1 Informace přes telefon

V dnešní době více než v minulosti hrají informace důležitou roli, zvláště pak ty informace získané v pravou chvíli. Aby se člověk mohl správně rozhodnout potřebuje mít přístup k informacím tehdy, kdy to nejvíce potřebuje, tehdy má-li se rozhodovat. Toto si stále více lidí uvědomuje a proto lze v dnešní době přistupovat k informacím (Internet, intranet, CRM systémy, systém pro správu objednávek atd.) nejen přes monitor počítače, ale i přes různé přenosné notebooky, PDA zařízení, *mobilní telefony* a jiné *telefonní přístroje*.

Výhoda prezentace dat přes počítač, resp. monitor je nesporná, ale pokud uživatel potřebuje být mobilní, ztrácí tato výhoda na významu. Potom je nutné zvolit jiný způsob komunikace se zdrojem informací (např. s vnitropodnikovým systémem) – notebooky mají určitě velký význam, zejména když je třeba pracovat s větším objemem dat, ale jistou nevýhodou je nutná vazba na zařízení (internetová přípojka, mobil s GPRS[1]), které uživatele připojí ke zdroji dat a umožní mu s ním komunikovat. PDA zařízení jsou hitem poslední doby a hlavně kvůli rozměrům a bohaté funkčnosti jsou velice oblíbené. Stejně jako u notebooků jsou starší PDA zařízení většinou závislá na mobilním telefonním přístroji, který zajišťuje připojení k Internetu. V dnešní době se již řada těchto zařízení dodává s vestavěným GSM modulem[2], který dovoluje automatické připojení k Internetu (samozřejmostí je podpora GPRS) bez dalších dodatečných zařízení, a který plnohodnotně nahrazuje mobilní telefonní přístroj. Nevýhodou u notebooku je vyšší pořizovací cena. Naopak z pohledu tvorby obsahu je výhodné přistupovat k informacím přes výše uvedená zařízení, protože se nemusí měnit forma prezentovaných informací, tj. k informacím můžeme beze změny struktury a formy přistupovat přes internetový prohlížeč v podnikovém počítači nebo přes prohlížeč v PDA zařízení na služební cestě.

A co mobilní přístroj? Ten vlastní v naší republice přes 6 miliónů lidí[3] a počet uživatelů se bude nadále zvyšovat. Kromě samotného volání nabízejí mobilní telefony řadu dalších možností, jak získat potřebné informace – technologie WAP[4] dovoluje prohlížení zjednodušených verzí webových stránek přímo na displeji mobilního telefonu, technologie GPRS poskytuje rychlé datové přenosy.

Kromě toho má skoro každý člověk přístup k pevné telefonní lince, což představuje dohromady s mobilními telefony obrovský lidský potenciál pro využití. Ne každý si může dovolit drahé PDA zařízení, ne každý si může dovolit koupit vlastní počítač, ne každý má mobil s WAP nebo GPRS, ale skoro každý má možnost použít obyčejný telefon, z kterého může kdykoliv v případě potřeby zavolat.

1.1 Hlasová komunikace

Další velkou výhodou přístupu přes telefonní zařízení je *hlasová komunikace*. Mluvená řeč je základním a nejdůležitějším prostředkem přenosu informace mezi inteligentními bytostmi, proto přirozeně existuje snaha o její využití i při komunikaci s technickými prostředky, které nás stále více obklopují i v běžném životě.

Hlavní výhody hlasové komunikace jsou:

[1] technologie umožňující připojení k síti Internet přes mobilní telefon

[2] modul pro připojení do GSM sítě, přes kterou v dnešní době komunikuje absolutní většina mobilních telefonů

[3] údaj z března 2002

[4] WAP (wireless application protocol) – obdoba služby World Wide Web pro jednoduchá bezdrátová za řízení

- nezávislost na přímém kontaktu – přístup k informacím je možný odkudkoliv a kdykoliv, jedinou podmínkou je telefonní přístroj
- možnost využití standardních přenosových sítí
- uvolnění ostatních smyslů a rukou člověka – tato vlastnost hlasové komunikace ještě více zvyšuje mobilitu přístupu k datům

Na druhou stranu je nutné uvést i řadu nevýhod, které sebou přináší hlasová komunikace:

- nižší rychlost přenosu informace vzhledem k textové podobě
- nelze získat okamžitý náhled na celou informaci
- detailní informace je postupně zapomínána

Stále byla řeč pouze o důležitosti získávání informací, ale samozřejmě lze nalézt i jiné využití pro hlasovou komunikaci, které jsou neméně důležité:

- automatické zpravodajské, informační a spojovací služby (např. objednávání lístků do kina, získávání výsledků přijímacích zkoušek, automatická podniková spojovatelka, zamluvení letenky a další)
- řízení a ovládání strojů hlasovými povely
- jazykové překladače
- diktování textu
- pomůcky pro postižené

Všeobecnou snahou je výše uvedené aplikace provádět bez účasti člověka, tedy pouze na základě komunikace s počítačem. To samozřejmě není vždy možné a často je potřeba řešit problémové operace za pomoci operátora-člověka. Ale i tak většinu práce provede operátor-počítač.

1.2 Přínosy

Dosud jsme stále mluvili o výhodách či nevýhodách hlasové komunikace (resp. přístupu přes telefonní síť) z pohledu uživatele, ale ještě jsme se nezmínili o tom, jaký přínos či problémy to přináší poskytovateli dané služby. Poskytovatele služeb můžeme rozdělit do dvou skupin – v první skupině se budou nacházet ty společnosti, které pomocí hlasové komunikace s počítačem budou chtít ušetřit své náklady na provoz vnitřních systémů, jako jsou např. systémy typu automatická spojovatelka nebo systémy pro poskytování obsahu o daném produktu či službě (možná náhrada callcenter). Do druhé skupiny potom zařadíme ty společnosti, které v rámci svých eBusiness[5] řešení využívají hlasových služeb pro komunikaci se zákazníky.

Pro nás je z pohledu poskytování služeb zajímavější druhá skupina, protože společnosti v první skupině pouze mění standardní formu komunikace člověk-člověk na komunikaci typu člověk-počítač za účelem ušetření nákladů a zautomatizování přístupu k informacím. Naopak společnosti ve druhé skupině dále rozšiřují své komunikační možnosti za účelem oslovení nových a udržení stávajících zákazníků či obchodních partnerů.

Většina společností v poslední době investovala nemalé částky do rozvoje své infrastruktury, zejména do rozvoje informačních technologií[6]. Samozřejmostí v dnešní době je přístup k informacím přes Internet (Web).

[5] souhrn nástrojů a služeb společnosti, které navazují na vnitropodnikové systémy a které jsou poskytovány třetím stranám (obchodním partnerům – B2B, spotřebitelům – B2C) na základě moderních komunikačních technologií – Internet, mobilní přístroj.

[6] souhrnný název pro technologie „internetového věku“ – podnikové CRM systému, systémy pro automatickou správu objednávek atd.

Pokud již firma má vybudovanou infrastrukturu s napojením na Internet, přechod na hlasovou komunikaci není z technického pohledu příliš složitý a navíc to sebou přináší nejednu výhodu:

- **společný kanál pro komunikaci se zákazníkem** – pokud bude zákazník s firmou komunikovat libovolným způsobem (Internet, telefon), měl by vždy dostat stejné informace. Obě formy komunikace využívají stejná data, stejné nástroje, stejné systémy, pouze koncová informace je prezentována v odlišné formě.
- **redukce nákladů** – postup zmíněný v předchozím bodě dovolí společnosti bez výrazného nárůstu nákladů poskytnout další komunikační kanál a tím oslovení dalších potencionálních zákazníků.
- **společný vývoj a řízení**

Na konec této podkapitoly bych ještě uvedl reálný příklad využití komunikace přes telefon.

Zákazník jede v autě, poslouchá rádio a během cesty uslyší zajímavou upoutávku na právě vydané CD jeho oblíbené hudební skupiny. Přístup k Internetu samozřejmě nemá a navíc je zatížen řízením svého auta. Během řízení zavolá na číslo v upoutávce a přes operátora-počítač si objedná CD.

Tento příklad si lze samozřejmě představit i bez komunikace s počítačem, ale za předpokladu omezení následujících výhod:

- operátor-počítač je 24 hodin denně připraven vyhovět zákazníkovým přáním
- operátor-počítač umí komunikovat s více zákazníky najednou
- vše probíhá zcela automaticky v závislosti na stavu vnitropodnikového systému (např. stav CD na skladě)

1.3 Současný stav poskytování hlasových služeb

Výhody hlasové komunikace si již uvědomila celá řada společností. Probíhá bouřlivý vývoj systémů pro automatické rozpoznávání řeči (ASR[7]), automatickou konverzi textu na řeč (TTS[8]) a řízení telefonních spojení. Vedle toho vznikají nové platformy, které budou umožňovat propojit všechny výše zmíněné systémy dohromady a tak nabídnout zákazníkovi ucelené řešení.

