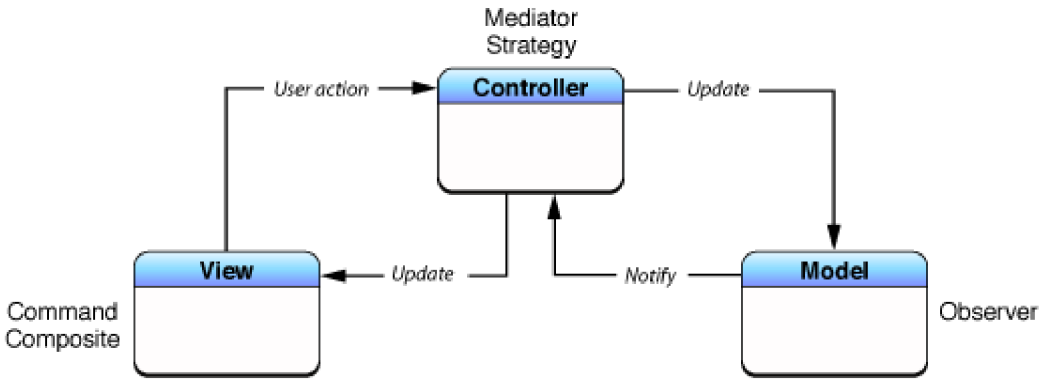
Z pohledu vývojáře jsou však veškeré nástroje a rozhraní pro přístup k funkcím zařízení na platformě Mac OS prezentovány ve formě *frameworků* neboli balíků nástrojů, podpůrných programů a aplikačních rozhraní [2]. Základními balíky nástrojů, které jsou téměř vždy využívány, jsou *AppKit* pro tvorbu uživatelského rozhraní aplikace, zpracování událostí, systému oken a ovládání se zaměřením na dotykové rozhraní a *Foundation*, který obsahuje základní třídy jazyka Objective-C pro strukturované uchování dat a podporu jednotného vývoje aplikací.

4.2 Vývoj na platformě Mac OS

Vývoj na platformě Mac OS je díky společnosti *Apple* téměř jednotný. *Apple* poskytuje sadu nástrojů pro vývoj nativních aplikací zdarma k dispozici ke stažení. Nutnou podmínkou je však vytvoření tzv. *Apple ID*, které slouží jako identifikace pro přístup ke službám společnosti *Apple*. Hlavním vývojovým nástrojem je *Xcode* [7], integrované vývojové prostředí dostupné pouze pro operační systém Mac OS X. *Xcode* obsahuje kromě všech balíků potřebných pro vývoj aplikací také precizně zpracovanou dokumentaci.

Důležitou součástí *Xcode* je také simulátor zařízení umožňující simulovat chování aplikace na zařízení s různou hardwarovou konfigurací. Pro možnost distribuovat aplikaci například do *App Store*[4] je nutné zakoupit členství ve vývojářském programu pro zařízení se systémem OS X. Poplatek činí aktuálně 99 $ ročně. *Xcode* také obsahuje nástroje pro efektivní ladění aplikací. Nejpoužívanější nástroj *Instruments* [7] poskytuje možnost provedení výkonových testů, analýzy zdrojového kódu aplikace nebo míry zdrojů využitých aplikací.

Vývoj aplikací je nativně prováděn v jazyce Objective-C [4] a nově v jazyce Swift [6], který byl představen společností *Apple*. Jazyk Objective-C vznikl jako nadstavba jazyka C spojením syntaxe jazyka Smalltalk a C. Lze jej zařadit mezi vysokoúrovňové objektově orientované jazyky. Swift představuje novou alternativu pro vývoj aplikací za použití vrstvy *Cocoa* s důrazem na co největší komfort pro vývojáře. Při vývoji lze tyto jazyky libovolně kombinovat bez nutnosti měnit strukturu aplikace. Mimo těchto standardních jazyků je také možné využít některého z jazyků rodiny C jako například C, C++ nebo Objective-C++. Vývoj v jiných programovacích jazycích je umožněn pouze externí formou, tedy při sestavení aplikace mohou být připojeny již přeložené zdrojové kódy.



Obrázek 4.2: *MVC* na platformě OS X [1]

[4]Obchod s aplikacemi pro zařízení se systémem OS X spravovaný společností *Apple*

V rámci vývoje v jazycích Objective-C a Swift je široce využívána automatická správa paměti [4]. Od roku 2011 je k dispozici plná integrace tohoto mechanismu, která zajišťuje kontrolu všech aktivních objektů překladačem. Překladač provádí analýzu zdrojového kódu a označuje objekty pro uvolnění z paměti.

Ve spojení s objektově orientovanými jazyky je také využívána koncepce *MVC* neboli *Model View Controller* [1]. Jedná se o softwarovou architekturu, která rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. *MVC* se také výraznou mírou podílí na čistotě návrhu aplikace a znovupoužitelnosti zdrojových kódů.

4.3 Uživatelské rozhraní systému Mac OS

Uživatelské rozhraní platformy Mac OS vychází především z politiky společnosti *Apple*. Jedním z hlavních důvodů jejího úspěchu je snaha o dodržení jednotného způsobu ovládání a také pravidla, že aplikace fungují tak, jak vypadají[5]. Největší důraz je kladen na jednoduchost a efektivitu ovládání aplikace. Pravidla tvorby aplikací od společnosti *Apple* jsou popsána v dokumentu *HIG*[6], který musí každý vývojář respektovat a vyvíjet aplikace v souladu s tímto soupisem pravidel. V opačném případě aplikace ve schvalovacím procesu pro distribuci v obchodu aplikací neuspěje. Tato pravidla [3] se týkají nejen rozmístění prvků na obrazovce zařízení, ale také jejich funkčnosti, sémantiky, hierarchického uspořádání, estetické celistvosti a zpětné vazby jednotlivých prvků.

Grafické uživatelské rozhraní charakteristické pro Mac OS je nazývané *Aqua* [2]. Přesněji lze takto označit vizuální styl ovládacích prvků, které tvoří základ uživatelského rozhraní aplikací. Styl *Aqua* prošel od počátků řadou změn a úprav stejně jako celý systém Mac OS X. Jeho hlavním cílem je začlenit barvu, hloubku, průsvitnost a komplexní textury do vizuálně přitažlivého rozhraní. Charakteristickým rysem *Aqua* stylu je také využití odlesků, průhlednosti a animací. Animace lze pozorovat například při spuštění aplikací, minimalizaci oken nebo při práci s dokem aplikací.

Bílá a modrá jsou dvě základní barvy, které definují styl *Aqua*. Bílá tvoří téměř veškerá pozadí oken, tlačítka a menu. Modrá zase znázorňuje posuvníky, položky menu a další ovládací prvky, které vystupují do popředí. Novější verze systému OS X nabízejí změnu základního stylu vzhledu, např. na grafitově šedou. Modré části objektů jsou pak nahrazeny zvolenou barvou.

Okno aplikace standardně obsahuje název integrovaný v horní liště. Součástí horní lišty jsou také tři tlačítka, která svým barevným vzhledem tvoří výjimky v rámci stylu *Aqua*. Tato tlačítka slouží pro uzavření okna, minimalizování do doku nebo maximalizaci do celoobrazovkového režimu.

Panely nástrojů lze charakterizovat do 2 kategorií. První zachovává klasický styl horní lišty a panel je umístěn pod horní lištu. Druhý rozšiřuje horní lištu a jednotlivé položky jsou integrovány do horní lišty, okno tedy vypadá jako by obsahovalo pouze rozšířenou horní lištu.

Prvky grafického uživatelské rozhraní, které postupně mizí z aplikací pro Mac OS X, jsou tzv. *drawers* neboli vysouvací okna. Jedná se o okna, která nemusí být nepřetržitě viditelná a jsou zobrazena vysunutím od hranic hlavního okna. *Apple* nedoporučuje využití tohoto ovládacího prvku vzhledem k aktuálně platným *HIG* [3].

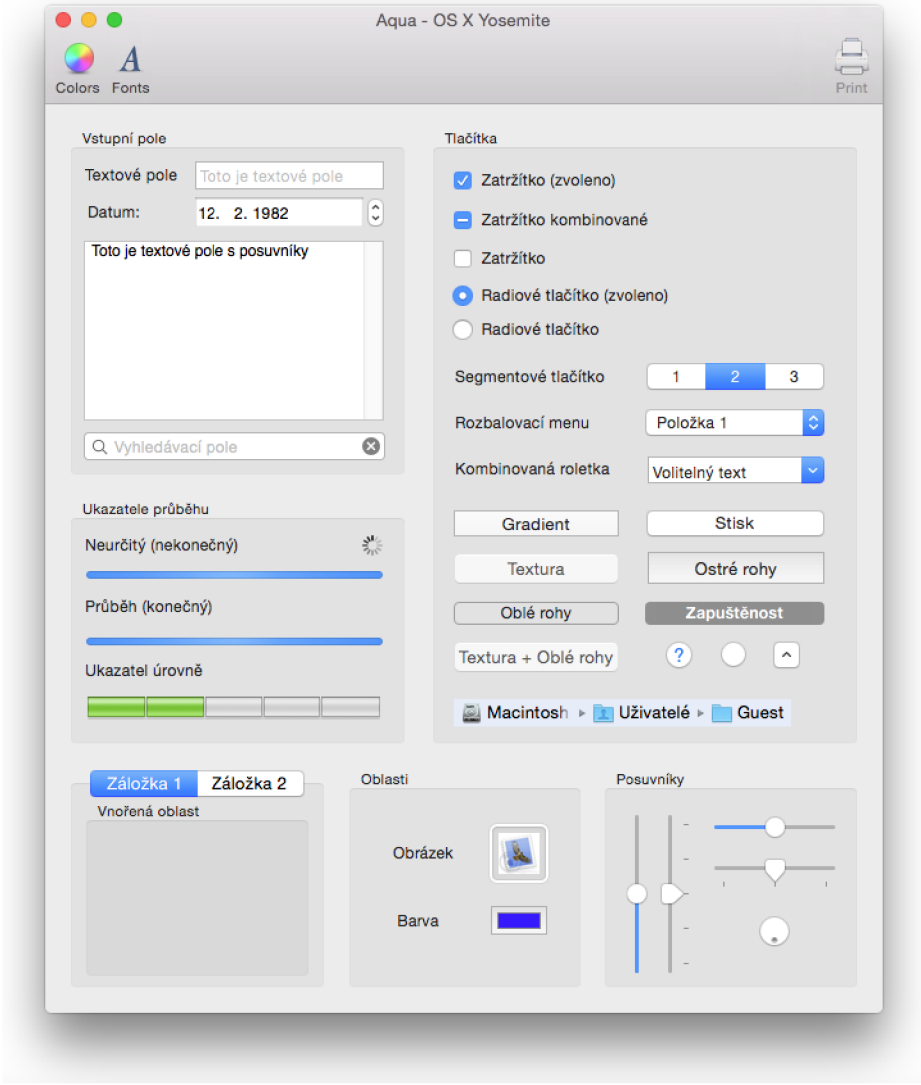
[5] V angličtině byl ustálen výraz *Form follows function*

[6] Human Interface Guidelines - soubor doporučení a pravidel tvorby a návrhu uživatelského rozhraní

Menu aplikace je obvykle integrováno do horní systémové lišty. Při stisknutí položky v menu je rozbalena doplňující nabídka této položky. *Drop-down* menu neboli rozbalovací roletka pro použití uvnitř okna je k dispozici ve dvou variantách. Standardní roletka, která umožňuje výběr jedné z přednastavených možností, obsahuje 2 šipky nasměrované protichůdně. Roletka v kombinaci s textovým polem je zobrazena téměř identicky, až na to, že obsahuje pouze jedinou šipku nasměrovanou dolů.

Za zmínku také stojí zatržítka a rádiová tlačítka. Prázdné zatržítko je zobrazeno jako bílý zaoblený čtverec. Při zatržení je čtverec vyplněn barvou, standardně vzhledem k *Aqua* stylu modrou, a na popředí čtverce je viditelné zaškrtnutí. Rádiová tlačítka mají podobný vzhled, až na kruhový tvar a tečku uprostřed kruhu, která symbolizuje zvolení tlačítka. Tato tlačítka jsou také sdružována do skupin, kdy pouze jediné je v jednom okamžiku aktivní.

Ostatní prvky uživatelského rozhraní jako klasické tlačítko, indikátory načítaní, seznamy a textová pole se úzce drží principů vzhledu, které byly popsány výše. Ukázku popsaných ovládacích prvků ve stylu *Aqua* pro OS X *Yosemite* zachycuje obrázek 4.3.



Obrázek 4.3: Ukázka stylu *Aqua* pro OS X Yosemite

Kapitola 5

Analýza současného stavu

Zvukové systémy popsané v kapitolách 2 a 3 představují poměrně velkou skupinu zařízení, která jsou rozdílná nejen svou funkcí, strukturou a uživatelským rozhraním, ale také komunikačními protokoly, které využívají pro komunikaci se svým okolím.

Softwarová podoba takového systému obvykle úzce souvisí s hardwarovým zařízením, které plní analogickou funkci. Jejich uživatelské rozhraní respektuje zvyklosti zvukařů, hudebníků a dalších, kteří obsluhují skutečná zařízení. U systémů, které se zabývají zpracováním zvuku, je nutné současně zobrazovat velké množství parametrů a stavových indikátorů. Tento fakt odráží také nutnost implementovat některé nestandardní ovládací prvky, které lépe vyhovují potřebám uživatelů a zvyšují tak přehlednost a použitelnost celého systému.

Pro implementaci uživatelského rozhraní se nabízí využít jedné z alternativ prezentovaných v kapitole 3.2. Při volbě jedné z knihoven je vhodné zvážit především zaměření systému na cílovou platformu a s tím související požadovaný vzhled uživatelského rozhraní. Důležité jsou také požadavky na specializované ovládací prvky, jejichž implementace může být poměrně obtížná. Prezentované alternativy jsou orientovány na multiplatformní implementaci systému. Pro implementaci specializovaných ovládacích prvků zvukových systémů se jeví jako nejlepší alternativa knihovna *VSTGUI*, která obsahuje některé nestandardní ovládací prvky, které mají v aplikacích pracujících se zvukem široké uplatnění. U multiplatformních knihoven však naopak může být poněkud obtížné přizpůsobit celkový vzhled aplikace charakteristice zvolené platformy.

