SIEMENS

ČSSZ

Standard metodiky vývoje

v. 1_0_19

Východiska

Dokument popisuje závaznou metodiku vývoje, je zasazen do prostředí vývoje aplikací pro ČSSZ a navazuje na projektovou metodiku a aktuální koncepční materiály ohledně architektury systémů. Popisuje závazná pravidla při tvorbě aplikací, cílem uvedených pravidel je dosažení

- Škálovatelnosti
- Udržovatelnosti
- Provozovatelnosti

vytvářených aplikací.

Dokument navazuje na „Cílový koncept“ a doplňuje části, které je potřeba z pohledu metodiky vývoje zdůraznit, zpřesnit, případně nebyly řešeny. Dále je nutno dodržovat směrnice České správy sociálního zabezpečení ohledně vývoje, správy a údržby SW , konkrétně Prováděcí pokyn vrchního ředitele úseku informačních a komunikačních technologií č. 7/2005 o pravidlech, zásadách a způsobu správy počítačových programů v ČSSZ a Prováděcí pokyn vrchního ředitele úseku informačních a komunikačních technologií č. 22/2006 o vývoji, provozu a údržbě jednotné aplikační softwarové podpory v ČSSZ

Úvod

Tento dokument je závazný pro všechny pracovníky Siemens Busines Services a všechny pracovníky subdodavatelů, kteří se podílejí na tvorbě aplikací pro ČSSZ (nejen pro programátory, ale pro všechny pracovníky). Cílem dokumentu je standardizovat obecné přístupy při tvorbě aplikací, tak aby bylo možné efektivně:

- integrovat aplikace mezi sebou a integrovat jednotlivé části aplikací
- podporovat aplikace v provozu
- přebírat a upravovat kód
- komunikovat s jednotlivými tvůrci částí systému

Pro jednotlivé případy je nutné dodržovat konvence schválené SBS, a to tak že přednost mají konvence SBS a nad nimi je vhodné používat interní konvence subdodavatele.

Pro tvorbu integrovaných aplikací musí všichni pracovníci velmi dobře znát pravidla používaných přístupů. Zde uvedená doporučení jsou povinná, jedná se však pouze o stručný výběr, ke správné tvorbě aplikací je potřebná zejména dobrá kvalifikace zúčastněných pracovníků.

Veškeré dokumenty jsou považovány za živé. Pokud si to situace vyžaduje, každý dokument může být v průběhu projektu modifikován se souhlasem zainteresovaných stran. Modifikace se dějí verzovaným způsobem, přičemž je nutné uchovávat všechny předchozí verze dokumentu.

Základním prostředkem komunikace o postupu projektu jsou seznamy pracovních položek.

Pro práci s dokumenty je využíván projektový portál, kde je zapnuta funkce verzování a případně i jednostupňové schvalování (v budoucnu možná vazba na workflow).

Pro okamžitou technickou komunikaci mezi členy týmu slouží e-mail (např. objevená chyba, přiřazená pracovní položka apod.)

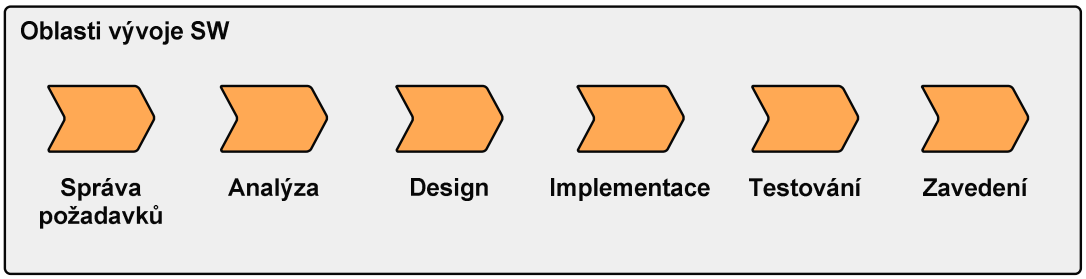
Management bude o postupu projektu informován prostřednictvím reportů zobrazujících klíčové metriky (postup práce, množství a úspěšnost testů, procento pokrytí kódu testy, míra změn v kódu apod.)

Dokument je vytvářen tak, aby nejprve standardizoval aktuálně prováděné a navazující činnosti, proto jsou některé kapitoly prázdné a nebo neúplné. Vše co je uvedeno v jednotlivých kapitolách je závazné, včetně odkazovaných dokumentů, rozpracované a neúplné kapitoly obsahují „bude doplněno“.