Především to jsou telekomunikační firmy (AT&T, Lucent, Motorola, Nokia) a technologické firmy (IBM, Intel, Sun, Microsoft, SpeechWorks, Nuance), které investují nemalé částky do vývoje a rozvoje hlasových systémů.

Vzniká řada poskytovatelů (voxbuilder, Tellme studio) hlasových systémů, kteří nabízejí zadarmo své hlasové systémy k vyzkoušení za účelem získání zákazníků pro placený provoz.

Již v dnešní době se můžeme setkat s celou řadou hlasových aplikací:

- automatická spojovatelka
- čtení emailů
- rezervace letenek, lístků do divadla, taxíka apod.
- přístup k podnikovým datům
- získání výsledků přijímacích zkoušek

[7] ASR – automatic speech recognition
[8] TTS – text-to-speech

Hlasové služby jsou nejrozšířenější v Americe a postupně se přes západní Evropu (Velká Británie, Itálie, Německo) dostávají i do naší republiky. I u nás již vznikla řada zajímavých projektů – uvedl bych např. přístup k emailové schránce přes telefon (VoiceGate, Paegas s aplikací Click Box), infocity (TUL) nebo přístup pomocí telefonního spojení k výsledkům přijímacího řízení na Západočeské univerzitě v Plzni.

1.4 Budoucnost hlasových služeb

Vývoj hlasových služeb bych přirovnal k vývoji a rozšíření Internetu v nedávné době. Dnes již existuje celá řada z technického pohledu dobře fungujících systémů, které umožňují hlasovou komunikaci a je tedy na čase začít se zaváděním těchto systémů do reálného života. Z počátku bude snaha o maximální nasazování hlasových aplikací za účelem oslovení zákazníka a tím získání lepšího postavení na trhu, případně za účelem ušetření provozních nákladů, ale teprve čas ukáže, v jakých oblastech si hlasové aplikace najdou své místo.

Na rozdíl od vývoje Internetu to nebude tak bouřlivý rozvoj, protože řada společností a hlavně investorů bude opatrnější v důsledku internetové krize[9] v poslední době, ale i tak věřím ve velký rozvoj a užitek těchto systémů již v blízké budoucnosti.

Abych se mohl v dalších kapitolách podrobněji věnovat jazyku VoiceXML, je nutné si přiblížit jazyk XML.

[9] podnikání přes Internet nesplnilo očekávání, které do něj na počátku většina společností a investorů vložila.

2 XML

Jazyk VoiceXML je založen na základech obecného jazyka XML. Pro práci s VoiceXML je tedy nutné znát syntaxi tohoto jazyka a znát jeho možnostmi.

2.1 Odkud se vzalo

Počítače byly již od samotného počátku využívány pro přípravu a publikování textu. Situace v šedesátých letech však byla dosti vzdálená té dnešní. Pokud se na počítačích připravovaly dokumenty pro profesionální tisk – knihy, časopisy, apod., výsledek se pomocí osvitové jednotky přenesl na film, ze kterého pak tiskárny dokázaly vyrobit knihu nebo časopis. Osvitové jednotky tehdy vyrábělo několik firem a každá z nich používala vlastní jazyk pro ovládání jednotky. Dokumenty pro sazbu se tedy připravovaly tak, že se přímo do textu vepisovaly speciální řídící sekvence pro ovládání té které osvitové jednotky. Jednou vytvořený dokument byl tak úzce svázán s výstupním zařízením konkrétního výrobce. Jeho převedení pro použití na konkurenční osvitové jednotce rozhodně nebylo jednoduchou záležitostí. V dnešní době, kdy všechny osvitové jednotky rozumí formátům PostScript a PDF to zní neuvěřitelně, ale skutečně to tak tehdy bylo.

Stav to rozhodně nebyl ideální a mnoho lidí si to uvědomovalo. Vzniklo proto několik systémů, které problém nekompatibility různých výstupních zařízení řešily. Princip byl většinou jednoduchý – v dokumentu se používaly některé obecné příkazy, které se pak pomocí speciálních konvertorů převedly do jazyka srozumitelného pro konkrétní zařízení. Dalo by se říci, že se jednalo o obdobu ovladačů různých výstupných zařízení, jak je známe dnes. Pokud jsme chtěli dokument vytisknout na nějakém novém zařízení, stačilo sehnat příslušný konvertor. Samotný dokument se měnit nemusel.

Mezi nejrozšířenější z těchto systémů patřily bezesporu *troff*[10] a *TeX*. Důležité je, že oba dva jazyky byly čistě prezentační – dalo se pomocí nich určit, jak se mají jednotlivé části textu formátovat. Troff používal poměrně kryptické dvouznakové příkazy. Oproti tomu byl TeX velice uživatelsky přívětivý – umožňoval definici maker ve vlastním programovacím jazyce a nekladl žádná omezení na délku jednotlivých názvů. Šlo tak vytvářet přehledné a srozumitelné zdrojové zápisy dokumentů. Zdrojový kód v TeXu je poměrně snadno čitelný, makropříkazy se míchají s textem.

Dokument v TeXu může obsahovat různé formátovací příkazy – můžeme např. přepnout {\it} na kurzívu nebo {\bf} na tučné písmo.

Pro účely formátování textu pro tisk je v mnoha směrech TeX dodnes nepřekonán a stále se používá. Vývoj TeXu se nezastavil – dnes lze například upravenou verzi původního TeXu generovat dokumenty ve formátu PDF, v několika komerčních programech pro sazbu textu je zaintegrováno jádro TeXu.

Programy jako TeX se však hodí pouze pro zpracování dokumentů, které se mají ve výsledku tisknout. Hlavně kvůli tomu, že nabízejí příkazy, které umožňují měnit druh použitelného písma, způsob zarovnání a nepřeberné množství dalších parametrů. S rozmachem Internetu a dalších médií (např. CD-ROM) vznikla potřeba jedny a tytéž informace prezentovat mnoha způsoby – kvalitním tiskem na papíře, jako hypertextovou příručku na CD-ROMu či sadu provázaných webových stránek. Pro tyto účely je však potřeba znát logickou strukturu dokumentu. Musíme vědět, že toto je nadpis, a toto zase popis obrázku. Konkrétní velikost písma a způsob formátování záleží až na tom, zda chceme produkovat tištěnou knihu nebo multimediální CD-ROM.

[10] http://www.kohala.com/start/troff/troff.html

Potřebujeme tedy jazyk, který umožní označit *význam* jednotlivých částí textu a ne jejich *vzhled*. Jedním z těchto „samopopisných“ jazyků označovaných jako značkovací jazyky (markup languages) je i XML.

Zřejmě prvním známým značkovacím jazykem byl GML (Generalized Markup Language), který vytvořili pánové Goldfarb, Mosher a Lorie při práci na systému pro uchování a následné využití právních textů pro IBM. Museli se tehdy vypořádat s nekompatibilitou jednotlivých systémů a programů a nejsnazší cesta vedla právě přes vytvoření nějakého obecného značkovacího jazyka.

Princip GML se osvědčil a v 80. letech se začala na základě GML vyvíjet standardizační organizace ANSI jazyk, který umožňoval definici vlastních značkovacích jazyků – uživatel si dle potřeb mohl vytvořit vlastní sadu značkovacích značek vhodnou pro daný druh dokumentů. Tou dobou se sdružení GCA (Graphics Communications Association) snažilo vytvořit standardní formátovací jazyk GenCode použitelný na širokém spektru zařízení. Mnohé cíle obou projektů byly podobné, a proto se obě aktivity spojily. Výsledkem byl jazyk SGML[11] (Standard Generalized Markup Language), který je definován v ISO normě 8879 z roku 1986.

Jazyk SGML je skutečně hodně obecný – samozřejmě umožňoval definici vlastních značkovacích jazyků (sad značek a jejich vzájemných vztahů) pomocí tzv. definic typu dokumentu (Document Type Definiton - DTD[12]). Navíc měl spoustu volitelných parametrů – počínaje maximální délkou názvů značek a konče určením znaků použitelných jako oddělovače značek od textu. Komplexnost standardu SGML poněkud zbrzdila jeho praktické využívání. Velkou podporou pro SGML bylo americké ministerstvo obrany, které od svých dodavatelů vyžadovalo dokumentaci k výrobkům právě ve formátu SGML. Důvod byl zřejmý – bylo třeba, aby byla dokumentace použitelná v poměrně dlouhém období. Nebylo možné tedy použít nějaký proprietární formát textového procesoru, který se každých pár let mění.

Asi nejznámější aplikací SGML je jazyk HTML[13] (Hypertext Markup Language), který se používá pro tvorbu webových stránek. To jaké značky můžeme na stránkách používat určuje DTD, které je pro každou verzi HTML poněkud odlišné.

V polovině 90. let došlo k paradoxní situaci. Jazyk HTML si získal velkou oblibu díky své jednoduchosti, která byla v ostrém kontrastu s komplexností SGML. Ukázalo se však, že pevně daná skupina značek, které HTML používá, už nestačí. Pro účely vyhledávání a vůbec efektivnější výměny dat by bylo lepší možnost používat vlastní značky, které by přesně vymezily význam textu. Požadavek by tedy mohl bez problémů splnit jazyk SGML.