Popis struktury a stavu zvukového systému vyžaduje jistou míru abstrakce díky různorodosti jednotlivých systémů a množstvím jejich možných parametrů. Příklad popisu z kapitoly 3.3 pomocí *XML* se zakládá především na hierarchické dekompozici systému na moduly, která může významně usnadnit přístup k parametrům rozsáhlejších systémů. Výhodou popisu pomocí *XML* je možnost formální definice schéma dokumentu za pomocí jazyka *XSD*. U rozsáhlejších systémů je popis v jazyce *XML* poměrně rozsáhlý, což při přenosu přes komunikační médium může působit problémy. Alternativou může být využití některého z jiných serializačních jazyků jako jsou např. *JSON* nebo *YAML*, u kterých bude velikost popisu znatelně menší.

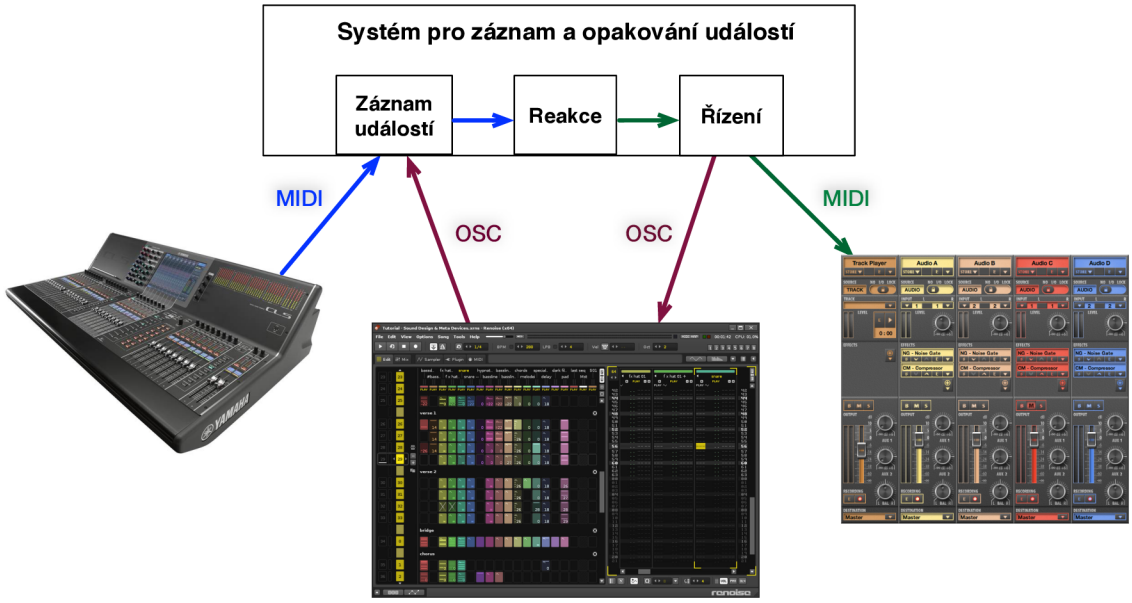
Komunikační protokoly zvukových systémů, jejichž alternativy byly popsány v kapitole 2.3, zapouzdřují funkce zvukových systémů, kterými jsou využívány. Adresování v rámci každého systému nebo zařízení je realizováno interními mechanismy, které jsou v každém protokolu odlišné. Většina protokolů však respektuje možnost hierarchického dělení zařízení na logické části, a to nejčastěji ve formě stromové struktury. Některé z protokolů, jako např. *OCP* a *OSC*, jsou navrženy jako vysokoúrovňové a nespecifikují tedy transportní médium, po kterém budou zasílány, což může být z hlediska implementace těchto protokolů výhodou.

5.1 Specifikace systému

Před začátkem návrhu systému bylo vhodné stanovit požadavky na jeho vzhled a především funkčnost. Specifikace těchto vlastností zahrnuje nejen požadavky na funkce systému, ale také některé parametry jeho realizace. Tyto vlastnosti byly stanoveny na základě předchozí analýzy zvukových systémů a odborných konzultací se společností DISK Multimedia, s.r.o. Požadované vlastnosti systému lze shrnout do následujících bodů:

- Úprava a tvorba událostí pro zvukové systémy
- Záznam vstupních událostí od zvukových systémů
- Možnost plánování a opakování událostí pro zvukové systémy
- Schopnost reakce na vstupní události od zvukových systémů
- Podpora komunikačních protokolů *MIDI* a *OSC*
- Schopnost pracovat s libovolným zvukovým systémem
- Snadná rozšiřitelnost systému o komunikační protokoly
- Implementace systému na platformě Mac OS X

Specifikované vlastnosti systému vychází ze 3 základních funkcí, kterými jsou záznam událostí, reakce na vstupní události a řízení zvukových systémů. Tyto funkce by měl systém zajišťovat současně a bez nutnosti neustálé pozornosti uživatele. Za tímto účelem bylo vhodné navrhnout do systému mechanismus, kterým lze řídit činnosti těchto základních funkcí. Navržený koncept událostí v kapitole 6.2 prezentuje možné řešení tohoto požadavku pomocí interních událostí.



Obrázek 5.1: Příklad komunikace systému se zvukovými systémy[1]

[1]Převzato z http://www.king.sk/img/gal/m/2993.jpg, [17], Audiffex inTone Mixer 2

5.2 Popis specifikovaných vlastností systému

První z vlastností specifikuje možnost úpravy a tvorby událostí. Systém by měl poskytovat rozhraní pro úpravu parametrů události, ať už se jedná o událost vstupní nebo událost vytvořenou v systému.

Druhou základní úlohou systému je záznam vstupních událostí od zvukových systémů. Příchozí události systém zaznamená a zobrazí jejich vlastnosti uživateli. Tyto události může systém také filtrovat na základě specifikace jejich vlastností a je schopen je dále reprodukovat a plánovat jejich vykonání.

Další vlastnost související s automatizací procesu řízení je mechanismus plánování událostí. Události je možné naplánovat na stanovený čas do budoucna a také provádět jejich opakováné zpracování ve stanovených časových intervalech.

Mimo záznam vstupních událostí by systém měl mít schopnost reakce na vstupní události. Tyto reakce definuje uživatel jako specifikaci vlastností vstupního podnětu a reakce ve formě provedení nové události nebo jiné interní akce. Systém na základě těchto reakcí může např. dynamicky modifikovat aktuálně vykonávaný plán událostí a měnit tak jeho provádění vzhledem k aktuálním potřebám.

Důležitou vlastností celého systému by měla být jeho rozšiřitelnost. Mnoho výrobců zvukových systémů využívá různých formátů příkazů, různých komunikačních protokolů a každý systém je specifický svou strukturou, a tedy i množinou ovladatelných parametrů. Tuto skutečnost by měl systém zohledňovat a umožnit tak jeho modulární rozšíření pro různá zařízení a jejich potřeby. Z této vlastnosti také plyne jistá abstrakce činnosti systému nad komunikačními protokoly, a tedy řídícími příkazy. Z tohoto důvodu systém pracuje s událostmi reprezentujícími řídící příkazy, které budou zajišťovat potřebnou míru abstrakce a rozšiřitelnosti nezávisle na zvukovém systému a jeho komunikačním protokolu.

Součástí systému by také měla být databáze pro uložení definic událostí a celých plánů událostí. Plánem událostí je charakterizována sekvence událostí včetně jejich pořadí, ve kterém budou vykonávány, a časový údaj reprezentující stav vykonávání plánu. Využití této databáze také usnadňuje tvorbu nových událostí a obsluhu systému pro uživatele.

Pro realizaci systému byl zvolen operační systém Mac OS X, a to vzhledem k řadě existujících zvukových systémů na této platformě. Pro implementaci uživatelského rozhraní systému bylo využito vrstvy *Cocoa*, která na platformě Mac OS X poskytuje prostředky pro tvorbu nativního uživatelského rozhraní. Vrstva *Cocoa* byla vhodnou volbou pro dosažení vzhledu implementovaného systému, který je charakteristický pro cílovou platformu.

Z komunikačních protokolů systém v základní implementaci podporuje protokoly *MIDI* a *OSC*. Protokoly *MIDI* a *OSC* byly zvoleny vzhledem k jejich rozšířenosti a univerzálnímu využití napříč mnoha zvukovými systémy. Mimo tyto protokoly byl systém navržen tak, aby byl rozšiřitelný o další komunikační protokoly.

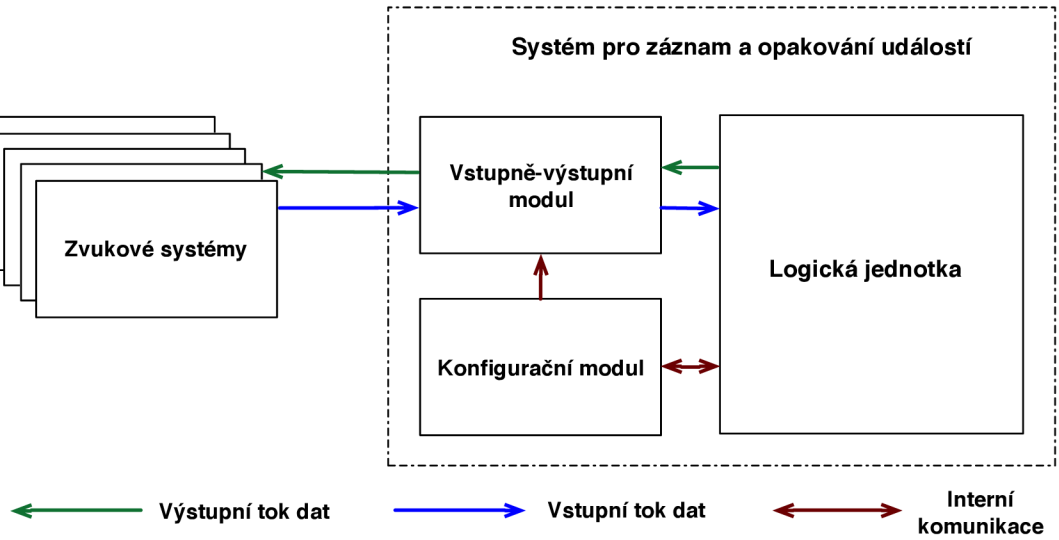
Kapitola 6

Návrh systému

Tato kapitola si klade za cíl představit návrh systému pro záznam a opakování událostí pro zvukové systémy. Nejdříve je prezentováno jeho schéma definující základní funkční bloky. Dále je uveden podrobnější popis jednotlivých částí systému a návrh reprezentace událostí v systému. Návrhovaný koncept událostí také ukazuje jejich možné zpracování a využití při reakci systému na vstupní události. Součástí této kapitoly je i návrh uživatelského rozhraní systému. Návrh je vytvořen s jistou mírou abstrakce, aby byl případně implementovatelný nezávisle na cílové platformě. Navrhovaný systém pro záznam a opakování událostí je nazýván také jako řídící systém.

6.1 Struktura systému

Cílem návrhu systému, tak jak jej zachycuje obrázek 6.1, je prezentovat model řízení zvukových systémů pomocí událostí a jeho možnou rozšiřitelnost bez nutnosti větších zásahů do struktury systému. Tento návrh je zaměřen na rozdělení systému do funkčních bloků a jejich vzájemnou interakci.

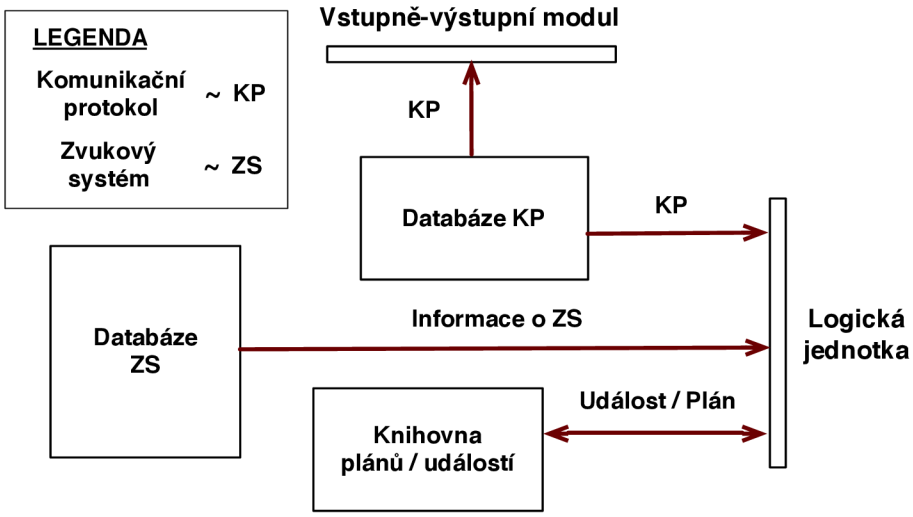


Obrázek 6.1: Rozdělení systému na základní funkční bloky

Řídící systém je rozdělen do 3 logických částí, které spolu úzce spolupracují. Toto rozdělení vychází ze specifikace požadovaných vlastností systému z kapitoly 5.1. *Vstupně-výstupní modul* systému modeluje především abstrakci událostí od komunikačních protokolů, a tedy nezávislost na řízeném zvukovém systému. Jako datová základna systému slouží *konfigurační modul*, který spravuje specifické informace pro daná zařízení a jejich komunikační protokoly. Pro rozšíření řídícího systému tak postačuje pouze modifikovat datové záznamy v této oblasti systému. Požadované vlastnosti související se záznamem vstupních událostí, procesem plánování událostí a jejich zpracováním zajišťuje *logická jednotka*. Tato jednotka kontroluje veškeré akce, které systém provádí, a automatizuje tak celý proces řízení.

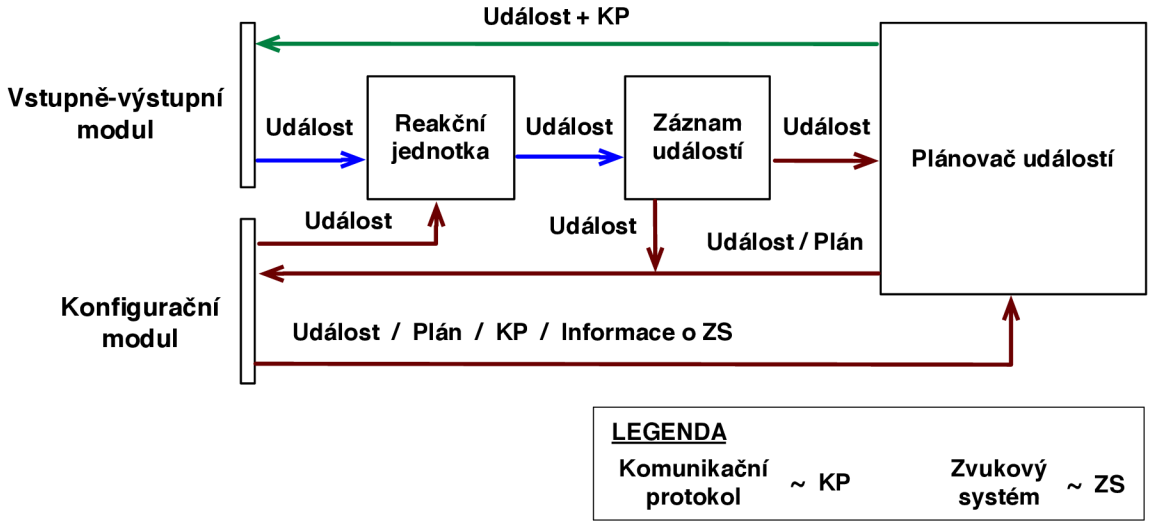
Před zahájením komunikace se zvukovým systémem je nutné zjistit množinu jeho ovladatelných parametrů a komunikační protokol, který je systémem podporován. Množinu ovladatelných částí zvukového systému je tudíž vhodné popsat konfiguračním souborem, např. tak jak byl specifikován v kapitole 3.3 pomocí jazyka *XML*. Kromě popisu ovladatelných parametrů zahrnuje tento konfigurační soubor také informace o podporovaných komunikačních protokolech. V případě zvukového systému s podporou více komunikačních protokolů uživatel řídícího systému při definici události zvolí, který bude využit pro komunikaci.