Zasazení standardů do prostředí ČSSZ

Je řešeno „Cílovým konceptem“ kapitola 7.1.1

Každý SW projekt prochází níže uvedenými oblastmi. V procesu vývoje pro ČSSZ budou kompetence za jednotlivé oblasti rozděleny mezi projektové týmy a Kompetenční centrum.



Jako podklad pro projektové řízení slouží seznamy pracovních položek (work items). Každá položka má povinná pole, udržuje se její historie (žádnou položku nelze smazat). Položka prochází definovanými stavy, přechod mezi jednotlivými stavy může provést pouze autorizovaná osoba.

V každém projektu je třeba udržovat minimálně seznam přidělených úloh a seznam objevených chyb. Doporučuje se též udržovat seznam požadavků (bezpečnost, výkonnost, spolehlivost, ...), scénářů (use case) a rizik v souladu s metodologií MSF Agile. (MSF v4 iterace)

Návaznost na architekturu systému

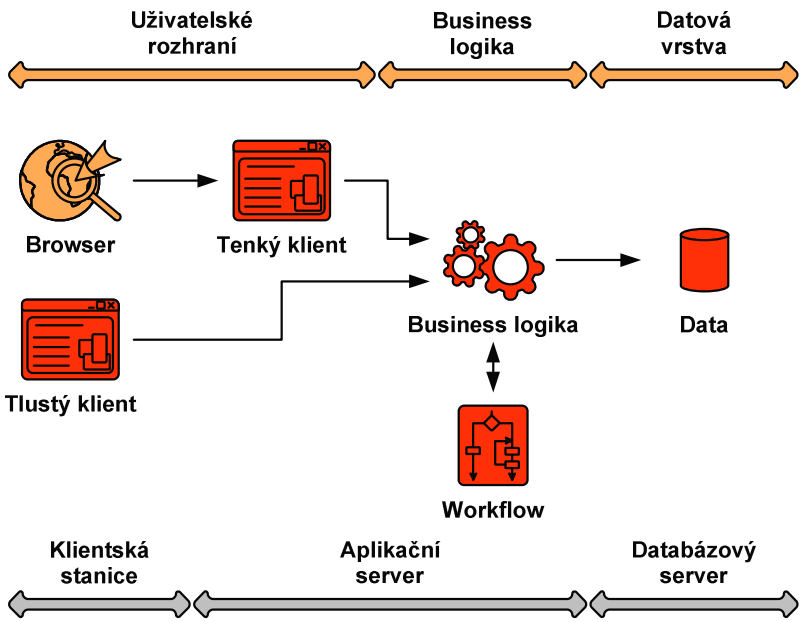
Uvedené architektonické standardy vycházejí z aktuálních koncepčních materiálů ČSSZ, zde jsou uvedeny nejdůležitější, tak aby byly respektovány při návrhu a tvorbě jednotlivých aplikací. Vychází z „Cílového konceptu“, kapitola 9

Metody a postupy

Uvedené metody a postupy jsou zaměřeny zejména na design a programování jednotlivých aplikací, tak aby byly zajištěny východiska tohoto dokumentu.

Třívrstvá architektura

Je potřeba dbát zejména na jasné oddělení jednotlivých vrstev a zejména na umístění správných částí do správných vrstev, je nutné aby aplikace byla tvořena tak, aby jednotlivé vrstvy byly od sebe odděleny a bylo možno libovolnou z vrstev, při dodržení rozhraní, nahradit novou verzí. Je využívána pro oddělení implementace jednotlivých typů logiky systému.



Databázová vrstva

- Je určena pro efektivní ukládání dat
- Data je vhodné publikovat přes datové adaptéry
- Neobsahuje business logiku

Aplikační vrstva

- Je určena zpracování busines logiky
- Je určena pro předání informací a dat presenční vrstvě

Prezenční vrstva

- Je určena pro prezentaci dat
- Neobsahuje business logiku

OOP (Object Oriented Programming)

Důsledně dodržovat paradigma objektově orientovaného přístupu, zejména:

- Nevytvářet monolitické aplikace – tvořit co nejmenší (pokud možno znovupoužitelné) komponenty – komponentový návrh a vývoj
- Využívat návrhové vzory (GoF)
- Důsledně vytvářet asociační třídy pro rozhraní mezi packagemi uvnitř systému

SOA (Service Oriented Architecture)

Dodržovat základní fundamenty SOA.

- **Zapouzdření služeb** (Service Encapsulation) - Služby skrývají svou vnitřní logiku před okolím, prezentují pouze své rozhraní.

  Dopad:

  - Není možné přistupovat k vnitřním datům služby, jinak než přes definované rozhraní.

- **Volné vazby mezi službami** (Service Loose coupling) - Služby udržují takové vztahy, které minimalizují vzájemné závislosti.