Jak již bylo zmíněno, standard SGML je dost komplexní a jeho úplná implementace velice náročná. Přitom se během deseti let SGML ukázalo, že je využívána pouze část jeho možností. Tato nejdůležitější podmnožina SGML proto byla vybrána jako nový jazyk, který dovede Web do třetího tisíciletí. Ten dostal jméno XML (eXtensible Markup Language). Jedná se o podmnožinu SGML, která si zachovává možnost definování vlastních DTD, a tedy značek pro jednotlivé skupiny dokumentů. Na rozdíl od SGML je mnoho parametrů předem určeno a nelze je měnit – délka názvů značek, použité oddělovače a speciální znaky atd. XML už rovnou počítá s podporou všech možných jazyků, takže není tak úzce svázáno s angličtinou jako většina předchozích počítačových technologií. Navíc je syntaxe zápisu dokumentů v XML oproti SGML poměrně přísná, což umožní mnohem snazší a levnější vývoj aplikací, které mohou s XML pracovat.

XML pochází z oblasti, která se zaměřuje na uchování a zpracování textových dokumentů. Pro tyto účely se XML výborně hodí. Mnoho velkých i malých firem vyrábějící software,

[11] A Gentle Introduction to SGML - http://www.uic.edu/orgs/tei/sgml/teip3sg/
[12] XML DTD Tutorial - http://www.xml101.com/dtd/
[13] http://www.w3.org/MarkUp/

hardware nebo třeba letadla používá pro tvorbu dokumentace systémy založené na XML nebo SGML.

Elektronické publikování dokumentů však není jedinou doménou XML. Značky umožňují v dokumentu zachytit důležité informace o struktuře a významu. Není proto problém do XML dokumentu uložit například obsah z relační databáze.

O dokumentech je lépe uvažovat spíše jako o nosičích informací – není už tak důležité, do jaké míry jsou v nich data strukturované. Některé aplikace pracují s dokumentem, který je filosofickou esejí, jiné za dokument považují řadu čísel s burzovními indexy.

2.2 Co přináší XML nového

Zda způsobí XML skutečnou revoluci ve způsobu práce s informacemi pro každého uživatele osobního počítače, ukáže jen čas. XML na to rozhodně ambice a možnosti má.

2.2.1 Standardní formát pro výměnu a sdílení dat

Dnešní doba přeje komunikaci. Komunikace není nic jiného, než výměna informací. V dnešním globálním světě není možné pro výměnu dat používat nějaké proprietární formáty, které jsou svázány s konkrétním softwarem nebo hardwarem. Není vhodné posílat informace ve wordovském dokumentu DOC, protože je nebude možné přečíst na unixovém počítači. Podobné problémy bude mít např. centrála nadnárodní společnosti používající tzv. americkou verzi kancelářského balíku, která dostane výroční zprávu české pobočky ve formátu T602. Je potřeba používat nějaký jednoduchý otevřený formát, který není úzce svázán s nějakou platformou.

Takovým formátem je například XML. Otevřený formát je to proto, že jeho specifikace je každému k dispozici zdarma na serveru konsorcia W3C[14], která se stará i o mnoho dalších technologií souvisejících s Webem. To je velký rozdíl oproti firemním formátům, k nimž není k dispozici žádná dokumentace a navíc se jedná v porovnání s XML o velice složité, často binární formáty.

Práci s XML usnadňuje i to, že celý formát je založen na obyčejném textu. I když pro většinu lidí zůstane XML kód skryt a budou ho používat pouze jako aplikaci pro vzájemnou komunikaci, není problém kdykoliv otevřít XML dokument v Poznámkovém bloku nebo jiném jednoduchém textovém editoru a několik potřebných úprav provést ručně. Použití textového formátu může někomu připadat jako zbytečné plýtvání místem. Dnes se však mnohem větší důraz klade na srozumitelnost a snadnou práci s daty – to, zda ušetříme několik kilobajtů paměti, dnes již nikoho příliš netrápí. Navíc většina protokolů pro síťovou komunikaci (včetně protokolu http používaného na Webu) umožňuje zcela transparentně pro potřeby přenosu data zkomprimovat a u příjemce zase dekomprimovat do původní podoby.

2.2.2 Mezinárodní podpora

XML je asi vůbec první formát, který hned od samého začátku myslel na potřeby i jiných jazyků než je angličtina. Jeho znaková sada používá ISO 10646 nazývaná též UNICODE[15]. ISO 10646 je 32bitová znaková sada, která dokáže pojmout všechny dnes používané znaky všech jazyků. V XML je proto možné vytvářet dokumenty, které obsahují texty v mnoha jazycích najednou – můžeme kombinovat např. češtinu, angličtinu, ruštinu, arabštinu a korejštinu zcela dle libosti. Pokud by dokumenty obsahovaly pouze český text, bylo by ukládání přímo v ISO 10646 zbytečné plýtvání místem. XML dokument proto může být v libovolném kódování (např. windows-1250, iso-8859-2). Kódování je však v každém

[14] http://www.w3c.org
[15] http://www.unicode.org

dokumentu přesně určeno, takže odpadají problémy s konverzí z jednoho kódování do druhého. Každému je tak ihned jasné, v jakém kódování dokument je.

2.2.3 Vysoký informační obsah

Pomocí XML značek je možné v dokumentu vyznačit význam jednotlivých částí textu. Je možné říci „toto je název výrobku, toto zase telefonní číslo a toto je číslo našeho účtu“. Dokumenty obsahují mnohem více informací, než kdyby se používalo prezentační značkování – toto je tučným písmem Arial o velikosti 12 bodů zarovnané vlevo.

XML dokumenty jsou informačně bohatší. Toto lze samozřejmě s výhodou využít v mnoha oblastech. Největší přínos bude samozřejmě pro vyhledávání. Dnešní vyhledávací služby Internetu jako Altavista podporují pouze fulltextové vyhledávání. Zadáme hledaná slova a doufáme, že se nám vrátí dokumenty, které chceme. Pokud bychom mohli určit, že např. hledané slovo nás zajímá ve významu název firmy, bylo by při použití XML a vhodném označkování hledání mnohem přesnější.

2.2.4 Snadná konverze do dalších formátů

V mnoha případech je potřeba XML dokument zobrazit na nějakém běžném médiu – na obrazovce, na papíře. V tomto případě je už nutné přesně ovlivnit, jak se obsah jednotlivých značek zobrazí. XML samo o sobě žádné takové prostředky nenabízí. Existuje však naštěstí hned několik *stylových jazyků*, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují jak se dokument převede do jiného formátu se říká *styl*.

Výhodou je, že jeden styl můžeme aplikovat na mnoho dokumentů stejného typu. Dosáhneme tak jednotného formátování. Zároveň můžeme na jeden dokument aplikovat několik různých stylů. Jedním stylem vygenerujeme postscriptový soubor pro naše DTP studio, druhým HTML kód pro zařízení na naše stránky a třetím například jen obsah dokumentu, který pošleme mailem svému nadřízenému.

Stylových jazyků existuje dnes několik. Mezi nejznámější patří asi kaskádové styly (CSS)[16]. Ty lze použít pouze pro jednoduché formátování, které dobře poslouží pro zobrazování dokumentu na obrazovce v XML editoru nebo v prohlížeči. Pro náročnější aplikace slouží jazyk XSL[17] (eXtensible Stylesheet Language). Ten umožňuje před samotným zformátováním dokument různě upravovat a transformovat (části dokumentu např. vypustit nebo naopak automaticky vygenerovat obsah dokumentu). Společně s XML lze použít i velice výkonný, i když pro některé aplikace příliš složitý, jazyk DSSSL[18] (Document Style Semantics and Specification Language), který byl původně vyvinut pro potřeby jazyka SGML.

2.2.5 Automatická kontrola struktury dokumentu

XML umožňuje definovat si vlastní sadu značek, které se budou v dokumentu používat. Tuto možnost samozřejmě není nutné využít – lze používat libovolné značky. Pokud je však předem pomocí DTD definováno, jaké značky může dokument obsahovat, lze zcela automaticky kontrolovat, zda dokument obsahuje pouze povolené značky. Programu, který kontroluje správnost XML dokumentů, se říká *parser*. To má velký význam i při vývoji aplikací. Pro čtení dat je možné použít parser, který automaticky detekuje většinu chyb v datech – programátorům to velmi ušetří práci.

[16] http://www.w3c.org/Style/CSS/
[17] http://www.w3c.org/Style/XSL/
[18] DSSSL Online Application Profile – http://www.ibiblio.org/pub/sun-info/standards/dsssl/dssslo/do960816.htm

DTD není jediný jazyk, který umožňuje definovat značky použitelné v dokumentech. DTD se hodí pro popis formátů, které reprezentují především textové dokumenty. Neobsahuje však nástroje na kontrolu různých typů dat jako čísla, měnové údaje, údaje o datu a čase.