Správu těchto informací o zvukových systémech bude zajišťovat *konfigurační modul* řídícího systému. Hlavním úkolem tohoto modulu bude mimo uložení informací o zvukových systémech také správa databáze komunikačních protokolů a knihovny plánů a událostí. Databáze komunikačních protokolů označuje množinu protokolů, jejichž komunikační zprávy je řídící systém schopen vysílat a také zpracovávat. Využití této databáze probíhá především při analýze vstupních zpráv a také transformaci událostí na komunikační zprávy. Knihovna plánů a událostí zde slouží jako uložiště definovaných událostí nebo jejich plánů uživatelem a také jako úložiště pro zaznamenané vstupní události. Plán událostí zapouzdřuje sekvenci událostí v pořadí jejich plánovaného vykonání a časový údaj reprezentující stav vykonávání plánu. Díky této knihovně uživatel nemusí vždy událost specifikovat znovu a může zvolit nebo případně upravit událost definovanou v této knihovně. Data uchovávaná v částech *konfiguračního modulu* jsou dále distribuována do ostatních částí systému.



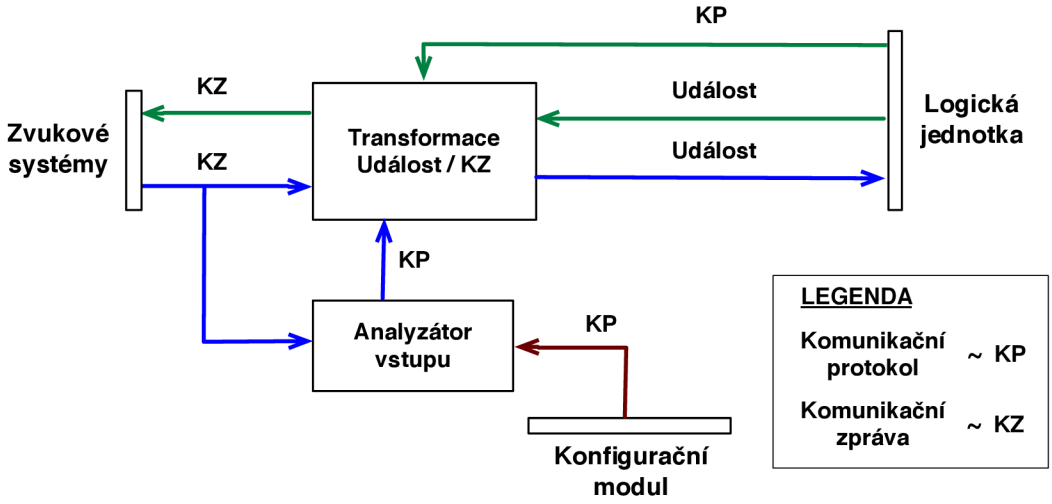
Obrázek 6.2: Konfigurační modul systému

Jádro řídícího systému tvoří *logická jednotka*. Úkolem této jednotky je především kontrolovat, plánovat a řídit provedení plánovaných událostí. Kromě plánovače událostí, který zajišťuje vykonávání plánů událostí, je zde také blok provádějící záznam vstupních událostí a jejich analýzu. Pokud vstupní událost odpovídá definovaným parametrům pro záznam událostí, pak je zařazena do sekvence reprezentované plánem událostí a uložena v *konfiguračním modulu* systému. Na vstupní událost může systém také reagovat, pokud je pro přijatou událost nalezena uživatelem definovaná reakce. Tato reakce může představovat vznik nové události, která je začleněna do aktivního plánu událostí. Aktivní plán událostí představuje plán, který je momentálně zpracováván plánovačem událostí. Plánovač událostí pracuje s interní reprezentací události, tak jak je definována v kapitole 6.2. Součástí každého plánu událostí je mimo reprezentaci události také její komunikační protokol. Při vykonávání plánu událostí je událost předána na výstup pro další zpracování společně se zvoleným komunikačním protokolem.



Obrázek 6.3: Logická jednotka systému

Poslední část řídícího systému tvoří modul, který provádí zpracování vstupních a výstupních událostí. Tento modul realizuje především transformaci přijatých komunikačních zpráv na interní reprezentaci události, která je nezávislá na komunikačním protokolu, a naopak převod interní reprezentace události na zprávu zvoleného komunikačního protokolu. Po přijetí zprávy je nejdříve nutné určit její komunikační protokol. Zpráva je analyzována za pomocí databáze komunikačních protokolů, které jsou systémem podporovány. Po zjištění komunikačního protokolu je možné extrahovat z komunikační zprávy její data. Tato data jsou pak doplněna o informace pro interní zpracování a upravena do formy reprezentace události. Takto zpracovaná vstupní událost je pak předána k další analýze *logické jednotce*. V procesu převodu výstupní událostí je proces opačný s tím rozdílem, že při převodu dat na komunikační zprávu je využito zvoleného komunikačního protokolu a není nutné komunikační protokol analyzovat.



Obrázek 6.4: Vstupně-výstupní modul systému

6.2 Události v systému

Za událost se obecně považuje určitý děj během činnosti systému. Pro účely navrhovaného systému lze definovat událost jako abstraktní jednotku, která zapouzdřuje nejen události vstupní generované zvukovými systémy, ale také události jako řídící příkazy definované uživatelem navrhovaného systému nebo jiné interní akce, které mají vliv na stav a chování systému.

Tato reprezentace ve formě událostí přináší několik výhod. První z nich je především nezávislost na komunikačním protokolu zvukového systému a potřebná míra abstrakce pro snadné rozšíření systému o nové události. Další výhodou je také možnost integrovat do této formy interní akce, které mohou dynamicky ovlivňovat proces řízení, nebo jiné požadavky uživatele. Manipulace s událostmi také může usnadnit proces řízení pro uživatele, vzhledem k jednotné reprezentaci děje v systému.

Specifikaci událostí je vhodné rozdělit na povinnou a volitelnou část. Povinná část události reprezentuje jednoznačnou identifikaci události v systému a také parametry nezbytné pro její zpracování. Volitelná část události umožňuje uživateli definovat doplňující parametry události, které lze případně nahradit výchozími hodnotami.

Analýzou vlastností komunikačních protokolů zvukových systémů, které byly popsány v kapitole 2.3, jsem stanovil 6 základních parametrů reprezentujících událost. První 4 parametry, kterými jsou identifikace, čas, cíl a operace, představují povinnou část, která musí být pro zpracování události definována. Zbývající 2 parametry definující opakování události a její prioritu jsou v rámci specifikace události nepovinné.

Identifikace události je určena jednak unikátním identifikátorem v systému, který může být například číselný, a také informačním názvem události, který specifikuje uživatel. Na základě těchto údajů je pak událost uložena v knihovně událostí a je možné se na ni odkazovat v rámci celého systému. Součástí identifikace je také specifikace typu události. Typ události zde pouze určuje mechanismus zpracování dané události.

Čas vykonání události je úzce spojen především s plánem událostí, jehož je daná událost součástí. Vzhledem k časovým intervalům, po které je nutné zvukové systémy řídit, je časový údaj určen relativně vzhledem k začátku vykonávání plánu událostí. Pokud je událost dynamicky přidána do již aktivního plánu událostí, pak čas události udává dobu

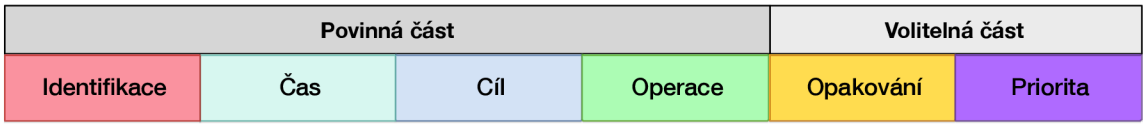
od aktuálního času plánu do jejího zpracování.

Určení systému, kterému je událost adresována, specifikuje cíl události. Adresa cíle souvisí se strukturou cílového systému a také množinou jeho ovladatelných částí. Na základě těchto informací může být adresa hierarchická, a lze tak specifikovat libovolné zařízení, jeho modul, objekt nebo parametr. Na nejvyšší úrovni hierachie adresy však vždy bude obsažena informace identifikující cílový systém.

Požadovanou akci, která má být provedena v cílovém systému, definuje parametr operace. Operací může být požadavek na hodnotu parametru, provedení změny konfigurace zařízení nebo také informace popisující určitý děj. Součástí operace jsou také její atributy, které upřesňují její vlastnosti. Ve specifických případech mohou být atributy operace nevyužity.

Nepovinný parametr opakování je ve výchozím stavu uvažován jako nulový. Prostřednictvím tohoto parametru je možné nastavit počet opakování události a také interval, ve kterém má být událost znovu spuštěna. Speciální možností tohoto parametru je také nekonečné opakování události.

Posledním a také nepovinným parametrem je priorita události. Priorita zde slouží především pro řešení konfliktů v plánu událostí, kdy je více událostí naplánováno na stejný čas, a také při nalezení více odpovídajících reakcí na stejnou vstupní událost.



Obrázek 6.5: Struktura definice události v systému

6.2.1 Typy událostí

Události lze rozdělit do kategorií na základě jejich funkce a zaměření. Každá kategorie může obsahovat libovolně rozšiřitelnou množinu událostí, která bude specifická pro potřeby daného systému. Pro účely navrhovaného systému lze definovat 2 základní kategorie, a to události řídící a interní.

Interní události jsou vykonávány uvnitř systému a ovlivňují tak stav a chování pouze u řídícího systému. Pomocí těchto událostí je možné dynamicky měnit chování systému např. na základě vstupní události, na kterou systém reaguje provedením interní události. Příkladem takové události může být modifikace aktivního plánu událostí zrušením naplánované události nebo naopak naplánováním nové události, uložení zaznamenaných událostí do knihovny nebo změna aktivního plánu událostí. Tyto události nejsou při jejich zpracování transformovány na komunikační zprávy, protože jsou vždy adresovány řídícímu systému.

Řídící události představují základní příkazy pro řízení zvukových systémů. Množina těchto událostí je tudíž do značné míry určena strukturou zvukových systémů a jejich parametry, které je možné ovládat. Události tohoto typu jsou při jejich vykonání transformovány na komunikační zprávy a zasílány cílovým systémům. Do této kategorie jsou řazeny i vstupní události generované zvukovými systémy.

Zmíněné dělení do kategorií slouží spíše pro lepší orientaci v systému. Události každé kategorie jsou specifikovány v souladu se zmíněnými parametry události v kapitole 6.2. V případě potřeby lze vytvořit další kategorie a manipulovat s jejich událostmi jednotně bez ohledu na jejich zařazení.

6.2.2 Zpracování událostí

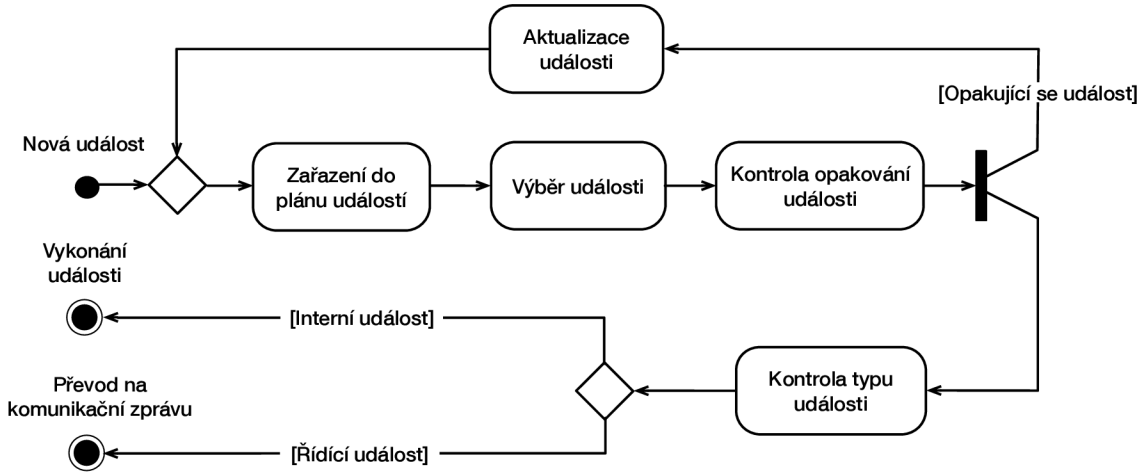
Události jsou v systému uchovávány buďto v rámci konfiguračního modulu v knihovně plánů a událostí, kde jsou uloženy v databázi jako neaktivní, nebo mohou být součástí právě vykonávaného plánu událostí, se kterým pracuje struktura plánovače událostí.

Datová struktura plánovače událostí modeluje frontu, která však respektuje priority událostí a umožnuje tak řazení na základě více kritérií. Události jsou v rámci této struktury řazeny primárně na základě specifikace času jejich vykonání. Sekundárním kritériem je pak zmíněná priorita události definující uspořádání událostí naplánovaných na stejný čas. Součástí této struktury je také časovač udávající čas od spuštění aktivního plánu událostí. Na základě tohoto časovače je prováděn výběr událostí z aktivního plánu.

Při vzniku nové události, která má být zpracována, je tato událost zařazena do aktivního plánu událostí. Vznik nové události iniciuje uživatel v rámci interakce se systémem nebo může jít o reakci na vstupní událost, která dává podnět pro vznik nové události.