  Dopad:

  - Žádná aplikace se nesmí zhroutit na základě nedostupnosti jiné služby.

- **Kontrakt služby** (Service contract) – Definovaný dokument, který popisuje charakter a možnosti komunikace se službou. Jsou zde explicitně definovány podmínky pro komunikaci a využití služeb.

  - V případě, že jsou známy základní datové typy (string, int, guid, char) prvků struktur rozhraní webové služby (parametry či návratové struktury), je třeba kontrakt definovat typově

  - Použití netypových struktur (XmlDocument, String, Object) v definicích struktur kontraktu webové služby (interní i přístupné z jiných systémů) k předávání strukturovaných zpráv (např. XML) je možné pouze výjimečně na základě schválení Odboru rozvoje a údržby IS ČSSZ

- **Znuvupoužitelnost služby** (Service reusability) – Aplikační logika je rozdělena na menší celky tak, aby bylo dosaženo vyšší míry opětovného použití služeb.

  Dopad:

  - Služby budou vyvíjeny ne pouze pro proprietární účel, ale s cílem učinit službu využitelnou i v jiných případech.

  - Snaha o využití existujících služeb má přednost před tvorbou nových.

- **Skládání služeb** (Service composability) – Skupiny služeb je možné skládat do „vyšších“ složených služeb.

- **Service autonomy** – Služby kontrolují veškerou zapouzdřenou logiku.

- **Bezstavový charakter služeb** (Service statelessness) – Služba minimalizuje držení stavů spojených s konkrétní aktivitou.

  Dopad:

  - Služba je navrhována pasivně, nepamatuje si data specifická pro odběratele.

- **Vyhledatelnost služeb (Service discoverability) –** Služba je navržena tak, aby bylo možné ji najít (objevit) pomocí stanoveného mechanismu.

  Dopad:

  - Služba je evidována v centrálním katalogu služeb

BRA (Business Rules Approach)

Business Rules Approach – formalizování pravidel podnikových procesů tak, aby byly srozumitelné bez nutnosti programátorských znalostí. Tyto pravidla je možné měnit a tím upravovat obchodní proces aniž by bylo nutné zasahovat do samotného kódu.

Tento přístup vyžaduje návrh a vývoj tak, aby bylo dosaženo co nejvyšší flexibility. Pravidla a omezení jsou stanovena pomocí externích pravidel. Pro změnu pravidel bude existovat GUI

Transakční přístup

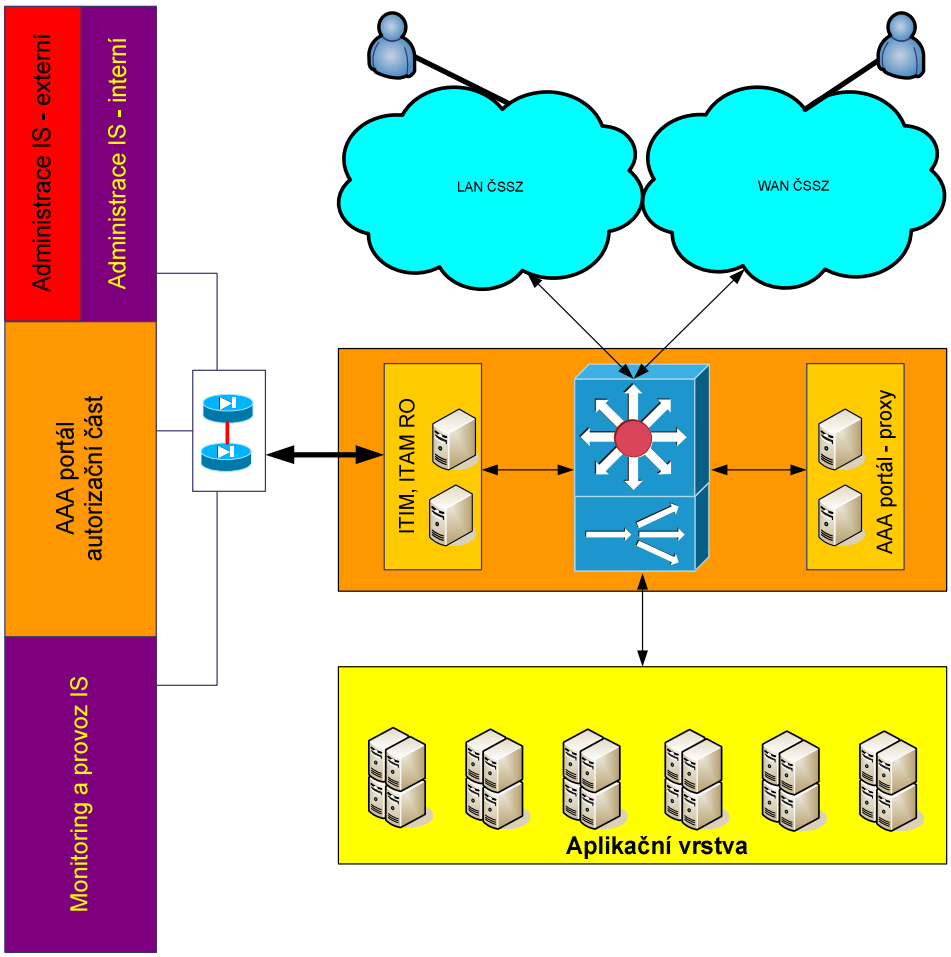
Je přístup k zajištění stability, konzistence a robustnosti systému za využití transakcí. Transakcí je označována skupina operací, které jsou realizovány s tím, že budou provedeny všechny úspěšně, nebo nebude provedena žádná. Většinou se jedná o sérii takovýchto operací. Dojde-li k selhání libovolného článku této série, systém zajistí zrušení ostatních transakcí.