Příliš svobody může i škodit. Je sice hezké, že si každý může pojmenovat značky dle svého uvážení, ale to zase přinese problémy při vyhledávání informací. Někdo název firmy označí pomocí značky `<název>`, někdo pomocí `<obchodníNázev>`, nebo třeba jako `<NázevFirmy>`. Jak se s tím pak má vyhledávací stroj vypořádat? Existují proto různé skupiny a sdružení, které vydávají DTD nebo schémata, jenž by se měla používat v dané oblasti. Nejde přitom o nic jiného, než se shodnout na několika značkách, které se budou standardně používat pro označování určitých částí dokumentu. Dnes u možnosti zachycovat informace o složitých chemických strukturách nebo astronomických údajích.

Velkou výhodou XML je to, že v jednom dokumentu můžeme používat najednou nezávisle na sobě několik druhů značkování díky tzv. *jmenným prostorům* (namespaces). Můžeme tak vytvářet dokumenty, které používají značky definované pro naše specifické účely a pouze části dokumentu důležité pro vyhledávání označkujeme navíc pomocí nějakého standardizovaného DTD nebo schématu.

2.2.6 Hypertext a odkazy

XML samozřejmě umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty. Nabízí však mnoho možností nad rámec odkazů v HTML. Můžeme vytvářet i vícesměrné odkazy, které spojují více dokumentů dohromady. Užitečná je i možnost uložení odkazů zcela mimo dokumenty, kterých se týkají. Tímto způsobem lze vytvářet různé anotace a komentáře k již existujícím stránkám.

Tvorba odkazů je dnes popsána ve třech standardech – XLink[19], XPointer[20] a XPath[21].

XPath (XML Path Language) je jazyk, který umožňuje adresovat jednotlivé části dokumentu. Jeho možnosti dále rozšiřuje jazyk XPointer (XML Pointer Language). Xpointer se používá k určování jednotlivých částí dokumentu ve stylu: „zajímá mě první odstavec třetí kapitoly“. Není proto potřeba všechny části dokumentu, na které chceme odkazovat, explicitně označovat pomocí návěstí jako v HTML.

XLink (XML Linking Language) je samotný jazyk pro tvorbu odkazů. Jednotlivé dokumenty se samozřejmě určují pomocí jejich URL adresy, za kterou lze uvést ještě XPointer část pro přesnější určení části dokumentu.

2.3 Základy jazyka XML

2.3.1 Elementy

Elementy se v textu vyznačují pomocí tzv. *tagů*. Většině elementů odpovídají dva tagy – počáteční a ukončovací.

```
<para>Toto je obsah elementu para.</para>
```

Ukázka obsahuje jeden element para. Jeho obsah je vyznačen pomocí tagů `<para>` (počáteční tag) a `</para>` (ukončovací tag). Výše uvedená ukázka je asi nejjednodušší XML dokument, který vůbec lze vytvořit.

Názvy tagů se zapisují mezi znaky ‘<’ a ‘>’. Ukončovací tag má před svým názvem ještě znak ‘/’, aby se snadno odlišil od počátečního.

Některé elementy nemusejí mít žádný obsah. Lze je samozřejmě zapisovat tak, že za počátečním tagem uvedeme hned ten ukončovací.

```
<para>Toto je obsah elementu para.<br></br>A tohle taky.</para>
```

[19] http://www.w3c.org/TR/xlink/
[20] http://www.w3c.org/XML/Linking
[21] http://www.w3c.org/TR/xpath/

Není to však příliš pohodlné, a proto je možné v XML použít ještě jednu variantu tagu, která říká, že element nemá žádný obsah. Za jméno elementu v počátečním tagu se uvede znak '/'. Ukončovací tag se pak už neopakuje.

```
<para>Toto je obsah elementu para.<br/>A tohle taky.</para>
```

Každý XML dokument musí obsahovat pro všechny počáteční tagy odpovídající ukončovací tag nebo musí být počáteční tag zapsán jako element s prázdným obsahem.

2.3.2 Atributy

Elementy jsou základním stavebním kamenem každého dokumentu. U každého počátečního tagu můžeme použít ještě *atributy*. Atributy se používají k upřesnění významu elementu, k přidání dalších důležitých informací.

```
<para zabezpeceni="důvěrné">Nějaká tajná informace.</para>
```

V ukázce je k atributu zabezpečení přiřazena hodnota důvěrné. Hodnota atributu musí být vždy uzavřena do uvozovek nebo do apostrofů. U jednoho tagu lze použít více atributů najednou, stačí je oddělit mezerou.

```
<para zabezpeceni="důvěrné" autor="Jan Novák">Nějaká tajná
informace.</para>
```

2.3.3 Znakové entity

Vzhledem k tomu, že se znaky '<' a '>' používají pro oddělování tagů od okolního textu, není možné tyto znaky zapsat do dokumentu jen tak. Pro jejich zápis musíme použít tzv. *znakové entity*. Pro zápis znaku '<' je určena entita < a pro '>' to je >.

Pokud je potřeba uvnitř hodnoty atributu použít zároveň uvozovky i apostrofy, s výhodou lze použít odpovídající entity " a '.

```
<monitor úhlopříčka="15&quot;"/>
```

2.3.4 Kořenový element

Každý XML dokument musí být celý obsažen v jednom elementu.

```
<článek>
    <nadpis>Pokusný nadpis</nadpis>
    <odstavec>První odstavec</odstavec>
    <odstavec>Druhý odstavec</odstavec>
</článek>
```

Splňuje-li dokument všechna výše uvedená pravidla, je syntakticky v pořádku a je označován jako *správně strukturovaný (well-formed)*. S takovým dokumentem si „snadno poradí" všechny aplikace podporující formát XML.

2.3.5 Kódování znaků

Aby bylo možné v XML bez problémů zapisovat znaky libovolného jazyka, používá se znaková sada ISO 10646. Ta je 32bitová, což znamená, že může obsahovat až 2^{32} znaků. Do tohoto prostoru se kromě znaků anglické abecedy a české abecedy vejdou i pro nás mnohdy kuriózní znaky dalších abeced z celého světa. Protože by pro většinu dokumentů bylo přímé použití ISO 10646 zbytečným plýtváním – jeden znak by zabral čtyři bajty – používá se v XML standardně kódování UTF-8.

Na samotném začátku dokumentu lze určit kódování, které náš dokument používá. K tomu je určena XML deklarace:

```
<?xml version="1.0" encoding="windows-1250"?>
```

Právě v atributu encoding se uvádí použité kódování. Pokud pracujeme s Windows, umí editor pravděpodobně pouze kódování windows-1250, na Unixu to bude zase iso-8859-2. Použije-li se deklarace na začátku dokumentu, prohlížeč pak dokument interpretuje správně.

2.3.6 Automatická kontrola syntaxe

Pro zápis XML dokumentů platí jednoduchá pravidla, je však nutné je dodržovat. Pokud bude v XML kódu chyba, nebudou s ním vůbec žádné aplikace komunikovat. Existují proto samozřejmě nástroje, které umožňují správnost dokumentu zkontrolovat. Lepší editory umí syntaxi kontrolovat průběžně a nedovolí vytvoření dokumentu, který by nebyl správně strukturovaný.

Program, který kontroluje syntaxi XML dokumentu, se jmenuje *parser*. Může mít mnoho podob. Existují parsery, které lze spustit z příkazové řádky a jako parametr jim předat vytvořený dokument ke kontrole. Většina parserů své služby nabízí i pomocí standardizovaného API a lze je proto využít v dalších aplikacích.

2.3.7 Komentáře

Nebyl by to počítačový jazyk, který by neumožňoval do svého zdrojového kódu zapisovat komentáře. Pokud je potřeba v dokumentu něco vysvětlit nebo část textu na čas skrýt, je použití komentáře velmi vhodné. Komentář je součástí dokumentu, ale parsery jej ignorují a není dále zpracováván. Komentář se zapisuje mezi znaky <!-- a -->.

Komentář může obsahovat cokoliv kromě posloupnosti znaků --. V komentáři lze dokonce používat tagy apod. Jsou však zcela ignorovány.

2.3.8 Sekce CDATA

Pokud je potřeba do textu vložit větší kus textu, kde se hojně používají znaky se speciálním významem jako ‘<’, ‘>’ a ‘&’, je pohodlné je zapisovat pomocí znakových entit. Význam sekcí CDATA je patrný zejména v případech, kdy je součástí XML dokumentu kód nějakého programu nebo HTML či XML kód.

```
<script language="Javascript">
<![CDATA[
    for (i=0; i < 10; i++)
    {
          document.writeln("<p>Ahoj</p>");
    }
]]>
</script>
```

2.3.9 Instrukce pro zpracování

Občas se může stát, že do dokumentu je potřeba přidat důležité informace pro jeho zpracování, avšak tyto informace nemají povahu samotného obsahu dokumentu. Aby se zabránilo proprietárního rozšiřování, obsahuje XML standardní mechanismus pro přidávání nestandardních dat. Mechanismus se nazývá *instrukce pro zpracování (processing instructions).*

Instrukce se nejčastěji používá pro připojení stylu s definicí vzhledu, pro zařazení příkazů pro různé preprocesory, některé editory si pomocí speciální instrukce označí místo dokumentu, kde editace skončila a příště má zase začít. Každá instrukce má na svém začátku identifikátor, kterým si jednotlivé aplikace označují své instrukce. V jednom dokumentu se pak může kombinovat více různých druhů instrukcí. Instrukce pro zpracování mají velice jednoduchou syntaxi.

```
<?identifikátor data?>
```

Například připojení stylu k dokumentu lze provést pomocí instrukce

```
<?xml-stylesheet href="styl.css" type="text/css"?>
```

2.4 Parsery

Parser je programátorská knihovna, která usnadňuje čtení dokumentů. V současné době existuje několik desítek parserů, které se liší svými schopnostmi, podporovanými jazyky a způsobem práce s XML dokumentem.