Jakmile nastane čas události na začátku fronty, pak je tato událost vybrána z aktivního plánu a provedeno její zpracování. Prvním krokem při zpracování události je kontrola jejího opakování. Pokud má událost nastavený počet opakování na více než 1, pak je tato událost znovu zařazena do plánu událostí. Před znovunaplánováním události je snížen počet budoucích opakování a upraven čas do vykonání události, který odpovídá intervalu mezi opakováním události.

Událost je dále analyzována na základě jejího typu. Pokud se jedná o událost interní, pak je událost vykonána na základě požadované operace. V případě řídící události je událost předána k dalšímu zpracování vstupně-výstupnímu modulu, který zajistí převod události na komunikační zprávu, která je následně zaslána zvukovému systému.

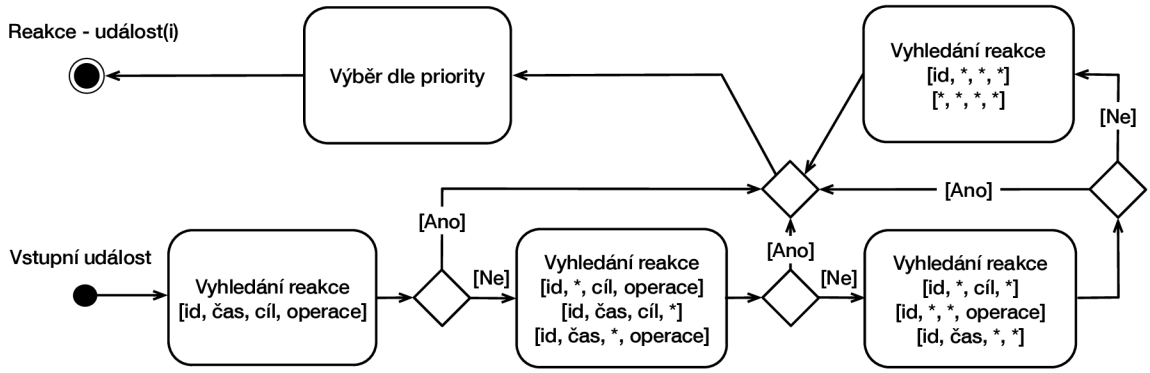


Obrázek 6.6: Životní cyklus události v plánovači událostí

Po příchodu komunikační zprávy od zvukového systému je zpráva transformována na událost. Identifikace události je utvořena na základě zdrojového systému, který zprávu zaslal. Čas události je nastaven na aktuální čas systému pro vyhledání reakce na tuto událost, při záznamu události je časový údaj nastaven na čas záznamu události, který udává relativní čas od spuštění záznamu událostí. Cíl zahrnuje zdrojovou entitu v komunikační zprávě a operace požadovanou akci nebo datovou informaci.

Na tuto vstupní událost je vyhledána odpověď v seznamu reakcí. Seznam reakcí představuje kolekci dvojic, kde první prvek je tvořen specifikací vstupní události, a druhý prvek definuje reakci ve formě události. Specifikace vstupní události nemusí obsahovat všechny povinné parametry, událost v tomto případě pouze představuje formu šablony. Vyhledání v seznamu reakcí probíhá od nejvíce specifického popisu vstupní události po nejobecnější.

Nejdříve jsou tedy vyhledány šablony událostí, které odpovídají definované identifikaci, cíli, času a operaci. Pokud není nalezena žádná odpovídající šablona, pak jsou postupně vypuštěny části specifikace vstupní události v pořadí od času, operace, cíle až po identifikaci a vyhledání je provedeno znovu, dokud není nalezena alespoň jedna šablona nebo jsou vyčerpány všechny kombinace odpovídající specifikaci vstupní události. V případě nalezení více šablon je využito specifikované priority události, která tvoří společně se šablonou vstupní události reakci na vstupní událost. Pokud jsou priority shodné nebo nejsou definovány, pak jsou zvoleny všechny nalezené reakce.



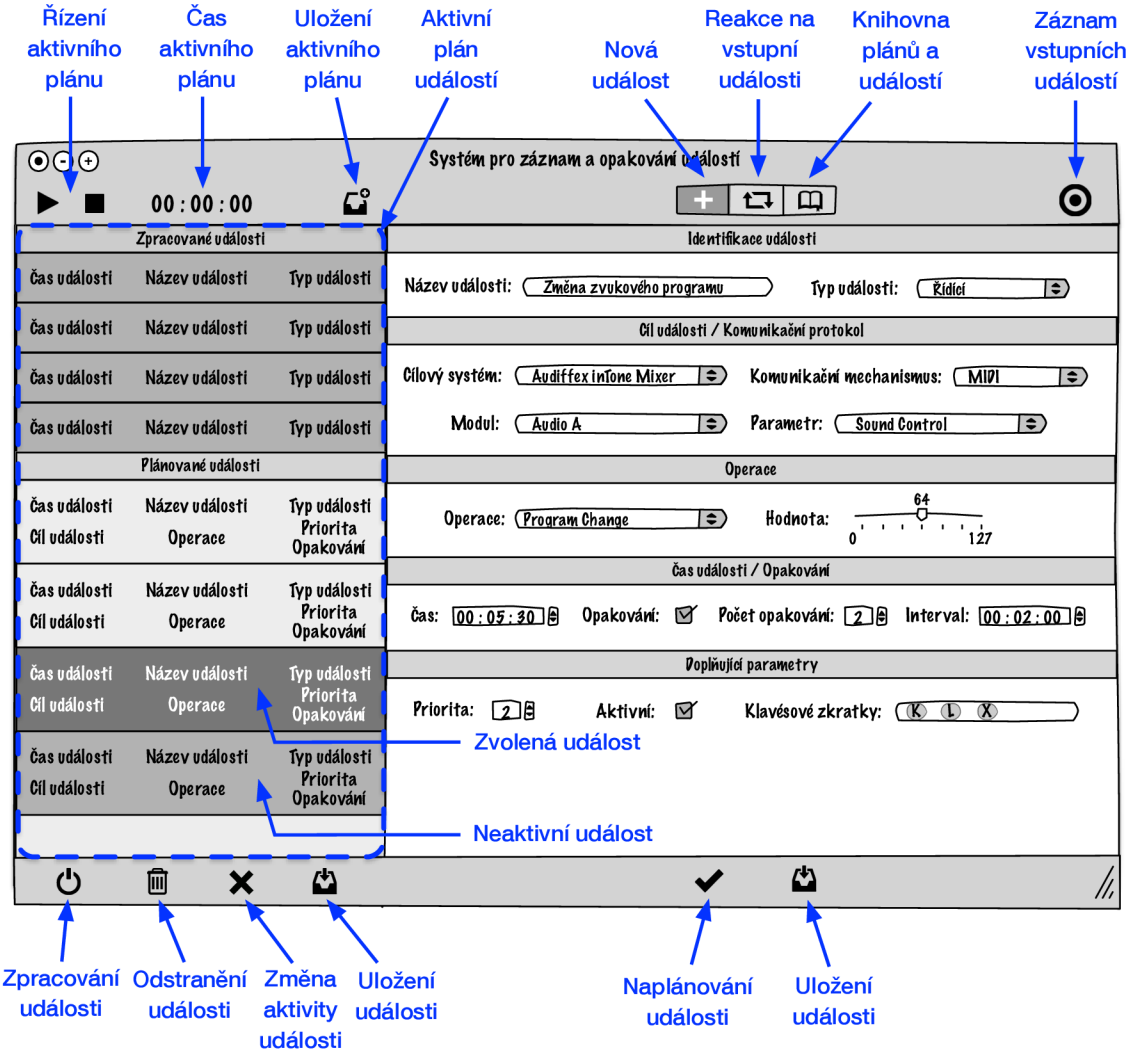
Obrázek 6.7: Vyhledání reakce na vstupní událost

6.3 Uživatelské rozhraní systému

Uživatelské rozhraní systému vychází z požadavků stanovených v kapitole 5.1. Návrh je orientován na snadnou manipulaci s událostmi a také na rychlý náhled na stav aktivního plánu událostí, který reprezentuje proces řízení zvukových systémů. Jeho součástí není specifikace implementačních detailů, které mohou být charakteristické pro cílovou platformu. Důraz je kladen především na rozmístění ovládacích prvků a interakci systému s uživatelem.

Hlavní obrazovka je rozdělena na 2 základní části. Levá část prezentuje aktivní plán událostí a jeho stav. V pravé části obrazovky jsou umístěny funkce pro záznam událostí, definování nové události, správu reakcí systému na vstupní události a také knihovny plánů a událostí. Levá část je v rámci celého systému nepřetržitě viditelná a poskytuje tak rychlý náhled na aktuální stav plánu událostí, prává část zobrazuje svůj obsah na základě kontextu, který uživatel stanoví.

Zobrazení událostí v systému je z velké části realizováno ve formě seznamů. Toto zobrazení je zahrnuto v návrhu především pro možnost reprezentovat potencionálně nekonečnou množinu událostí a také snadnou manipulaci s událostmi. Ovládací prvky, které se vztahují k celém segmentu hlavní obrazovky, jsou umístěny v horní části na liště ovládacích prvků. Lišta v dolní části obrazovky je pak vždy úzce spojena se zvoleným výběrem událostí nebo plánů v rámci seznamu nebo s aktuálním kontextem pravé části obrazovky.



Obrázek 6.8: Návrh uživatelského rozhraní - tvorba nové události

Levá část obrazovky reprezentující aktivní plán událostí využívá již zmíněné zobrazení ve formě seznamu. Plán je rozdělen do 2 sekcí, jejichž každá položka odpovídá jedné události. První sekce obsahuje události, které již v rámci daného plánu byly zpracovány. Druhá sekce zahrnuje plánované události, které ještě nebyly zpracovány. Události jsou reprezentovány všemi parametry definujícími událost, tedy včetně jejího cíle, operace, priority a specifikace opakování.

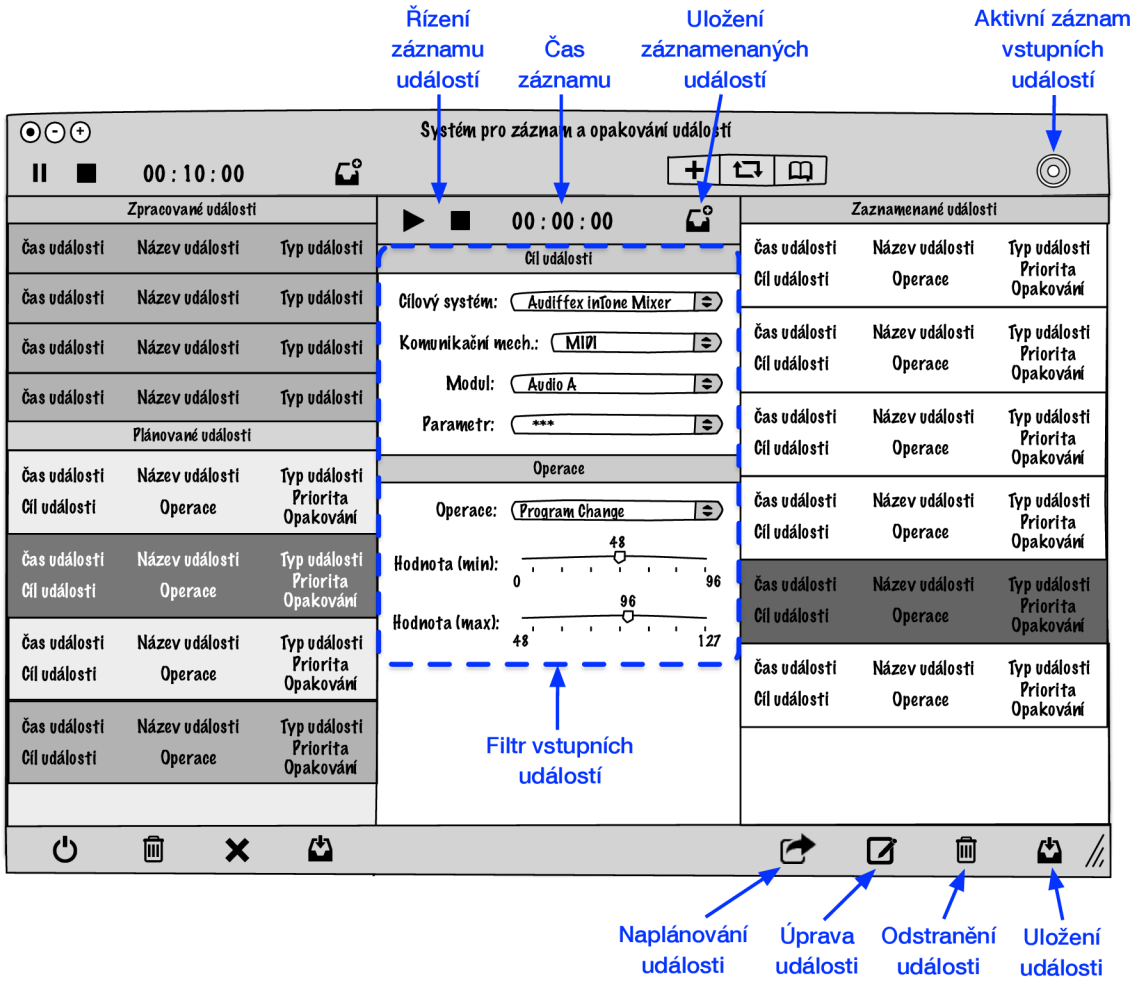
Řízení činnosti plánu událostí je prováděno ovládacími prvky situovanými na liště nad plánem událostí. Tyto ovládací prvky slouží ke spuštění plánu, jeho pozastavení nebo úplnému zastavení, které uvede plán do výchozího stavu. Součástí lišty je také informace o době, po kterou byl plán vykonáván. Ve výchozím stavu je tento čas vynulován. Pro uložení celého aktivního plánu do knihovny plánů a událostí je zde volba v pravé části nad plánem událostí.

Ovládací prvky specifické pro každou událost lze nalézt na liště pod plánem událostí. Jejich viditelnost je dána zvolením události v seznamu. Jejich funkce představuje možnost provést zpracování události okamžitě, bez ohledu na čas události, odstranit událost z aktivního plánu, nastavit událost jako neaktivní, což při jejím zpracování způsobí, že nebude

provedena žádná akce, a uložit událost do knihovny plánů a událostí, pokud se v ní již nenachází. Neaktivní události jsou v plánu odlišeny barvou jejich pozadí.

Pro rozlišení kontextu v pravé části obrazovky je na horní liště umístěn přepínač. Přepínač obsahuje 3 stavy měnící zobrazení na formulář pro tvorbu nové události, definování reakcí systému na vstupní události, správu knihovny plánů a událostí. Součástí horní lišty je také tlačítko pro změnu kontextu na záznam vstupních událostí. Barvou pozadí tohoto tlačítka je signalizována aktivita záznamu událostí, aby měl uživatel systému tuto informaci k dispozici nezávisle na volbě zobrazení v pravé části obrazovky. Na liště pod touto zobrazovací oblastí jsou rovněž ovládací prvky vztahující se k aktuálnímu obsahu.