Poznámka: **Nejedná se pouze o databázové transakce, ale o transakce obecně !**

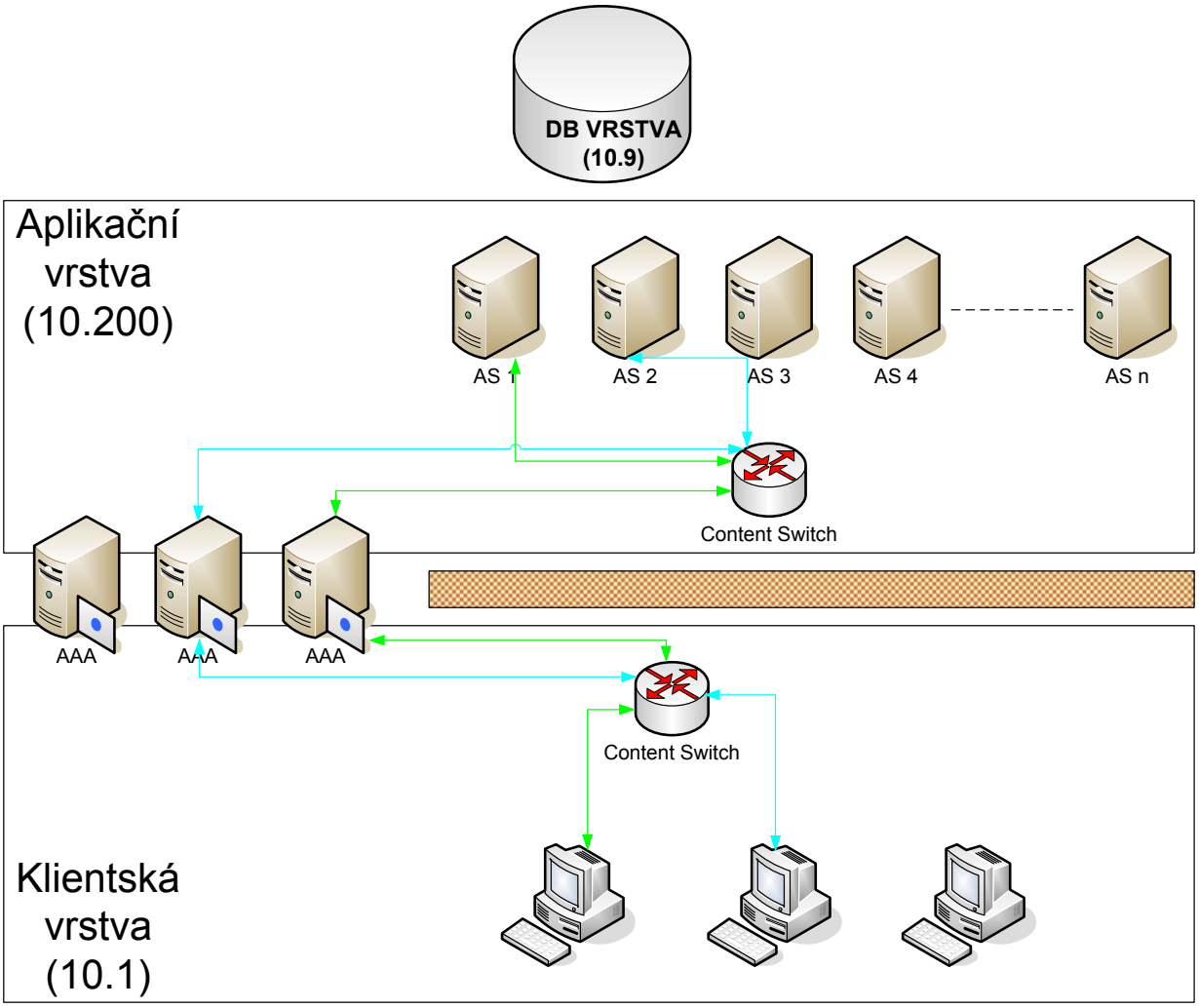
Infrastruktura a technologie

Pro podporu infrastruktury a využívané technologie jsou využity standardy ČSSZ, kopie dokumentace je umístěna v úložišti Architektury:
http://172.20.2.13/twiki/bin/view/Architektura/StandardsCSSZ

Základní schéma infrastruktury



Dopady síťových standardů do metodiky vývoje



V rámci síťových standardů lze jednodušeným pohledem rozdělit síť na fragmenty

- Klientská vrstva – obsahuje klientské stanice
- Aplikační vrstva – obsahuje aplikační servery
- Databázová vrstva – Oracle RAC

Tyto 2 sítě jsou fyzicky odděleny a jsou propojeny přes proxy AAA a je povolena komunikace pouze přes protokol HTTP TCP na portu 80 na obou vrstvách. Na úrovni aplikační vrstvy je mezi aplikačními servery komunikace povolena, včetně prostupu do databázové vrstvy.

Směrování jednotlivých požadavků je řízeno pomocí content switchů, ty zajišťují směrování včetně load balancingu (LB). V rámci LB zajišťují držení session (Sticky session) na konkrétním aplikačním serveru na základě předané source IP adresy, která je uvedena v příslušném requestu. Jelikož proxy AAA nahrazuje source IP svou vlastní, je možné tímto způsobem zajistit Sticky session pro APV, které má vyhrazený stejný nebo menší počet serverů než je aktuální počet instancí AAA portálu. Z toho plyne omezení – Případný výpadek některé z instancí AAA portálu má za následek nevyužití všech přidělených aplikačních serverů. V opačném případě je nutno řešit LB jiným způsobem.

Aktuálně jsou vybrány tyto způsoby:

- Načtení informace z http leaderu daného požadavku – položka IV-USER na content switchi

- Uložení session do DB (persistence session)

Pro výběr konkrétní varianty je potřeba vždy provést zátěžový test, který prokáže vhodnost použití konkrétního řešení. Proto bude proveden zátěžový test pro vybranou aplikaci.