2.4.1 Validující vs. nevalidující

První kritérium, které jednotlivé parsery odlišuje, je úroveň kontroly XML dokumentů, která se provádí při jejich čtení. Rozlišují se dva druhy parserů:

- nevalidující
- validující

Nevalidující parser při čtení dokumentu pouze kontroluje, zda je správně strukturovaný (well-formed). Ohlídá použití správného kódování, párovost tagů, uzavírání atributů do uvozovek apod.

Validující parser se chová jako nevalidující, ale navíc ještě kontroluje, zda dokument vyhovuje danému DTD nebo XML schématu. Již na první pohled je zřejmé, že validující parser je z podstaty věci mnohem pomalejší než validující. Pokud záleží na rychlosti aplikace a nepotřebujeme dokumenty validovat, je lepší používat nevalidující parsery.

2.4.2 Přístup k dokumentu

V dnešní době se pro přístup k XML dokumentu používají dva základní způsoby:

- událostmi řízené čtení
- práce se stromovou reprezentací dokumentu

Při zpracování řízeném událostmi volá parser během čtení dokumentu námi definované funkce. Funkce je vyvolána vždy při vzniku důležité události, která odpovídá určité části dokumentu – začátku elementu, obsahu elementu, konci elementu, komentáři, instrukci pro zpracování apod. Naší funkci jsou pak předány všechny potřebné parametry jako např. název elementu nebo seznam atributů elementu.

Výhoda událostmi řízeného přístupu je v jeho rychlosti a malé spotřebě paměti. Jednotlivé události jsou vyvolávány postupně, jak je čten dokument. Práce se stromovou reprezentací vyžaduje načtení celého dokumentu předtím, než s ním začneme pracovat. Čtení dokumentu řízené událostmi je obvykle rychlejší a má menší paměťové nároky než práce se stromovou reprezentací.

Při zpracování řízeném událostmi musí být celý XML dokument zpracován jedním průchodem. Není možnost přístupu k libovolné části dokumentu v libovolném čase. Tato omezení odstraňují parsery, které pracují se stromovou reprezentací. Při jejich práci je nejprve celý XML dokument načten do paměti, kde je uchován ve stromové struktuře. Jednotlivé elementy a atributy dokumentu tvoří uzly stromu, obsah elementů tvoří listy (uzly na nejnižší úrovni). Umístění uzlů ve stromu odpovídá struktuře dokumentu – vzájemnému zanoření elementů.

2.4.3 Rozhraní pro práci s XML dokumenty

Tvůrcům parserů samozřejmě nikdo nebrání v tom, aby jejich parser vybavili libovolným API. Některé parsery skutečně tuto možnost využívají a používají vlastní specifické API. Velká většina parserů naštěstí podporuje standardizovaná API. Dnes se používají tato dvě API:

- SAX (řízené událostmi)
- DOM (stromová reprezentace)

SAX – Simple API for XML

Rozhraní SAX[22] je založeno na řízení pomocí událostí (event-driven control). Pomocí rozhraní vytvoříme vazbu mezi událostmi, které generuje parser a naším kódem. V praxi to znamená, že si definujeme funkce, které se zavolají v okamžiku, kdy parser narazí na začátek elementu, na obsah elementu, na konec elementu, na komentář, na instrukce pro zpracování a podobně.

Rozhraní SAX dnes podporuje velké množství parserů, i když samotné rozhraní není definováno pomocí žádného standardu konsorcia W3C nebo jiné standardizační organizace. Rozhraní vzniklo společným úsilím vývojářů z diskusní skupiny xml-dev a představuje de facto standard. Parsery se rozhraní SAX drží, takže je možné v aplikaci zaměnit jeden parser za druhý.

SAX dnes existuje ve dvou verzích. Většina parserů podporuje SAX1. Novější SAX2 přidává podporu některých důležitých věci, z nichž nejvýznamnější jsou jmenné prostory (namespaces).

DOM – Document Object Model

Rozhraní DOM[23] je postaveno na zcela odlišném principu než SAX. Dokument je reprezentován jako stromová struktura, kdy každému elementu odpovídá jeden uzel stromu. Odpovídající uzly mají samozřejmě i komentáře, instrukce pro zpracování atd. Tomuto způsobu reprezentace se říká *grove* (Graph Representation Of property ValuEs). Rozhraní DOM obsahuje funkce, které umožňují celý strom dokumentu procházet, modifikovat jeho jednotlivé uzly, mazat je a přidávat. Na rozdíl od SAXu není nutné dokument procházet od začátku do konce, ale je možné se v něm pohybovat dle naší potřeby. Proto se rozhraní DOM uplatní v aplikacích, které provádějí náročnější operace s dokumentem – editory, prohlížeče a formátovače.

Rozhraní DOM je standardem z dílny konsorcia W3C. Původně byl DOM vytvořen zejména proto, aby nové verze prohlížečů podporující XML používaly stejný objektový model pro přístup k dokumentu ze skriptovacích jazyků jako třeba JavaScript. Bez tohoto standardu bychom si o kompatibilitě prohlížečů mohli nechat jenom zdát.

Většina parserů dnes podporuje DOM1. V současné době se dokončuje specifikace druhé verze. Pro práci s XML je nejpodstatnější doplnění podpory pro práci s jmennými prostory. Současně se připravuje i specifikace pro DOM3, který bude standardizovat i takové věci jako načtení/uložení XML dokumentu z/do souboru.

[22] http://www.saxproject.org/
[23] http://www.w3.org/DOM/

3 VoiceXML

Jazyk VoiceXML (Voice eXtensible Markup Language) byl vytvořen VoiceXML fórem, založeným firmami AT&T, IBM, Lucent a Motorola. Fórum bylo založeno pro vývoj a podporu jazyka VoiceXML, nového počítačového jazyka určeného pro zpřístupnění obsahu a informací z Internetu přes hlas a telefonní zařízení.

Veškeré bližší informace týkající se VoiceXML jsou popsány ve specifikaci jazyka a v návrhu W3C.

3.1 Vývoj jazyka

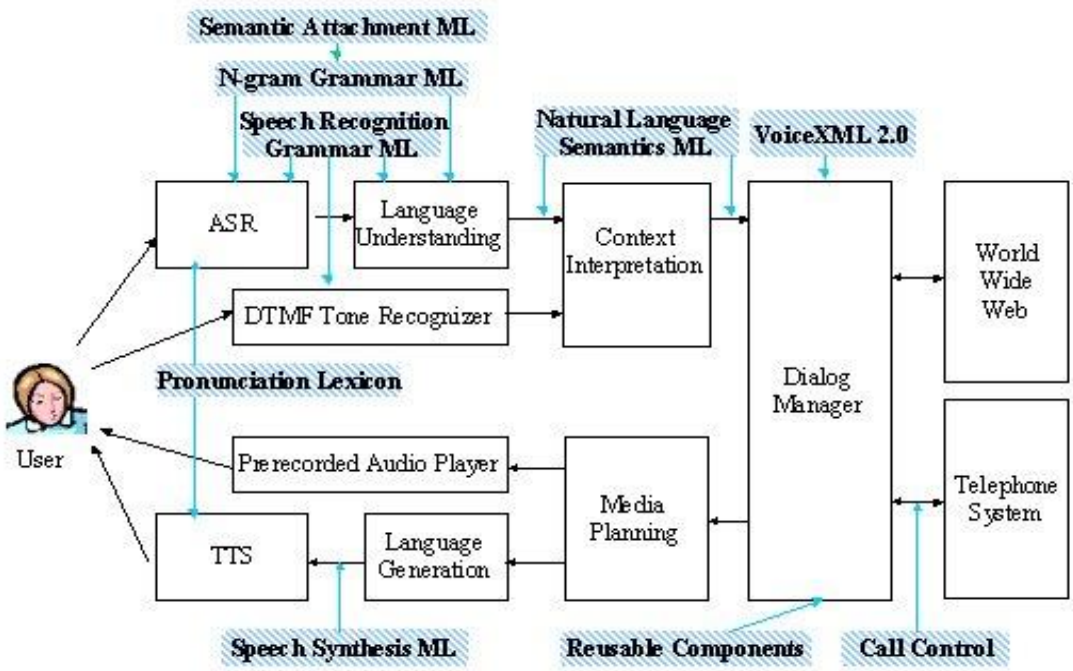
První návrh jazyka VoiceXML vznikl 17. srpna 1999. O necelý rok později 7. března 2000 byla veřejnosti prezentována první oficiální verze tohoto jazyka. Ta zahrnovala základní koncept jazyka – návrh struktury dokumentů, resp. dialogů, včetně vlastní správy telefonních spojů.