Tvorba nové události je rozdělena do sekcí na základě parametrů události. Ovládací prvky pro specifikaci identifikace, cíle, operace a času jsou z velké části tvořeny roletkami, které nabízí možnosti daných parametrů. Specifickým ovládacím prvkem je zde nastavení hodnoty v rámci operace. Tato hodnota může nabývat různých rozsahů a může se jednat i o speciální ovládací prvek charakteristický pro zvukový systém. Součástí definice události jsou také klávesové zkratky, které mohou provést naplánování události. Po vytvoření události je možné provést její vložení do aktivního plánu událostí nebo ji uložit do knihovny plánů a událostí.



Obrázek 6.9: Návrh uživatelského rozhraní - záznam vstupních událostí

Zobrazení knihovny plánů a událostí je rozděleno na 2 segmenty. Levý segment obsahuje uložené plány událostí, v pravém segmentu jsou umístěny události knihovny. Zobrazení plánů a událostí v knihovně je rovněž realizováno seznamy. Po výběru plánu v levém segmentu knihovny je nahrazen pravý segment obsahem daného plánu ve formě seznamu událostí. Ovládací prvky na dolní liště zde slouží k úpravě událostí v knihovně nebo v rámci jednotlivých plánů. Z knihovny je také možné vybrat jeden z plánů a nastavit jej jako aktivní.

Reakce na vstupní události respektují také zvolené seznamové zobrazení. Seznam je zde pomyslně rozdělen na 2 části, kde levá část tvoří šablonu specifikující vstupní událost a v pravé části je odpovídající reakce ve formě události. Šablona pro vstupní událost je definována pomocí formuláře, který je svou podobou blízký tvorbě nové události. Událost reakce je možné měnit pomocí roletky, která nabízí uživateli události uložené v systému. Návrh obrazovky reakcí na vstupní události a knihovny plánů a událostí lze nalézt v příloze A.

Záznam vstupních událostí je poslední volbou zobrazení v pravé části obrazovky. Záznam je řízen podobně jako vykonávání aktivního plánu událostí. Uživatel může řídit jeho spuštění, pozastavení nebo úplné zastavení. Mimo ovládací prvky pro kontrolu záznamu je v levé části umístěn filtr vstupních událostí. Specifikace filtru je realizována stejným způsobem jako u definice šablony vstupní události v rámci reakcí systému. Vstupní události jsou řazeny v pravé části na základě času jejich záznamu. Zaznamenané události je možné upravit, naplánovat ke zpracování v aktivním plánu událostí, uložit nebo odstranit jejich záznam. V rámci nastavení vstupního filtru událostí je také uživateli umožněno uložit všechny události odpovídající danému filtru jako plán událostí.

Vzhledem ke zvolenému zobrazení pomocí seznamů se pro manipulaci s událostmi nabízí využít funkce *drag&drop* neboli potáhnout a upustit. S událostmi, které jsou reprezentovány řádky seznamů, lze tak v rámci systému zacházet interaktivně jejich potažením a umístěním do patřičného seznamu. Tímto způsobem lze například měnit pořadí vykonávání událostí v plánu nebo naplánovat událost jejím přesunutím z knihovny do plánu událostí.

Kapitola 7

Implementace systému

V této kapitole je představena implementace systému pro záznam a opakování událostí pro operační systém Mac OS X. V následujícím textu je nejdříve stručně charakterizována objektově orientovaná koncepce systému a její návaznost na návrh systému v kapitole 6.1. Dále je popsáno implementované uživatelské rozhraní systému a možnosti rozšíření systému. Závěr této kapitoly shrnuje vlastnosti implementovaného systému.

7.1 Objektově orientovaná koncepce systému

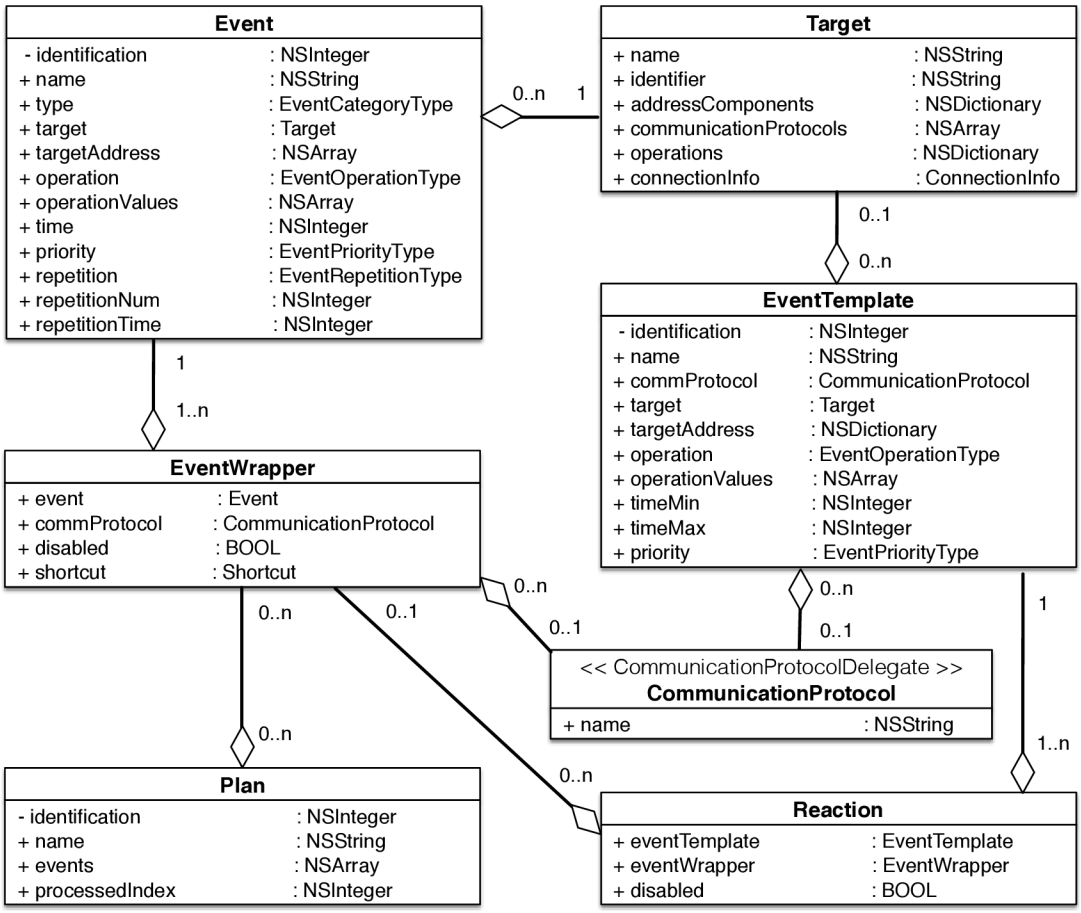
Pro implementaci systému bylo využito jazyka Objective-C. Tento jazyk lze svými vlastnostmi označit za vysokoúrovňový objektově orientovaný jazyk, který využívá koncepce *MVC* [1]. Díky této koncepci je v rámci implementace rozdělen datový model systému, řídící logika a uživatelské rozhraní do téměř nezávislých komponent.

Řídící logika systému je rozdělena do 4 základní tříd, které odpovídají hlavním funkcím systému. První z nich, třída *PlannerManager*, je zaměřena na správu aktivního plánu událostí. Tato třída řídí vykonávání plánu událostí, změny v plánovacím procesu a předává události na výstup ze systému. Druhá třída *RecordManager* provádí záznam příchozích událostí a jejich filtraci. Třetí třídou *ReactionManager* jsou kontrolovány definované reakce systému a je zde také implementováno vyhledání reakce na základě vstupní události, tak jak bylo popsáno v kapitole 6.2.2. Tyto třídy představují řízení *logické jednotky* systému vzhledem k dekompozici systému na funkční bloky z kapitoly 6.1.

Poslední kontrolním prvkem je třída *LibraryManager*. *LibraryManager* spravuje knihovnu plánů a událostí a umožňuje ostatním modulům přístup k uloženým datům prostřednictvím datových rozhraní. Tato třída také úzce spolupracuje s třídou *Configuration*, která má za úkol udržovat informace o řízených zvukových systémech a komunikačních protokolech. *LibraryManager* společně s třídou *Configuration* realizují *konfigurační modul* systému.

Zpracování vstupních komunikačních zpráv a jejich převod do interní reprezentace ve formě událostí implementuje třída *IOProcessor*. Tato třída pracuje s implementovanými komunikačními protokoly, pomocí kterých analyzuje vstupní zprávy a také provádí transformaci událostí na komunikační zprávy. *IOProcessor* tak implementuje vstupně-výstupní operace systému realizující navrhovaný *vstupně-výstupní modul* systému.

Datový model systému je charakterizován třídami *Event*, *EventWrapper*, *Plan*, *Target*, *Reaction*, *EventTemplate* a *CommunicationProtocol*. Jejich vzájemné propojení je zobrazeno na obrázku 7.1.



Obrázek 7.1: Závislosti tříd datového modelu systému

Třída *Event* tvoří základní reprezentaci události v systému. Její obsah je tvořen datovými atributy, které specifikující událost, tak jak byla definována v kapitole 6.2. Rozšířením této třídy je třída *EventWrapper*. Tato třída doplňuje k definici události atributy charakteristické pro zpracování události v systému jako je příznak aktivity události nebo klávesová zkratka pro její zpracování. Součástí třídy *EventWrapper* je také reference na komunikační protokol události definovaný abstraktní třídou *CommunicationProtocol*.

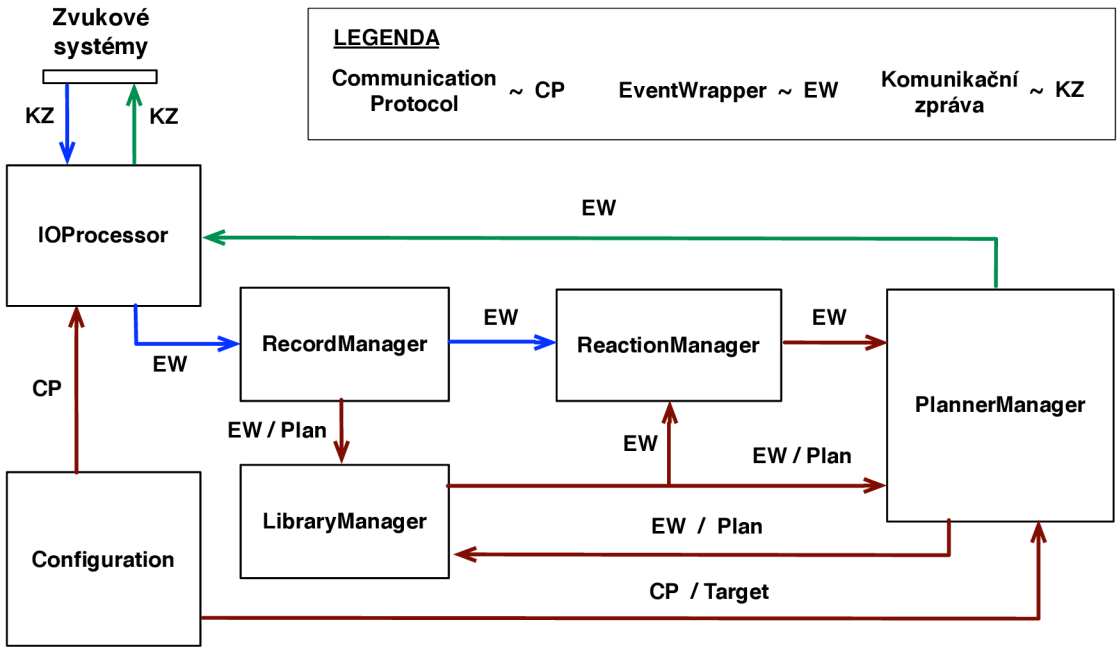
Implementace všech komunikačních protokolů v systému vychází z této abstraktní třídy. Součástí její definice je rozhraní, které musí být implementováno každou podtřídou, a tedy komunikačním protokolem. Toto rozhraní definuje základní požadavky pro integraci komunikačního protokolu v systému a bude podrobněji popsáno v kapitole 7.3, která je orientována na možnosti rozšíření systému.

Plán událostí jako sekvenci událostí v pořadí jejich plánovaného zpracování a časový údaj reprezentující stav vykonávání plánu obsahuje třída *Plan*. Sekvence událostí je uvnitř plánu reprezentována objekty *EventWrapper*. Řízený zvukový systém je zastoupen třídou *Target*. *Target* popisuje strukturu zvukového systému, a tedy i množinu jeho ovladatelných parametrů, komunikační protokoly a jejich operace, které jsou v systému podporovány. Podrobnější specifikace třídy *Target* je součástí kapitoly 7.3.

Zbývající třídy datového modelu systému jsou věnovány reprezentaci reakcí na vstupní události. Šablona definující parametry vstupní události je implementována třídou *Event-*

Template. Tato šablona společně s událostí reakce definovanou třídou *EventWrapper* tvoří třídu *Reaction*. Součástí reakce jsou také doplňující atributy charakteristické pro zpracování reakce v systému.

Schéma systému, které zachycuje obrázek 7.2, je doplněno o třídy implementující vlastnosti funkčních bloků. Datové závislosti mezi těmito třídami jsou znázorněny pomocí implementovaného datového modelu systému.