Na úrovni komunikace aplikačních serverů Sticky session není zajištěna, to znamená, že jednotlivé požadavky mezi aplikačními servery jsou rozkládány na všechny aplikační servery, které jsou k dispozici. Pro různé požadavky mezi aplikačními servery v jedné session, mohou tedy být využity různé servery.

Integrace

Datová integrace

Pro podporu datové integrace je připraven Datový katalog

Cílem je sjednocení datových typů, musí být využity jak v analýze (definice popisek se nesmí odvolávat na elementární datové typy, ale na typy z DK), tak následně v implementaci. Protože datové typy jsou vlastně třídy (nedefinují jen rozsah prvku, ale i povolené hodnoty a kontroly, bavíme se tady implementaci kódu !)

Integrace aplikační vrstvy

Navazuje na „Cílový koncept“ kapitola 9.1.3 – integrace bude zajištěna využitím Metod a postupů architektury systému a kapitola 7.3.4 Design aplikací.

Správa rozhraní

Rozhraním je zde myšlena definice formátů zpráv a souborů, pomocí kterých mezi sebou jednotlivé aplikace komunikují. Organizačně se s tím zachází trochu jinak, než s datovými typy, DK zde má především evidenční funkci (za definici je zodpovědná oblast, která rozhraní zveřejňuje).

- prvky rozhraní musí být definovány pomocí datových typů z DK
- ke každému rozhraní musí existovat invertovaný seznam všech modulů a aplikací, které ho používají
- řešitelé musí vést seznam rozhraní i pro komunikace „uvnitř“ oblasti (tyto popisky nemusí být zveřejňovány do DK, ale musí být interně spravovány obdobným způsobem)

Programování a design

Pro podporu standardizace výstupů od programátorů: (PSM model, Zdrojový kód, Programátorská dokumentace, Vlastní aplikace a Instalace aplikace), jsou připraveny konvence a postupy. Ty jsou určeny pro podporu zvyšování kvality vytvářených aplikací, zejména pro snazší převzetí kódu, rychlou orientaci v kódu při vyhledávání chyb, řešení problémů, provádění změnových požadavků, sjednocení ovládání a tedy snazší zaškolení uživatelů. Dále jsou sjednocovány administrační nástroje, tedy konfigurace a umístění a struktura administrátorských logů.