S postupem času, kdy se zvyšovaly nároky na informační systémy, hlavně na jejich robustnost a kvalitu, se začala připravovat další verze jazyka VoiceXML. Nyní již vývoj nové verze probíhá pod záštitou konsorcia W3C a její pracovní skupiny *Voice Browser working group*[24] za podpory VoiceXML fóra. Návrh druhé verze byl veřejnosti představen 23. října 2001.

3.1.1 Co nového s VoiceXML 2.0

VoiceXML 1.0 představoval jednotný jazyk zahrnující kromě vlastního popisu dialogů i popis řízení telefonních spojů (přijmutí a ukončení hovoru, přesměrování hovoru), gramatik (standardně podporovány gramatiky typu Java Speech Grammar Format, ovšem je možné použit i jiné) a značek pro řečovou syntézu (Speech Synthesis Markup Language). Druhá verze VoiceXML byla rozšířena do několika příbuzných samostatných jazyků, každý se zaměřením na konkrétní oblast, společně s cílem vytvořit množinu jazyků pro souhrnný popis dialogových systémů. Tak vznikl návrh W3C Speech Interface Framework, blíže znázorněný na následujícím obrázku.

[24] pracovní skupina, která se soustředí na vývoj jazyků pro vytváření hlasových systémů http://www.w3c.org/voice/



obrázek 3-1 W3C Speech Interface Framework

W3C Speech Interface Framework obsahuje následující části[25]:

Automatic speech recognizer (ASR) – přijímá hlasový vstup od uživatele a převádí jej na text. ASR užívá gramatiky pro rozpoznání slov hlasového vstupu, např. gramatiky ve formátu **Speech Grammar Markup Language.** Gramatiky mohou obsahovat elementy ze **Semantic Attachment Language**, který říká ASR, jak má vytvářet výstupní text na základě sémantických znalostí. Jiným typem gramatiky je statistická gramatika zapsaná v **N-gram Grammar Markup Language.**

DTMF tone recognizer – přijímá DTMF vstup (vstup přes tlačítka telefonu)

Language understanding – snaží se získat sémantický význam promluvy z textu produkovaného *ASR*. Tato komponenta také užívá gramatiky **Speech Grammar Markup Language** nebo **N-gram Grammar Markup Language**. Výstup je vyjádřen pomocí **Natural Language Semantics Markup Language**.

context interpreter – snaží se vylepšit sémantickou informaci promluvy na základě historie výstupů od komponenty *Language understanding*. Vstup i výstup je prezentovan pomocí **Natural Language Semantics Markup Language**.

dialog manager – tato komponenta řídí průběh celého dialogu - žádá uživatele o vstup, na základě vstupu a podle instrukcí zapsaných ve VoiceXML určuje co se bude dělat dále. V závislosti na získaném vstupu *dialog manager* může přejít na jiný soubor nebo může uživateli prezentovat informace. *Dialog manager* přijímá vstup specifikovaný pomocí **Natural Language Semantics Markup Language**. Dialogové skripty mohou využívat **Reusable components**, části jiných dialogů. *Dialog manager* může také získat skript pro

[25] návrh struktury je pouze demonstrativní a v konkrétních případech se může lišit. Snahou je ukázat nový přístup k návrhu hlasových dialogových systémů.

interpretaci z Internetu, odeslat data do databáze nebo může řídit telefonní systém za pomoci jazyka **Call Control Markup Language**, určeného pro správu a řízení telefonních spojů.

Media Planner – určuje, zda má být vstup prezentován pomocí syntetizované řeči nebo pomocí audio nahrávek.

Recorder audio player – přehrává audio soubory

Language generator – přijímá text od *Media Planner* a připravuje ho pro syntézu TTS. Text může obsahovat elementy vyjádřené pomocí **Speech Synthesis Markup Language**, které poskytují rady a návrhy, jak akusticky prezentovat daný text. Tyto elementy mohou být vytvářeny automaticky v jazykovém generátoru nebo ručně vloženy programátorem aplikace.

Text-to-Speech synthesizer (TTS) – vstup tvoří text z *Language generator*, který je převáděn na akustický signál, který uživatel pak slyší. Při syntéze je využito „rad“ ze **Speech Synthesis Markup Language.** TTS může také využívat *Pronunciation lexikon* k získání výslovnosti daného slova.

3.1.2 Výhled do budoucna

Jazyk VoiceXML si získal velkou oblibu mezi vývojáři a programátory hlasových aplikací. Spolu s ostatními jazyky prezentovanými ve W3C Speech Interface framework lze navrhovat a programovat opravdu velice rozsáhlé a robustní aplikace. Nyní hlavní snahou je jazyk VoiceXML standardizovat, protože jedině tak bude zaručena kompatibilita jednotlivých VoiceXML interpreterů na trhu. Ze strany VoiceXML fóra se připravuje certifikační program, který bude na základě testů nezávislé organizace (resp. společnosti) vydávat osvědčení ke správné implementaci interpreteru dle specifikace VoiceXML. Tedy jasnou snahou konsorcia W3C a VoiceXML fóra je plně sjednotit jazyk VoiceXML a plně zaručit přenosnost mezi odlišnými VoiceXML interpretery, což je pro další rozvoj a rozšíření jazyka nezbytné.

Vedle použití VoiceXML k vytváření hlasových aplikací se již začíná přemýšlet o využití v multimodálních aplikacích, tj. aplikacích umožňující více způsobů komunikace (hlas, klávesnice, obrazovka) najednou – uživatel zavolá z mobilního telefonu počítači-operátorovi, vyžádá si určité informace, které mu budou následně poslány na display mobilního telefonu.

Předpokládá se, že multimodální aplikace budou představovat další krok ve vývoji hlasových aplikací. O tom svědčí i fakt, že konsorcium W3C založilo pracovní skupinu s názvem Multimodal activity[26], jejíž hlavní náplní bude vývoj standardů pro multimodální aplikace.

3.2 Popis jazyka

3.2.1 Co je VoiceXML

VoiceXML je jazyk založený plně na standardu XML (viz. kapitola XML) určený pro popis dialogů mezi člověkem a strojem, který podporuje:

- výstup syntetizované řeči (text-to-speech)
- výstup souborů ve formátu audio
- rozpoznávání mluvené řeči
- rozpoznávání vstupu pomocí tónové volby
- nahrávání a ukládání mluveného vstupu
- některé telefonní možnosti jako například přepojení hovoru a zavěšení

[26] http://www.w3.org/2002/mmi/

Jazyk umožňuje zachytávání znakového a mluveného vstupu. Tato data může přiřazovat proměnným definovaným v dokumentu a podle nich má možnost rozhodovat o dalším chování celého systému.

3.2.2 Účel zavedení VoiceXML

Hlavním smyslem zavedení VoiceXML je velmi silná možnost vývoje webových aplikací pro zpřístupnění internetového obsahu pro nejrůznější hlasové aplikace. Autoři takových aplikací jsou díky VoiceXML oproštěni od nízkoúrovňového programování. Velice jednoduše umožňuje spojení hlasových a datových služeb.

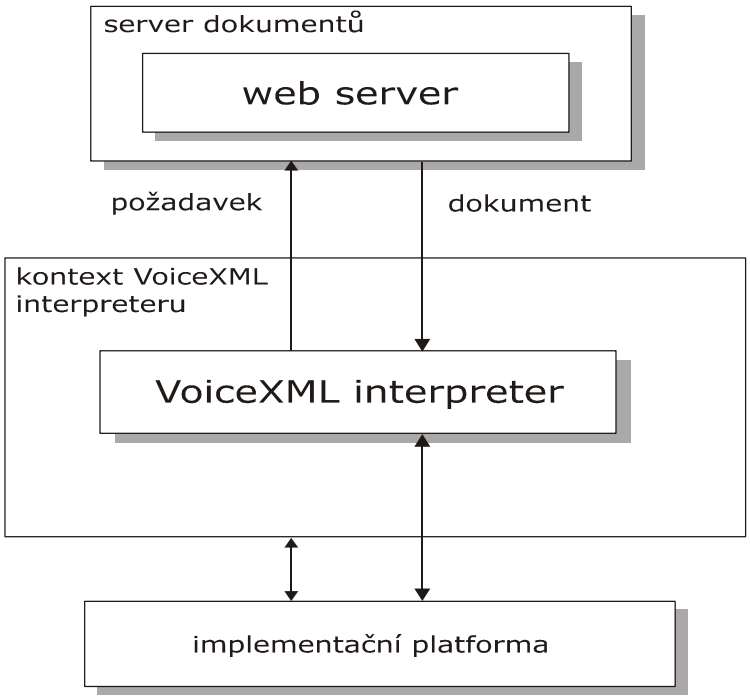
Hlasové služby jsou ve VoiceXML vedeny jako sekvence dialogů mezi uživatelem a aplikací. Dialogy jsou aplikaci poskytovány servery dokumentů (document server), které by měly být od samotné aplikace oddělené. Mohou to být například WWW servery. Servery dokumentů provádějí veškerou aplikační logiku, přistupují do databáze, zajišťují kontrolu přístupových práv k datům a jako výsledek vracejí dialogy ve formátu VoiceXML.