Obrázek 7.2: Implementované funkční bloky systému

7.2 Uživatelské rozhraní systému

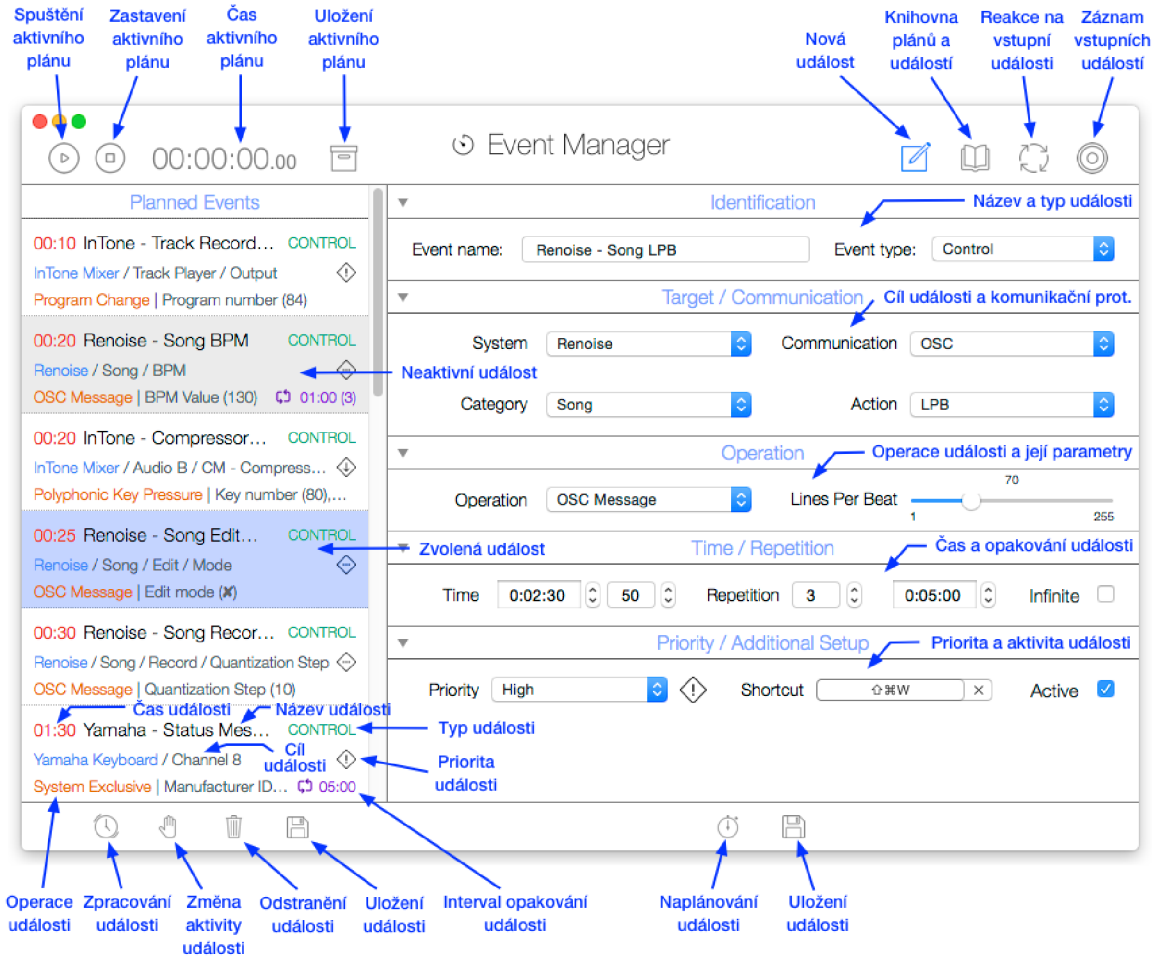
Uživatelské rozhraní implementovaného systému realizuje návrh popsaný v kapitole 6.3. Ovládací prvky jsou implemetovány pomocí vrstvy *Cocoa* na platformě Mac OS X.

Základní členění hlavní obrazovky respektuje rozdělení na plán aktivních událostí a zbývající funkce systému, které jsou zobrazovány v závislosti na zvoleném kontextu. Velikost těchto oblastí je možné interaktivně přizpůsobit pomocí dělicí čáry, kterou je možné pohybovat v horizontálním směru.

Zobrazení událostí a také plánů v systému je implementováno seznamem, kde každý řádek odpovídá jedné události. Cílem tohoto zobrazení je poskytnout uživateli co nejpřesnější charakteristiku dané události bez nutnosti zobrazení všech jejich vlastností. Nejdůležitější informace jako je čas události, cílový zvukový systém, operace události a její opakování jsou proto barevně odlišeny. Pro určení priority události byly zvoleny 3 úrovně. Nastavená úroveň je signalizována ikonou u pravého okraje zobrazené události.

Mimo informace charakterizující událost je také uživatel informován o stavu události. Stav události je reprezentován barvou pozadí daného řádku seznamu. Při aktivním stavu události je pozadí bílé, v neaktivním stavu světlě šedé, při chybě zpracování události červené a při aktivním výběru události modré.

Manipulace s událostmi je implementována dvěma způsoby. První způsob je orientován na volby reprezentované tlačítky ve formě ikon. Tyto volby se vztahují ke zvoleným událostem v seznamu a provádí tak požadované akce s aktuálním výběrem. Druhý typ je implementován přímou manipulací s událostmi pomocí jejich uchopení a upuštění na požadovaném místě. Takto lze měnit pořadí naplánovaných událostí, naplánovat událost na stanovený čas nebo přesunout události mezi plány. Zmíněné možnosti manipulace se vztahují obecně k množině událostí. Lze tedy takto pracovat současně s několika událostmi, pokud to akce, která má být provedena, umožňuje.

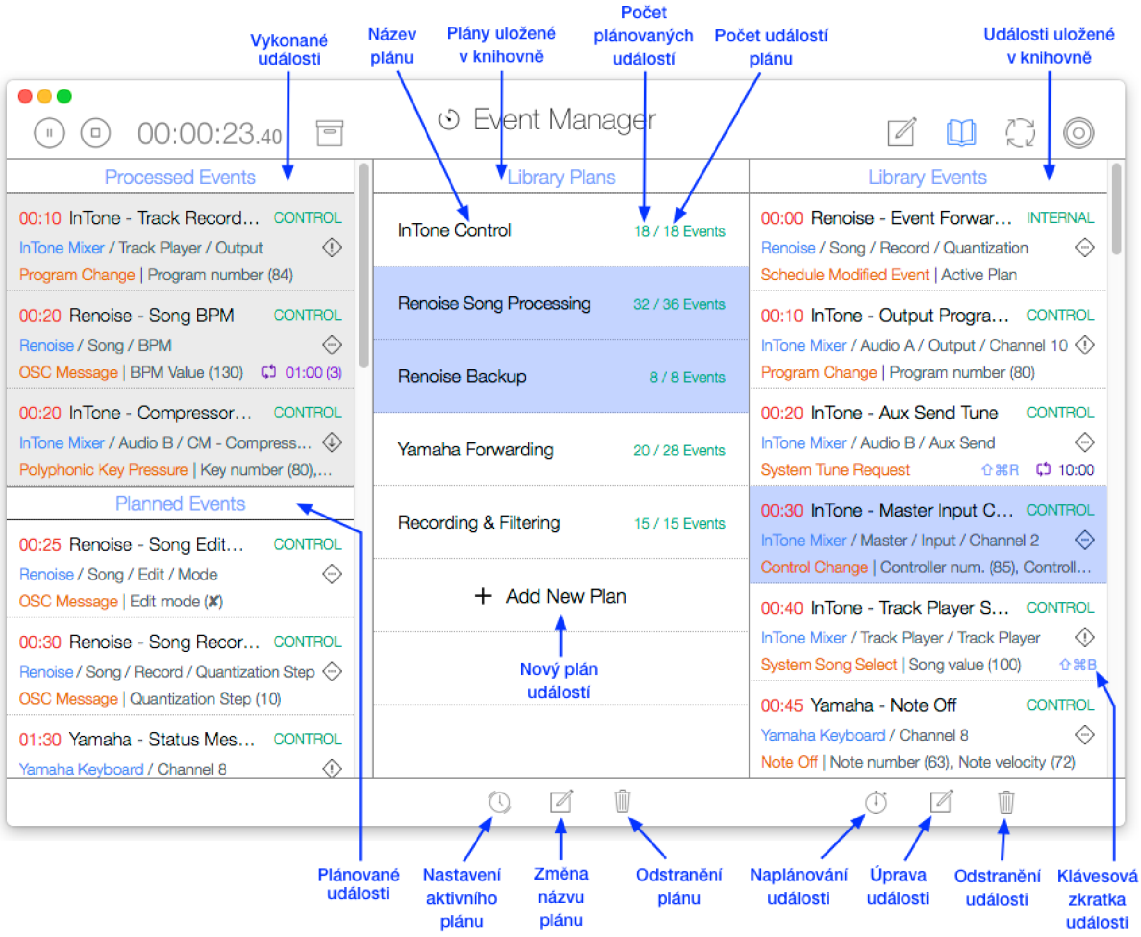


Obrázek 7.3: Uživatelské rozhraní systému - tvorba nové události

Při tvorbě nové události je uživatelské rozhraní rozděleno do sekcí na základě parametrů události. Zobrazení každé sekce je možné volitelně minimalizovat vzhledem k tomu, že některé sekce, jako například operace události, mohou zabírat mnoho prostoru. Velikost každé sekce se dynamicky mění na základě jejího obsahu.

Většina ovládacích prvků této oblasti je tvořena rozbalovacími nabídkami pro výběr z možností a nebo klasickými textovými poli a tlačítky. Za pozornost zde stojí ovládací prvky, které slouží pro nastavení parametrů operace. V závislosti na typu parametru je zvolen odpovídající ovládací prvek. Pro logickou hodnotu je tomu přepínač, pro hodnotu v určitém rozsahu posuvník s volitelně definovaným krokem, pro číselnou hodnotu pole

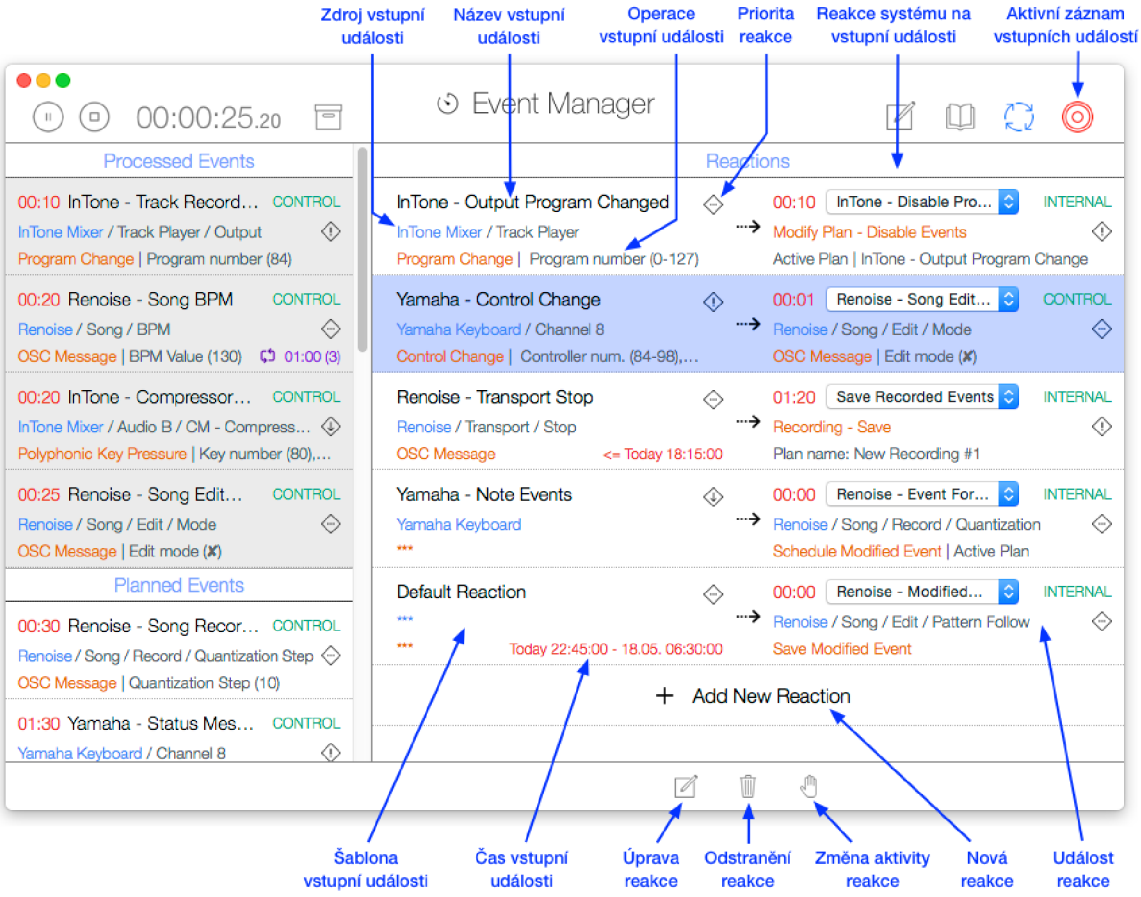
akceptující číselné hodnoty a pro řetězec textové pole proměnlivé délky. Vzhledem k velkým rozsahům číselných hodnot je posuvník obvykle kombinován s možností přesně zadat číselnou hodnotu ve vstupním poli. Speciální ovládací prvek je zde také implementován pro nastavení klávesové zkratky události. Jedná se o vstupní pole, které po stisknutí přejde do režimu, kdy očekává kombinaci kláves, které zaznamená jako klávesovou zkratku pro naplánování události.



Obrázek 7.4: Uživatelské rozhraní systému - knihovna plánů a událostí

Knihovna plánů a událostí je rozdělena na 2 segmenty. Levý segment je vždy tvořen uloženými plány událostí. Každý plán je reprezentován řádkem seznamu, stejně jako tomu je u události. U plánu je zobrazen jeho název a informace o počtu jeho událostí. S plány událostí lze také přímo manipulovat a nastavit tak například plán jako aktivní.

Pravý segment mění své zobrazení v závislosti na výběru plánu v levém segmentu. Ve výchozím stavu, kdy není žádný plán zvolen, je zobrazen seznam uložených událostí v knihovně. Při výběru jediného z plánů v levém segmentu jsou události knihovny nahrazeny obsahem tohoto plánu. Jiný způsob zobrazení plánu událostí je umožněn dvojklikem na řádek reprezentující plán, který otevře nové okno s obsahem plánu.



Obrázek 7.5: Uživatelské rozhraní systému - reakce na vstupní události

Uživatelské rozhraní obrazovky pro správu reakcí systému na vstupní události je realizováno analogicky k předchozím obrazovkám. Zobrazení reakcí implementuje seznam, jehož prvky reprezentují definované reakce. Pozadí prvku seznamu signalizuje stav reakce stejně jako u událostí.