Uživatelské rozhraní

Aplikace jsou vytvářeny jako desktopové (tlusté), pokud:

- Vyžadují intenzivní zadávání dat s nutností vysoké produktivity (vkládání údajů apod.)
- Používají hardware připojený k lokálnímu počítači (scanner, čtečka čárového kódu apod.)
- Musí fungovat v off-line režimu odpojeném od sítě

V ostatních případech se aplikace vytvářejí jako webové (tenké).

Uživatelské rozhraní aplikace musí být konzistentní s používaným operačním systémem (Windows XP) a používat stejné ovládací prvky a principy.

Každá aplikace musí být opatřena uživatelskou dokumentací zpracovanou formou průvodce jednotlivými úlohami (task-based, nikoliv tedy pouhý referenční popis jednotlivých položek menu).

Jazykové verze

Veškeré texty aplikace musí být v češtině, obzvláště texty zobrazované koncovým uživatelům v rámci ČSSZ / OSSZ či dokonce texty přístupné uživatelům mimo ČSSZ. V českém jazyce je třeba udržovat i texty zápisů v logu a výstupu trasování (výstupy do logů a trasovací informace je možné provádět bez diakritiky).

V anglickém jazyce je vhodné ponechat názvy proměnných, konstant, shapes, datových typů, struktur, tříd, konfiguračních parametrů, služeb, operací, souborů apod.

Knihovna grafických prvků

Pro sjednocení vzhledu aplikací a sjednocení ovládání, tedy pro podporu snazšího zaškolení uživatelů, sjednocení práce s aplikací a eliminaci chyb způsobených nesprávným ovládáním, případně pro sjednocení prezentace problémů je připravena Knihovna grafických prvků tento dokument vychází z platných dokumentů grafický manuál pro aplikace ČSSZ.

Pro podporu grafického manuálu je připraven katalog prvků a pro podporu tvorby aplikací jsou připraveny hotové prvku, s různou podporou, dle typů klientů a použitých technologií:

Pro všechny aplikace

- Společné jednotné ikony

Pro jednotlivé typy klientů

Tenký klient:

- Společné Css styly s příklady použití
- Fragmenty HTML kódu pro sjednocení použití css stylů
- Pro aplikace v .NET 2005 je připravena a distribuována masterpage s dokumentací a příklady použití

Tlustý klient

- Společné komponenty pro podporu grafického manuálu (informační pruh, status bar a chybové hlášení)

Programátorské konvence

Pro prostředí .NET (preferované prostředí) jsou připraveny závazné konvence:

- Programátorské konvence .NET 2.0

Dále jsou závazné dodržovat doporučení Microsoft obsažené v:

- MSDN – Design Guidelines for Class Library Developers

Pro prostředí Java a J2EE (používáno pouze ve výjimečných schválených případech) musí být dodržovány doporučení SUN:

- http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Kódování

Veškerý kód a výstupy, včetně komunikace jsou v UTF-8

Jmenné konvence

Pro všechna prostředí je nutné package (namespacy) pojmenovávat dle klíče:
`AMCSSZ.JmenoProduktu.Vrstva.package`

Pro prostředí .NET jsou jmenné konvence detailně popsány v Programátorské konvence .NET 2.0

Jmenné prostory XML

V produktu je třeba důsledně odlišovat interní datové struktury od externích, které jsou předávány při komunikaci s okolními systémy (zprávy). Interní datová struktura nesmí být použita pro komunikaci na rozhraní. Tím bude eliminována nutnost měnit rozhraní v důsledku změn interních datových struktur. Je tedy třeba, aby vlastními jmennými prostory byla oddělena interní (XML) schémata od XML schémat rozhraní, respektive schémat sdílených (veřejných) datových struktur.

Interní jmenné prostory

Interní jmenné prostory XML pro datové struktury, které nejsou zprávou, musí odpovídat následující struktuře:

http://schemas.cssz.cz/ProductName/[PackageName]/SchemaName/SchemaVersion

kde

[PackageName]	je volitelný identifikátor modulu / balíčku produktu
SchemaName	je jméno schématu
SchemaVersion	je identifikace verze v podobě: [majorVerze].[minorVerze].[release]