VoiceXML dokument detailně popisuje každý dialog, který je nakonec prováděn interpreterem tohoto jazyka. Uživatelský vstup, ať už pomocí tónové volby nebo hlasu, je vždy spojen do dalšího dotazu a je odeslán souborovému serveru. Server dokumentů pak odpoví dalším dialogem ve formátu VoiceXML, který je znovu interpretován a tím pokračuje dialog mezi uživatelem a celým systémem.

VoiceXML je tedy jazyk, který:

- minimalizuje nároky klient/server spojení spojením více takových interakcí do jednoho dokumentu (nejprve zjistí všechna potřebná data od uživatele a pak provede dotazy na server dokumentů).
- před autory výsledných hlasových aplikací skrývá nízkoúrovňové a platformově závislé detaily. Vývojáři takových aplikací se tedy například nemusí starat o nastavení telefonní karty, pouze pomocí VoiceXML navrhnou dialogy, pomocí kterých má aplikace s uživatelem komunikovat..
- podporuje přenos aplikací mezi jednotlivými aplikačními platformami. VoiceXML je obecný jazyk pro všechny platformy.
- je možné jednoduše a rychle využít pro vytváření jednoduchých i složitějších dialogů mezi uživatelem a systémem.

3.2.3 Architektura aplikace využívající VoiceXML



obrázek 3-2 Architektura VoiceXML aplikace

Server dokumentů (document server) zpracovává požadavky od klientské aplikace (VoiceXML Interpreter). Jako odpovědi na požadavky vytváří *VoiceXML dokumenty*, které jsou pak interpretovány *interpreterem* jazyka VoiceXML (VoiceXML interpreter). Kontext VoiceXML interpreteru může monitorovat uživatelský vstup zároveň s interpreterem. Například může vždy zachytávat některé speciální sekvence, které mění uživatelské nastavení, jako například hlasitost nebo nastavení syntézy řeči.

Implementační platforma je řízena kontextem VoiceXML interpreteru i samotným interpreterem. Kontext by měl být v interaktivní aplikaci zodpovědný za rozpoznání příchozího hovoru, získání prvního VoiceXML dokumentu atd. Implementační platforma generuje události jako reakce na akce uživatele (vstup řeči nebo tónové volby, zavěšení telefonu) a systémové události (timeout vstupu). Každá z těchto událostí je zpracována VoiceXML interpreterem, jak je určeno ve VoiceXML dokumentu. Ostatní události jsou pak zpracovány přímo kontextem VoiceXML interpreteru.

3.2.4 Požadavky na implementační platformu

Získávání dokumentů

Požadavky pro získávání dokumentů jsou generovány interpretací VoiceXML dokumentu. Jindy jsou generovány kontextem interpreteru jako reakce na události mimo platnost jazyka VoiceXML. Například jako reakce na nový příchozí hovor je generován výchozí dokument dialogu.

Audio výstup

Implementační platforma by měla podporovat audio výstup pomocí audio souborů a/nebo pomocí syntézy psaného textu na řeč (text-to-speech, TTS). Pokud podporuje obě možnosti, pak by měla být schopna volně spojovat TTS a audio soubory. Umístění audio souborů je v jazyce VoiceXML popsáno pomocí URI (Uniform Resource Indicator).

Audio vstup

Implementační platforma musí hlásit detekci znakového nebo hlasového vstupu. Musí tedy:

- hlásit znaky vložené uživatelem pomocí tónové volby (DTMF)
- dynamicky přijímat data pro gramatiky použité k rozpoznávání hlasového vstupu. Některé elementy VoiceXML obsahují data gramatik, některé se odkazují na tato data pomocí URI.
- ukládat hlasová data přijatá od uživatele

3.2.5 Struktura VoiceXML dokumentu a jeho interpretace

Každý VoiceXML dokument se primárně skládá z vysokoúrovňových elementů, které nazýváme *dialogy*. Existují dva druhy dialogů: *formuláře* (forms) a *menu* (menus). Dokument může obsahovat také další elementy jako meta, var, property, link, catch atd.

Formuláře – FORMS

Formuláře jsou klíčovým elementem VoiceXML dokumentu. Zpravidla obsahují:

- sadu *položek formuláře* (form items). Tyto položky jsou postupně zpracovány v hlavní smyčce algoritmu pro interpretaci formuláře (FIA). Položky formuláře lze rozdělit na *pole* (field items) a *řídící položky* (control items). Pole jsou položky, které uživatel během dialogu vyplní pomocí tónové volby nebo přímo mluvenou řečí. Řídící položky pomáhají kontrolovat vyplňování polí.
- deklarace proměnných formuláře, které nejsou pole. Uživatel jim tedy během dialogu nemůže přímo vložit hodnotu. Tato hodnota je většinou nějak vypočítána.
- deklarace pro obsluhy událostí
- akci *filled*. Je to blok procedurální logiky, který je zpracováván, když jsou všechny nebo jen některé pole formuláře vyplněny.

Interpretace formulářů – FIA

Všechny formuláře jsou interpretovány implicitním algoritmem. Tento algoritmus se nazývá „*Form interpretation Algorithm*“ (FIA). FIA obsahuje hlavní smyčku, která opakovaně vybírá položky formuláře a pokouší se o jejich vyplnění. Vybraná položka je vždy první položka v dokumentu, která ještě není vyplněna. Interpreter ve smyčce vždy zkontroluje, zda má příslušná položka již nějakou hodnotu a pokud ji ještě nemá, pak vyzve uživatele k jejímu vyplnění.

Každé zpracování jednotlivé položky formuláře vždy obsahuje:

- vybrání a přehrání jednoho nebo více elementů *prompt*. Tyto elementy určují, co má být přehráno a to jak pomocí TTS nebo přehráním audio souboru.
- zpracování uživatelského vstupu. Buď zpracování vstupu, který vyplní hodnotu pole nebo generování nějaké události.
- interpretace akce *filled*, která určuje co se má provádět s právě vloženými daty.

FIA končí v okamžiku, kdy interpretuje skok na jiný dialog (element *goto*) nebo zaslání dat serveru dokumentů (element *submit*).

Položky formuláře - pole (field items)

Pole jsou proměnné, které slouží ke vstupu dat od uživatele. Položky obsahují elementy *prompt*, které určují, co má být uživateli řečeno, položky *grammar*, které definují povolené vstupy a obsluhy událostí (*event handlers)*. Položky mohou také obsahovat element *filled*. Tento element určuje akci, která bude provedena ihned po vyplnění příslušné položky.

Položky formuláře jsou:

- `<field>` - položka, jejíž hodnota je získána pomocí tónové volby nebo rozpoznáním mluvené řeči
- `<record>` - položka, jejíž hodnota je audio vstup nahraný uživatelem, například nějaká hlasová zpráva
- `<transfer>` - pole, které přesměruje uživatele na jiné telefonní číslo
- `<object>` - toto pole spustí nějaký platformově závislý objekt s různými parametry
- `<subdialog>` - něco podobného jako volání funkce. Zavolá jiný dialog v dokumentu a po jeho zpracování se opět vrátí zpět.

Řídící položky (control items)

Existují pouze dva typy řídicích položek:

- `<block>` - procedurální sekvence příkazů, které se používají pro přehrání výstupu a pro různé výpočty. Nelze ji použít pro uživatelské vstupy.
- `<initial>` - tento element řídí počáteční interakci uživatele se systémem v dialozích, kde se počítač střídá s uživatelem ve vedení dialogu (mixed initiative dialogs).

Příklad formuláře

```
<?xml encoding="windows-1250" ?>
<vxml version="1.0">
    <form id="rodne_cislo">
          <block>
          Vítá Vás systém informující o výsledcích přijímacího řízení.
          </block>
          <field name="RC">
                <grammar src="digits.esgf"/>
                <prompt>Zadejte Vaše rodné číslo</prompt>
                <filled>
                      <submit next="zpracuj_rc.php" method="post"/>
                </filled>
          </field>
    </form>
</vxml>
```

Tento formulář uživatele nejdříve přivítá slovy „Vítá Vás systém informující o výsledcích přijímacího řízení“ a hned na to se zeptá na rodné číslo. Vstup od uživatele je pak odeslán serveru dokumentů na další zpracování. Vstup může být sekvence znaků (DTMF) nebo promluva podle pravidel uvedených v externí gramatice digits.esgf.

Menu – MENUS

Pokud potřebujeme vytvořit formulář, který obsahuje pouze jedno anonymní pole podle jehož vstupu je pak dialog přesměrován na jiný VoiceXML dokument nebo jinou část dokumentu, můžeme s výhodou využít možností *menu.*

Menu vždy vyzve uživatele, aby vybral jednu z nabízených možností a podle jeho volby přenese zpracování dialogu na jiné místo, například si vyžádá od serveru dokumentů jiný VoiceXML dokument.