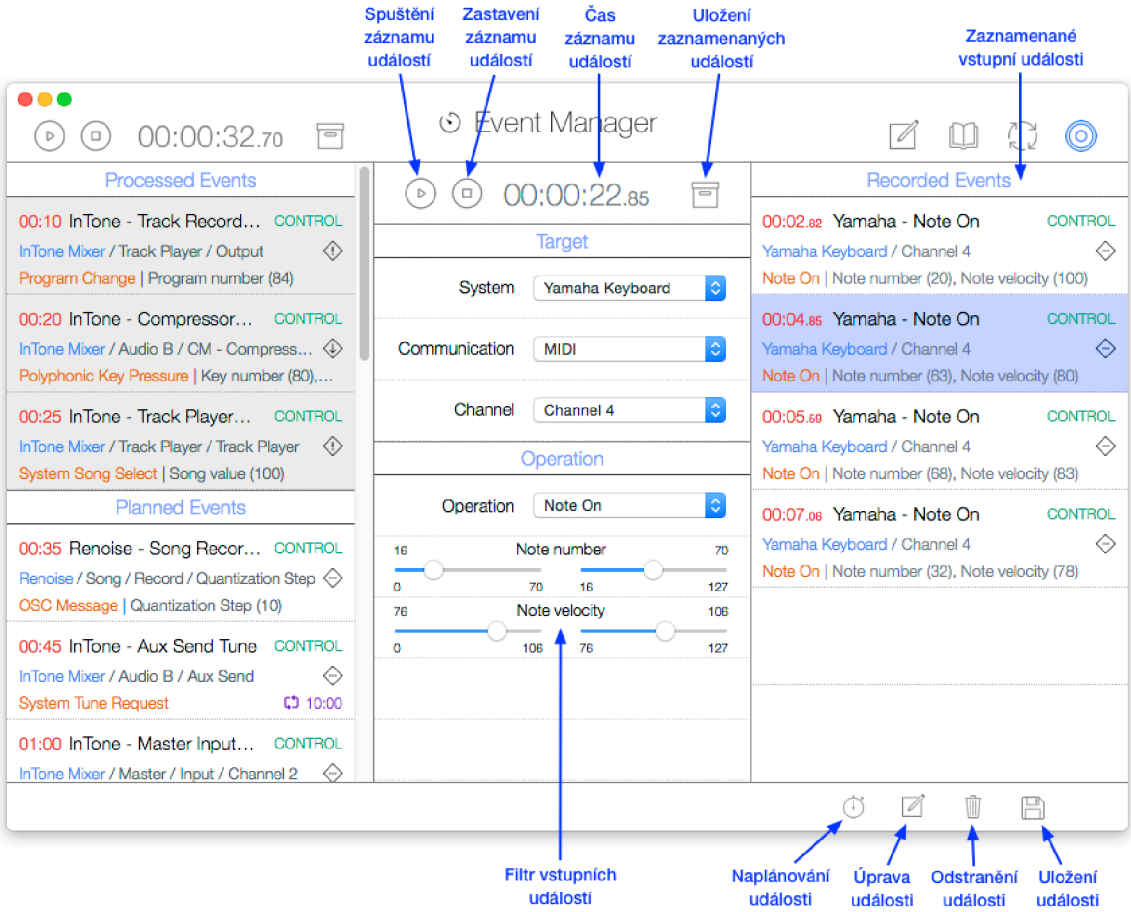
Vzhledem ke struktuře reakce je zde implementována jednotná konvence barevnosti označující elementy jak šablony definující vstupní událost, tak i události reakce. Událost reakce lze přímo měnit v seznamu, a to pomocí rozbalovací roletky, která nabízí události uložené v knihovně. Pro úpravu šablony vstupní události je implementována oddělená obrazovka, která je založena na uživatelském rozhraní využívaném pro úpravu nebo tvorbu událostí.

Poslední obrazovka sloužící pro řízení záznamu vstupních událostí je také rozdělena na 2 části. Levá část obsahuje ovládací prvky pro řízení záznamu a také filtr definující zobrazení v pravé části. Filtr představuje šablonu pro zobrazení zaznamenaných vstupních událostí, podobně jako tomu bylo u reakce systému. Ovládací prvky pro nastavení parametrů filtru jsou voleny na základě typu atributů. V případě číselného intervalu jsou to posuvníky definující jeho rozmezí, pro ostatní typy je přímo nastavena konkrétní hodnota. U řetězcových atributů lze ve specifikaci hodnoty rovněž využít regulárních výrazů.

V pravé části jsou řazeny vstupní události na základě času jejich záznamu. Zobrazení zaznamenaných událostí již respektuje definované parametry filtru. S událostmi lze manipulovat stejně jako v aktivním plánu událostí nebo v knihovně.

Aktivita záznamu vstupních událostí je také signalizována změnou barvy tlačítka, které

přepíná kontext na obrazovku záznamu. Tento způsob byl implementován vzhledem k tomu, že uživatel může pracovat v systému na jiných obrazovkách a ovládací prvky záznamu tak nemusí být viditelné. Aktivní záznam vstupních událostí je signalizován na obrázku 7.5.



Obrázek 7.6: Uživatelské rozhraní systému - záznam vstupních událostí

7.3 Rozšiřitelnost systému

Důležitou vlastností implementovaného systému je jeho rozšiřitelnost. Tato vlastnost může nabývat mnoha významů, pro účely implementovaného systému se jedná především o rozšíření množiny zvukových systémů, které je možné řídit a zaznamenávat jejich události. Druhým pohledem je rozšiřitelnost komunikačních protokolů, jejichž zprávy je systém schopen zpracovávat, a také s tím související množina událostí, které lze v systému vytvářet.

Za tímto účelem je v systému implementováno rozhraní, které umožňuje rozšířit systém o specifikaci zvukového systému. Integrace těchto informací probíhá pomocí konfiguračních souborů, které lze jak načíst, tak i generovat. Konfigurační soubor je vytvořen formou popisu struktury zvukového systému v jazyce *XML*, která byla představena v kapitole 3.3. Součástí tohoto souboru je nejen popis struktury systému, ale také další informace důležité pro integraci a řízení zvukového systému. Následující úsek konfiguračního souboru ukazuje základní bloky specifikující informace potřebné pro integraci zvukového systému.

```
<dict>
   <!-- Atributy identifikující systém -->
   <key>CommunicationProtocols</key>
   <array>
      <!-- Komunikační protokoly systému -->
   </array>
   <key>Operations</key>
   <dict>
      <!-- Omezení příkazů komunikačních protokolů -->
   </dict>
   <key>ConnectionInfo</key>
   <dict>
       <!-- Informace pro navázání spojení -->
   </dict>
   <key>AddressComponents</key>
   <dict>
      <key> <!-- Komunikační protokol --> </key>
      <dict>
         <!-- Adresový prostor systému -->
      </dict>
   </dict>
</dict>
```

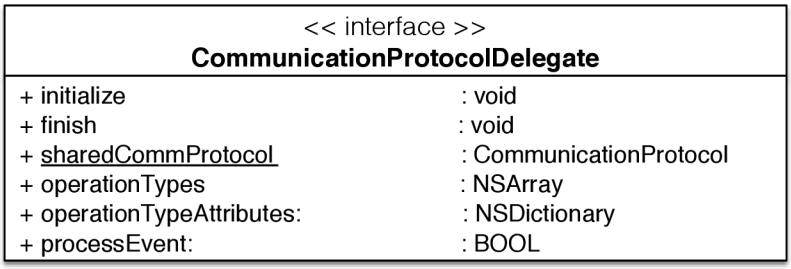
Ukázka popisu 7.7: Úsek konfiguračního souboru popisující zvukový systém

První blok souboru je tvořen atributy, které identifikují zvukový systém. Nachází se zde především jeho název, unikátní identifikátor a volitelně další atributy charakterizující konfigurační soubor. Druhý blokem jsou definovány komunikační protokoly, které systém podporuje. Na tento blok navazuje možnost specifikace omezení komunikačních protokolů na zúženou množinu jejich operací, které jsou zvukovým systémem rozeznávány. Poslední blok slouží pro popis adresového prostoru systému. Pro každý komunikační protokol je možné definovat alternativní hierarchii adresového prostoru a její položky.

Vzhledem k rozsahu konfiguračního souboru zde není uveden přesný formát zápisu jednotlivých elementů. Příklady konfiguračních souborů včetně jejich formální specifikace jazykem *XSD* jsou k dispozici v příloze B.

Takto vytvořený popis zvukového systému je v rámci objektově orientované koncepce systému reprezentován třídou *Target*. Řídící systém s tímto popisem pracuje především při úpravě zaznamenaných událostí a jejich reprodukci.

Rozšiřitelnost systému o komunikační protokoly je implementováno na základě objektově orientované koncepce systému. Základní reprezentaci komunikačního protokolu v systému představuje třída *CommunicationProtocol*. Tato třída definuje rozhraní, které musí být implementováno každým z komunikačních protokolů. Systém pak je schopen pouze na základě znalosti jména implementační třídy komunikačního protokolu provádět potřebné akce pro analýzu vstupních zpráv a tvorbu komunikačních zpráv.



Obrázek 7.8: Rozhraní definující implementaci komunikačního protokolu

Rozhraní definující vlastnosti implementovaného komunikačního protokolu se skládá z 6 metod. Metoda *initialize* je vykonána vždy po spuštění systému. Umožňuje nastavit výchozí hodnoty potřebných datových struktur a zahájit tak příjem komunikačních zpráv. Analogicky metoda *finish* je vyvolána před ukončením činnosti systému, čímž je umožněno komunikačnímu protokolu korektně ukončit datová spojení. Každá třída implementující komunikační protokol je instanciována pouze jednou, aby nedocházelo k duplikaci sdílených datových struktur. Jedná se tedy o koncept tzv. *jedináčka*. Pro přístup ke sdílené instanci této třídy pak slouží metoda *sharedCommProtocol*.

Operace komunikačního protokolu definují metody *operationTypes* a *operationTypeAttributes*. První ze zmíněných metod definuje identifikace operací, které jsou v systému zaregistrovány. Druhá z metod poskytuje podrobnější informace o dané operaci jako je počet parametrů, jejich datové typy, datové rozsahy, výchozí hodnoty a názvy zobrazené v systému. Poslední metodou *processEvent* je implementována transformace interní reprezentace události na zprávu daného komunikačního protokolu a případně její odeslání.

Rozšíření systému z hlediska událostí, které je možné zpracovávat, se v případě řídících událostí odvíjí od implementovaných komunikačních protokolů. Pro interní události byl implementován stejný mechanismus jako v případě komunikačních protokolů.

7.4 Vlastnosti implementovaného systému

Vlastnosti implementovaného systému lze souhrně označit jako jeho 3 hlavní funkce, kterými jsou řízení, schopnost reakce a záznam událostí. Tyto funkce se nejvíce podílí na koncepci systému a celém návrhu, ze kterého vychází i implementace systému a jeho uživatelského rozhraní.

Řízením je možné v systému charakterizovat tvorbu událostí, jejich zpracování a zaslání zvukovým systémům ve formě řídících příkazů. Za tímto účelem jsou v systému implementovány 2 komunikační protokoly - *MIDI* a *OSC*. Implementovaný systém umožnuje vytvořit libovolnou komunikační zprávu jednoho z těchto protokolů a zaslat ji řízenému systému. Součástí řízení je také schopnost plánování, opakování a vykonávání definované sekvence událostí. Pro zajištění těchto funkcí je v systému implementován koncept událostí a jejich plánů. V systému lze specifikovat čas události a její opakování. Definovanou událost je možné zařadit do plánu, který systém umí zpracovat a vykonávat na základě vlastností jeho událostí.

Druhým pohledem na řízení, které úzce souvisí se schopností reakce na vstupní události, je ovládání implementovaného systému bez zásahu uživatele. Za tímto účelem jsou v systému implementovány interní události. Vykonání těchto událostí způsobí interní změnu chování

systému. Tyto interní události lze rozdělit na základě oblasti systému, kterou ovlivňují.

První skupina interních událostí se vztahuje k aktivnímu plánu událostí. Pomocí těchto událostí je možné spustit nebo zastavit vykonávání plánu, uložit plán do knihovny nebo nastavit nový plán jako aktivní. Druhá skupina událostí slouží k úpravě plánů událostí. Tyto události modifikují buď aktivní plán, nebo některý z plánů, který je uložen v knihovně. Lze tak změnit stav událostí plánu na aktivní nebo neaktivní, odstranit události z plánu nebo naopak přidat do plánu nové události. Třetí množina je specifická svým charakterem a návaznosti na reakce systému na vstupní události. Tyto události jsou implementovány tak, aby mohly být efektivně využity při definování reakce systému. Patří sem událost, která provede úpravu vstupní události tak, že např. nahradí její cíl a zařadí ji do některého z plánů nebo ji uloží do knihovny. Další událostí této kategorie lze také okamžitě zpracovat libovolnou sekvenci událostí.

Předposlední skupina umožňuje pracovat s reakcemi systému. Provedením těchto událostí tak lze změnit stav reakce na aktivní nebo neaktivní a reakci odstranit. Poslední kategorie je určena pro řízení záznamu vstupních událostí. Záznam lze spustit, zastavit, nastavit filtr definující šablonu pro vstupní události a uložit události odpovídající zvolenému filtru.

Schopnost systému reagovat na vstupní události je další vlastností implementovaného systému. Systém umožňuje definovat reakce na vstupní události pomocí šablony, která specifikuje vlastnosti vstupní události. Pro tyto události je pak vyhledána reakce, kterou může být buď jedna z výše zmíněných interních událostí, nebo libovolná jiná událost. Systém tak může reagovat na vstupní podněty změnou své konfigurace nebo aktivního procesu řízení.

Třetí z hlavních funkcí systému je záznam vstupních událostí. Na základě implementovaných komunikačních protokolů *MIDI* a *OSC* je systém schopen zaznamenat libovolnou událost popsanou pomocí těchto protokolů. Zaznamenané události lze v systému filtrovat, upravit, uložit nebo případně přeposlat jinému cíli.

Implementovaný systém je vzhledem ke své objektově orientované koncepci snadno rozšiřitelný. Lze tak implementovat nový komunikační protokol pro záznam událostí nebo tvorbu nových událostí, definovat nové interní události nebo rozšířit množinu zvukových systémů, se kterými je systém schopen pracovat, bez nutnosti zásahu do struktury systému.

Pokud se podíváme na systém z hlediska jeho použití v praktických situacích, pak budou jeho vlastnosti vycházet z funkcí zmíněných výše. Vzhledem ke schopnosti řízení více zvukových systémů může systém fungovat jako centrální řídící jednotka, která má na starosti skupinu určitých zařízení. Od těchto zařízení přijímá a vyhodnocuje informace o jejich stavu a upravuje tak svůj plán řízení. Systém také může automaticky reagovat na situace při poruše zařízení, kdy je nutné zajistit dostupnost jiného zařízení.

Jinou funkci může implementovaný systém plnit např. při potřebě efektivně zpracovávat zvukové signály. Systém může přijímat veškerou komunikaci od zdrojového systému a provádět tak filtraci informací nutných pro další zpracování. Tyto filtrované informace může přeposílat systému, který provede jejich zpracování, a zbývající informace zasílat cílovému systému. Současně se zmíněnými možnostmi využití může systém také zaznamenávat vstupní události a periodicky je ukládat pro pozdější analýzu nebo jiné zpracování.