Jmenné prostory sdílených dat

Sdílené datové struktury jsou ty, které mohou být používány více systémy a vyskytují se na více rozhraních. Příkladem může být podmnožina metadat dokumentu DMS. Tyto datové struktury nemusí být pouze ve vlastnictví jednoho produktu a nemusí být nějakým konkrétním produktem definovány. Mohou vznikat jako produkt (datové) architektury na základě požadavků či potřeb

jednoho či (zpravidla) více projektů. Takovéto typy musí být uvedeny v katalogu datových typů a jejich jmenný prostor bude vytvářen podle následujícího vzoru:

http://schemas.cssz.cz/OblastKDT/SchemaName/SchemaVersion

kde

OblastKDT	je oblast definovaná katalogem datových typů
SchemaName	je jméno schématu
SchemaVersion	je identifikace verze v podobě: [majorVerze].[minorVerze].[release]

WebServices

zvolená technologie pro realizaci služeb

Vytvářené služby musí splňovat následující podmínky:

- Musí volány asynchronně
- Musí podporovat UDDI
- Musí mít návratovou hodnotu
- V případě chyby musí vracet jasný popis chyby
- Veškeré chyby a varování musí být logovány
- Musí být autorizovány
- V definici rozhraní webových služeb není bez předchozího schválení dovoleno používat netypové předávání strukturovaných informací (XML) s použitím XmlDocument, String či Object.
- V definici rozhraní webových služeb není doporučeno z důvodu problematické přenositelnosti napříč platformami používat DataSet.
- Webové služby musí být bezestavové, mezi jednotlivými voláními neudržují stav (session).
- **Webové služby musí vždy definovat strukturu požadavků request/response. Nesmí se jednat o primitivní datový typ (string, int apod.), ale musí to být třída či struktura zapouzdřující jednotlivé členy (např. Struct {string result, int code})**
- Target namespaces webových služeb musí obsahovat číslo verze – blíže viz popis jmenných prostorů rozhraní.

Namespace rozhraní

Namespace rozhraní je textový řetězec jednoznačně identifikující dané rozhraní, je uložen v atributu targetNamespace elementu wsdl:definitions. Atibut bude ve tvaru:

targetNamespace = [namespaceRozhraní]/[majorVerze].[minorVerze].[release]

kde:

[namespaceRozhraní] je string ve tvaru

namespaceRozhraní =
http://interfaces.cssz.cz/[názevProjektu]/[PackageName]/[názevRozhraní]

[PackageName] je volitelný řetězec a nemusí být vyplněn

příklad targetNamespace : = http://interfaces.cssz.cz/PPV/VZTLog/ActivationInterface/1.1.2

Dále soapAction bude ve tvaru : soapAction="[targetNamespace]/[jménoOperace].

Tímto dosáhneme stavu kdy pokud bude klient komunikovat s nesprávnou verzí vebové služby dostane odpověď HTTP 500 s textem výjimky : *SoapException: Server did not recognize the value of HTTP Header SOAPAction*.

Verze rozhraní

Verze rozhraní je řetězec ve tvaru [**majorVerze**].[**minorVerze**].[**release**] ,kde [majorVerze],[minorVerze],[release] jsou celá nezáporná čísla. Při změně definice rozhraní je doporučeno následující:

- přidá/ubere se z rozhraní metoda nebo se změní její název - zvednout major verzi
- změní/přidá se datová struktura - zvednout minor verzi
- změní se název datové položky - zvednout minor verzi
- minimální změna primitivního typu, která nebude mít příliš důsledků ve změně kódu např: int na long – zvednout release
- jakákoliv jiná změna (za změnu je považováno že se soubor wsdl změní) – zvednout jakoukoliv verzi, dle uvážení závažnosti změny

příklad verze : 1.0.2 nedoporučuji používat zaznamenání verze ve formátu 001.000.002

Autorizace, autentizace

V Postupu autorizace a autentizace se musí dodžet definice specifikována dokumentem :

Požadavky na nové aplikace při integraci do AAA portálu.

Komunikace webová služba webová služba

Tento odstavec uvádí shrnutí pro tento způsob komunikace, vše zde popsané vychází z již zmíněného dokumnetu. Pokud se provádí tento způsob komunikace a cílová služba vyžaduje autentizaci je nutné do HTTP streamu přidat hlavičku *iv-user : [jméno uživatele*]. Tím se provede zastoupení websealu, který jinak při komunikaci klient-webová služba tuto hlavičku sám doplní,ale protože se obě služby nachází v bezpečné zóně, kde není dostupný Domain Controller domény ČSSZ, komunikaci iniciující webová služba nemůže použít stejný postup autentizace jako klientská aplikace. A také ani není možné z webové služby požadovat komunikaci jdoucí přes webseal.

Stavy aplikace

Jednotlivé komponenty aplikace musí být vytvářeny jako bezestavové, případě potřeby využití technologických stavů je ve výjimečných případech možné stavy ukládat do DB.