Element choice

Menu obsahuje vždy několik elementů *choice*, které určují, jaká akce se má provést při konkrétním vstupu. Pokud je požadován vstup tónovou volbou, pak obsahuje atribut *dtmf,* jehož hodnota určuje, pro kterou hodnotu vstupu bude tato volba vybrána. Pokud uživatel

vybere jednu z možností, pak je přesměrován na další dialog, který je určen příslušnou hodnotou atributu *next* nebo je vygenerována událost definovaná atributem *event*.

Stejného výsledku jako při ovládání tónovou volbou lze dosáhnout, pokud hlasový vstup uživatele bude odpovídat definované promluvě, kterou lze také uvést v elementu *choice*.

Příklad menu

```
<vxml version="1.0">
  <menu>
    <prompt count="1">Jaké noviny chcete přečíst?</prompt>
    <prompt count="2">Prosím vyberte si z následující nabídky
<enumerate/></prompt>
    <choice dtmf="1" next="#prvni">Mladá fronta</choice>
    <choice dtmf="2" next="#druha">Lidové noviny</choice>
    <choice dtmf="3" next="#treti">Blesk</choice>
    <catch event=”noinput nomatch“>
       Vyberte si noviny, které chcete přečíst.
       <reprompt/>
    </noinput>
  </menu>
  <form id="prvni">
    <block>
      Vybrali jste si Mladou frontu.
      <disconnect/>
    </block>
  </form>
  <form id="druha">
    <block>
      Vybrali jste si Lidové noviny.
      <disconnect/>
    </block>
  </form>
  <form id="treti">
    <block>
      Vybrali jste si bulvární plátek Blesk.
      <disconnect/>
    </block>
  </form>
</vxml>
```

Při vstupu do tohoto dialogu se systém zeptá uživatele „Jaké noviny chcete přečíst?“. Pokud uživatel v zadaném časovém limitu nic nezadá, resp. neřekne nebo zadá špatně, bude upozorněn „Vyberte si noviny, které chcete přečíst“ a bude mu přehrána nabídka titulů novin: „Prosím vyberte si z následující nabídky. Pro Mladá fronta stiskněte jedničku, pro Lidové noviny stiskněte dvojku, pro Blesk stiskněte trojku“. Pokud si nyní správně vybere (např. Blesk), dialog bude pokračovat konstatováním „Vybrali jste si bulvární plátek Blesk“ s následným ukončením spojení.

Gramatiky (grammars)

Každý dialog má jednu nebo více gramatik. V *počítači řízené aplikaci* (machines directed application) má každý dialog svoji gramatiku aktivní pouze tehdy, když je daný dialog navštíven uživatelem. V aplikacích, které jsou řízeny počítačem i uživatelem současně (*mixed initiative applications*), mohou být gramatiky aktivní přes více dialogů, tzn. uživatel může na základě své promluvy řídit další postup v konverzaci. V situaci, kdy uživatel řekne něco odpovídající aktivní gramatice z jiného dialogu, zpracování dokumentu se přesune do tohoto dialogu spolu s promluvou od uživatele. Promluva je pak zpracována, jako kdyby byla řečena v tomto dialogu.

Linky (links)

Element *link* je součástí aplikací řízených uživatelem i počítačem zároveň. Určuje gramatiku, která je aktivní v oblasti umístění elementu *link* (*link* umístěný v elementu *field* bude mít svoji gramatiku aktivní pouze při návštěvě elementu *field*). Pokud uživatelův vstup odpovídá gramatice definované v *linku*, další zpracování dokumentu se přesune na místo určené atributem *next*. *Link* může také vyvolat událost.

Události (events)

VoiceXML definuje mechanizmus pro zpracování událostí. Události jsou výsledkem různých okolností, jako třeba, když uživatel nic neřekne nebo naopak řekne a promluva neodpovídá definované gramatice, objeví se nečekaná chyba v průběhu zpracování dokumentu (např. nepodaří se načíst další VoiceXML dokument, dokument neodpovídá syntaxi jazyka). Všechny události jsou zachytávány speciálními elementy (*catch elementy*):

- *noinput* – nebyl zadán žádný vstup
- *nomatch* – zadaný vstup neodpovídal žádné z nabízených možností
- *error* – v průběhu vykonávání skriptu se objevila chyba
- *help* – událost určená pro nápovědu

Příklad událostí:

```
<?xml version="1.0" encoding="windows-1250"?>
<vxml version="1.0">

   <property name="timeout" value="4"/>

   <help>Pro další pokračování v dialogu je nutné vybrat fakultu na
kterou jste dělal přijímací zkoušky.</help>
   <link event="help"><grammar>nápověda</grammar></link>

   <menu id="vyber_fakulty" dtmf="true">
      <prompt>Vyberte prosím fakultu. <enumerate/></prompt>

      <choice dtmf="1" next="vyber_oboru.php?fakulta=FAV">aplikované
vědy</choice>
      <choice dtmf="2" next="vyber_oboru.php?fakulta=FHS">humanitní
studie</choice>

      <noinput>
         <reprompt/>
      </noinput>
      <noinput count="3">
         <prompt>Program bude ukončen. Naslyšenou.</prompt>
         <disconnect/>
      </noinput>
      <nomatch>
         <prompt>Nesprávná volba.</prompt>
         <reprompt/>
      </nomatch>
      <nomatch count="3">
         <prompt>Program bude ukončen. Naslyšenou.</prompt>
         <disconnect/>
      </nomatch>
   </menu>
</vxml>
```

Uživatel je vyzván k výběru fakulty. Podle toho, jak se uživatel zachová se bude dále pokračovat v dialogu:

- uživatel vybere fakultu (stiskne jedničku nebo řekne „aplikované vědy“) – zpracování se přesune na další dokument
- uživatel nezadá žádný vstup – událost *noinput* bude vyvolána a zachycena elementem `<noinput>` (při prvním výskytu se pouze znovu přehraje žádost o výběr fakulty, při třetím se dialog ukončí).
- uživatel zadá špatný vstup – událost *nomatch* je vyvolána a zachycena elementem `<nomatch>`
- uživatel si neví rady a řekne „nápověda“ – událost *help* je vyvolána a zachycena elementem `<help>`. Poté co uživateli bude přečtena nápověda, bude se zase pokračovat v dialogu.

Vlastnosti (properties)

Vlastnosti nám dovolují v průběhu vykonávání dokumentu si blíže nastavit parametry pro rozpoznávání a syntézu.

Příklad vlastností:

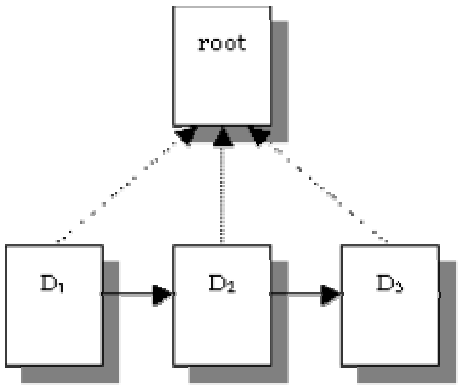
```
<?xml version="1.0" encoding="windows-1250"?>
<vxml version="1.0">

    <property name="timeout" value="4"/>

   <form id="registrace">
      <field name="un_cislo" modal="true">
         <property name="interdigittimeout" value="3"/>
         <property name="maxdigits" value="5"/>

         <grammar src="digits.esgf"/>
         <prompt>Zadejte prosím vaše univerzitní číslo.</prompt>
         <filled>
            <submit next="zpracovani_reg.php" method="post"
namelist="un_cislo"/>
         </filled>
    </form>
</vxml>
```

V rámci celého dokumentu je definována vlastnost *timeout*, která říká, jak dlouho má interpreter čekat na vstup od uživatele. Pokud v zadané době není nic řečeno (resp. zadáno), je vyvolána událost *noinput*.

V rámci elementu *field* jsou ještě více upřesněny parametry rozpoznávání – vlastnost *interdigittimeout* nastavuje povolenou časovou prodlevu mezi jednotlivými zadanými znaky (pouze pro DTMF), vlastnost *maxdigits* omezuje vstup na maximální počet pěti znaků.

Aplikace (applications)

Aplikace je množina dokumentů sdílející stejný aplikační kořenový dokument (application root document). Kdykoliv je uživatel v interakci s dokumentem v aplikaci, kořenový dokument je vždy přítomen. Kořenový dokument je stále v paměti zatímco uživatel se přesouvá po jednotlivých dokumentech aplikace. Pokud ovšem přestoupí na dokument z jiné aplikace, je změněn i kořenový dokument. Do kořenového dokumentu je vhodné si uložit proměnné, gramatiky nebo nastavení, které potřebujeme mít k dispozici ve všech dokumentech aplikace.



obrázek 3-3 Application root document

V další části diplomové práce se budu věnovat popisu interpreteru jazyka VoiceXML a aplikaci pro výsledky přijímacího řízení.