Kapitola 8

Závěr

Úkolem této práce bylo navrhnout a implementovat systém pro záznam a opakování událostí pro zvukové systémy. Tento úkol byl splněn a navržený systém byl implementován na platformě Mac OS X.

Pro návrh a implementaci systému bylo nutné se seznámit se zvukovými systémy. Nejdříve byla provedena analýza jejich uživatelského rozhraní pro získání představy o jejich vlastnostech. Dále byl nastudován možný popis stavu a struktury těchto systémů a komunikační protokoly, které jsou těmito systémy využívány. Vzhledem ke zvolené implementační platformě Mac OS X byly nastudovány základní vlastnosti tohoto systému, možnosti vývoje aplikací a charakteristické rysy jejich uživatelského rozhraní.

Před stanovením požadavků na funkčnost systému byly analyzovány získané poznatky a diskutovány možnosti jeho multiplatformní realizace. Požadavky na výsledný systém byly stanoveny ve spolupráci se společností DISK Multimedia, s.r.o., která poskytovala odborné konzultace při řešení této práce. Mezi tyto požadavky patří především záznam vstupních událostí, jejich plánované vykonávání a opakování, úprava jejich vlastností a tvorba nových událostí. Součástí stanovených vlastností je také schopnost systému na vstupní události reagovat a možnost rozšířit systém tak, aby pracoval s libovolným zvukovým systémem.

Na základě definovaných požadavků byl představen návrh systému a jeho rozdělení do funkčních bloků. Tento návrh pracuje s konceptem událostí, který byl rovněž specifikován při návrhu. Koncept událostí zahrnuje specifikaci parametrů událostí a návrh jejich zpracování. Navržený systém byl implementován na platformě Mac OS X pomocí jazyka Objective C a pro implementaci uživatelského rozhraní systému bylo využito vrstvy *Cocoa*. V systému je implementována podpora komunikačních protokolů *MIDI* a *OSC*. Systém umožnuje provést záznam a úpravu vlastností libovolné události popsané pomocí těchto protokolů. Mimo záznam vstupních událostí je v systému implementováno plánování událostí a možnost reakce na vstupní podněty od zvukových systémů. Požadované vlastnosti implementovaného systému byly ověřeny ve spolupráci se společností DISK Multimedia, s.r.o.

Jako plánované pokračování této práce lze implementačně rozšířit navržený koncept událostí. Úroveň abstrakce tohoto konceptu poskytuje dostatek prostoru pro definici nových interních událostí systému nebo jiných typů událostí. Systém je také možné rozšířit o komunikační protokoly pro potřeby specializovaných zvukových systémů, jako je např. protokol *DMP*.

Jinou alternativou pokračování této práce je rozšířit systém z hlediska uživatelského rozhraní. Do systému by bylo vhodné implementovat specializované ovládací prvky pro lepší vizualizaci parametrů událostí a také integrovat informace o významu hodnot parametrů událostí, které poskytnou uživateli lepší přehled při jejich úpravě.

Literatura

[1] Concepts in Objective-C Programming [online]. Cupertino, USA: Apple Inc., 2012 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/library/mac/documentation/General/Conceptual/CocoaEncyclopedia/CocoaEncyclopedia.pdf.

[2] Mac Technology Overview [online]. Cupertino, USA: Apple Inc., 2014 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/OSX_Technology_Overview/OSX_Technology_Overview.pdf.

[3] OS X Human Interface Guidelines [online]. Cupertino, USA: Apple Inc., 2014 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf.

[4] Programming with Objective-C [online]. Cupertino, USA: Apple Inc., 2014 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf.

[5] Apple - Logic Pro X - In Depth [online]. Cupertino, USA: Apple Inc., 2015 [cit. 2015-05-01], Dostupné z: https://www.apple.com/logic-pro/in-depth/.

[6] The Swift Programming Language [online]. Cupertino, USA: Apple Inc., 2015 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/library/mac/documentation/Swift/Conceptual/Swift_Programming_Language/.

[7] Xcode - Features - Apple Developer [online]. Cupertino, USA: Apple Inc., 2015 [cit. 2015-05-01], Dostupné z: https://developer.apple.com/xcode/features/.

[8] Reason Overview – Reason – Propellerhead [online]. Propellerhead Software AB, 2014 [cit. 2015-05-01], Dostupné z: https://www.propellerheads.se/reason.

[9] VST Plug-Ins Software Developer Kit 3.0.1. Steinberg Media Technologies GmbH., 2008.

[10] Qt Documentation [online]. The Qt Company, 2014 [cit. 2015-05-01], Dostupné z: http://doc.qt.io/qt-5/reference-overview.html.

[11] Berryman, J.: Open Control Architecture (OCA) Technical Overview [online]. 2014 [cit. 2015-05-01], Dostupné z: http://ocaalliance.com/documents/AES%2044%20-%20OCA%20Technical%20Overview%20v02%20.pdf.

[12] Gao, S., Sperberg-McQueen, C. M., Thompson, H. S.: W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [online]. 2012 [cit. 2015-05-01], Dostupné z: http://www.w3.org/TR/xmlschema11-1/.

[13] Guérin, R.: *Velká kniha MIDI*. Brno, Česká Republika: Computer Press, 2004, ISBN 978-8072269852.

[14] Krause, A.: *Foundations of GTK+ Development*. Berkeley, USA: Apress, 2007, ISBN 978-1590597934.

[15] Krkavec, P. a kol.: Protokol DMP verze 2. Výzkumná zpráva projektu podporovaného MPO ČR - Prostorové akustické efekty pro systémy vícekanálového digitálního zpracování zvuku, 2007.

[16] Krkavec, P. a kol.: Uživatelské rozhraní aplikací pro řízení zpracování zvukových signálů v reálném čase. Výzkumná zpráva projektu podporovaného MPO ČR - Vícenásobně využitelný systém číslicového zpracování multimediálních signálů, 2010.

[17] Müller E., Tesic Z., Rogalinski P., Alnaes M., Finne S., Asnaghi L., Jälevik E., Foster K.: Renoise - What is Renoise? [online]. 2014 [cit. 2015-05-01], Dostupné z: http://renoise.com/products/renoise.

[18] Quin, L.: Extensible Markup Language (XML) [online]. 2003 [cit. 2015-05-01], Dostupné z: http://www.w3.org/XML/.

[19] Rathmann, U., Wilgen, J.: Qwt User's Guide: Qwt - Qt Widgets for Technical Applications [online]. 2014 [cit. 2015-05-01], Dostupné z: http://qwt.sourceforge.net/index.html.

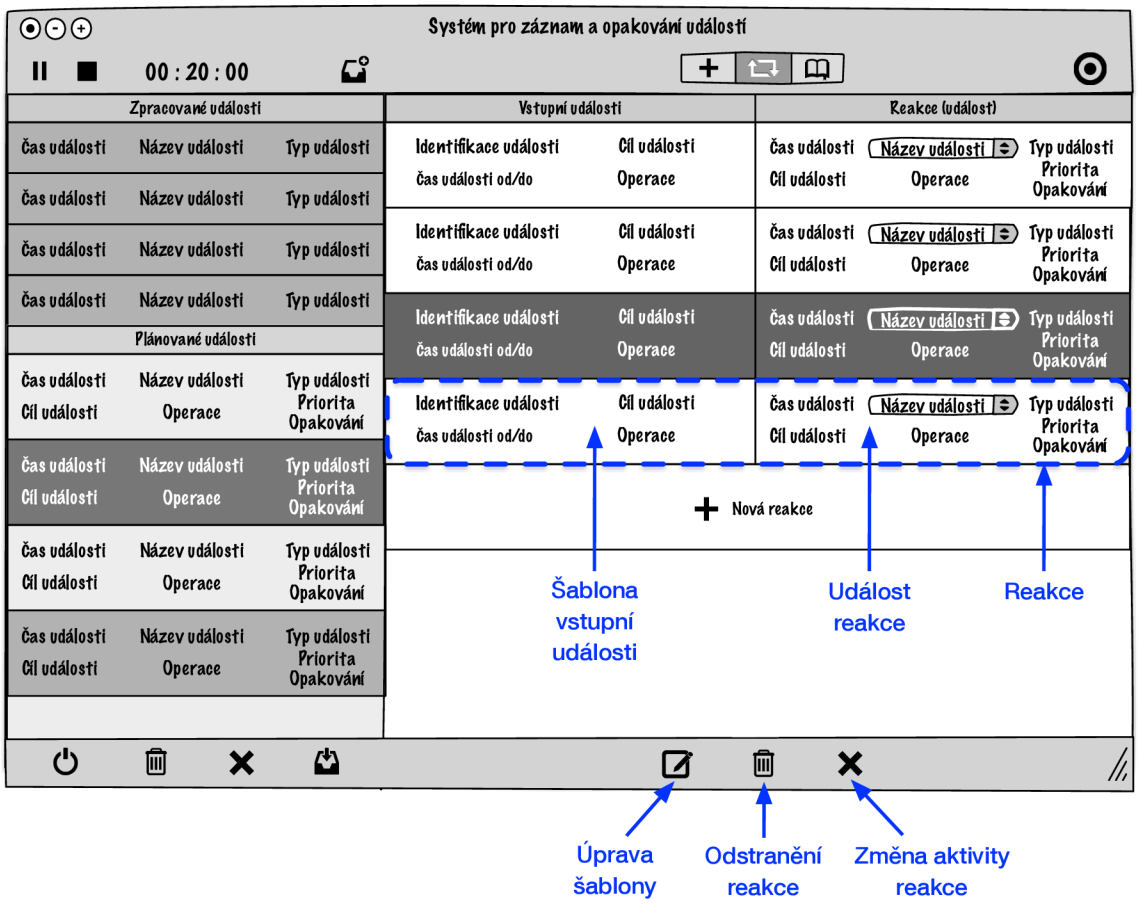
[20] Reichl, J., Všetička, M.: Encyklopedie fyziky [online]. 2006 [cit. 2015-05-01], Dostupné z: http://fyzika.jreichl.com.

[21] Smart, J., Zeitlin, V., Dunn, R., Csomor, S., Petty, B., Montorsi, F., Roebling, R.: wxWidgets: Documentation [online]. 2014 [cit. 2015-05-01], Dostupné z: http://docs.wxwidgets.org/stable/.

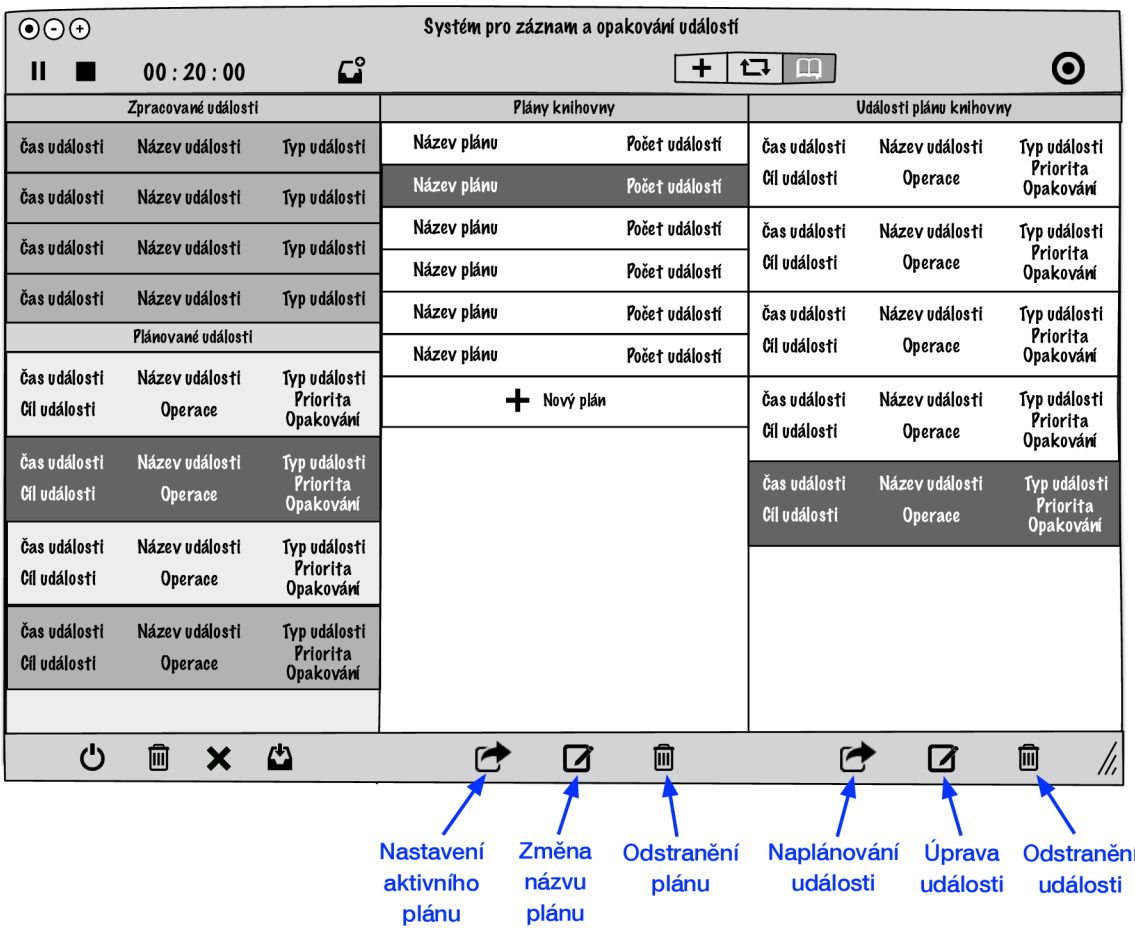
[22] Wright, M.: OpenSound Control Specification [online]. 2002 [cit. 2015-05-01], Dostupné z: http://archive.cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html.

Příloha A

Návrh uživatelského rozhraní systému



Obrázek A.1: Návrh uživatelského rozhraní - reakce na vstupní událostí



Obrázek A.2: Návrh uživatelského rozhraní - knihovna plánů a událostí

Příloha B

Obsah CD

- Zdrojový kód implementovaného systému ve formě zdrojových souborů a projektu v prostředí *Xcode*. Kromě zdrojových dat je také přiložen spustitelný balík aplikace pro Mac OS X ve formátu *app*.
- Zdrojové soubory této technické zprávy pro LaTeX včetně souboru makefile pro jejich přeložení. Součástí je také elektronická verze této práce ve formátu pdf.
- Konfigurační soubory popisující zvukové systémy pro jejich integraci do systému.
- Formální popis konfiguračního souboru zvukového systému v jazyce *XSD*.