Ošetření výjimek

- Všechny hlavní metody a procedury musí být ošetřeny pomocí bloku try (catch, end try)
- Výjimky nesmí být potlačovány, všechny musí být zobrazeny
- Popisy výjimek (návratové hodnoty) musí být v českém jazyce, srozumitelné koncovému uživateli
- Třídy pro ošetření výjimek jsou umísťovány do namespace Err
- Všechny výjimky musí být logovány viz. Logování
- Všechny chyby musí být evidovány v číselníku chyb, **samotná aplikace pracuje pouze s kódy chyb**. Zobrazované výjimky vygenerují chybový kód. K tomuto kódu aplikace dotáhne odpovídající popis chyby z číselníku chyb. Nadále je s chybou pracováno ve formátu „kódChyby – popis chyby“, např. „011122332 – popis chyby“. V tomto formátu je chyba zobrazována uživateli a logována.
- V detailních informacích ErrorFormu je možné zobrazit ostatní položky z číselníku chyb.

Logování

- Logování je nutné provádět s využitím komponent, které umožňují změnu formátu logované zprávy a cíle logování úpravou konfigurace bez zásahu do zdrojového kódu aplikace
- Komponenty pro logování musí podporovat logování do
  - systémového logu (EventLog na platformě Windows, syslog na platformě UNIX)
  - textového souboru
  - databáze MS SQL
  - databáze ORACLE

Pro platformu JAVA je povinné logování s využitím komponent Log4j.

Pro všechny programy platí, že musí logovat veškeré výjimky, které nastanou. Program obsahuje debugovací a informační log.

Všechny aplikace na základě konvencí logují standardní komponentou log4net (log4j pro J2EE). Tato komponenta zajišťuje, že formát a cílové umístění jsou konfigurovatelné v rámci aplikace, je samozřejmě možné mít v aplikaci konfiguraci více a ty mezi sebou kombinovat (například standardní logu do úrovně chyba se ukládají do standardního úložiště logů a zde archivovány a informační hlášení jsou ukládány pouze na aplikační server, kde jsou každý den přepisovány.

Upozornění: *logování znamená ukládání chybových případně dalších hlášení aplikace, logování a tedy ani tento dokument* ***neřeší*** *uchování auditovacích informací..*

Řešení v aplikacích

Všechny aplikace logují dle použité technologie:

.NET	LOG4NET
J2EE	LOG2J

S následujícími úrovněmi:

Úroveň	Popis	Použití
FATAL	Neošetřené výjimky a chyby infrastrukturního charakteru	Standardní formulář pro zachycení výjimek (komponenta ErrorForm), handler pro odchycení neošetřených výjimek, …
ERROR	Aplikační chyby	Veškeré aplikační výjimky a chyby
WARN	Varování	Veškeré varování aplikace
DEBUG	Debugovací informace	Do důležitých částí systému, pro prověření běhu programu. Do všech větví programů (if, case, ….)
TRACE	Trasovací informace	
INFO	Informační hlášení	Veškeré informační hlášky

Pro oblast komunikace pomocí webových služeb je pro úrovně DEBUG/TRACE požadováno zaznamenávání událostí:

- SEND – informace o odeslání zprávy (klient: HTTP POST, HTTP GET, SMTP send, SOAP request apod., server: HTTP response, SOAP response)
- RECEIVE – informace o přijetí zprávy (klient: HTTP response, SOAP response, server: HTTP receive, SOAP receive)

Příslušná komponenta (Log4NET, Log4J) bude nakonfigurována vrstvy na které bude aplikace provozována:

Aplikační servery

Protože aplikační servery nejsou žádným způsobem zálohovány budou veškeré aplikační logy ukládány do databáze a event logu (nakonfigurováno pro úroveň FATAL a ERROR současně pro oba typy zápisů).

Každý aplikační modul ve svém schématu vytvoří databázovou tabulku log se strukturou:

Sloupec	Datový typ	Povinnost	Popis
DateTime	Timestamp	Y	Datum a čas zápisu do logu
ApplicationName	Varchar2(10)	Y	Zkratka aplikace dle číselníku zkratek projektů
ApplicationVersion	Varchar2(20)	Y	Verze aplikace dle Pravidel release managementu

LogThread	Varchar2(255)	N	Jméno vlákna
LogLevel	Varchar2(255)	Y	Úroveň logu (FATAL / ERROR / WARN / DEBUG / TRACE / INFO)
Logger	Varchar2(255)	N	
LogMessage	Varchar2(2000)	Y	Popis chyby ke které došlo
LogException	Clob	N	Detailní výpis chyby
ApplicationDataID	number	N	Identifikátory zpracovávané sady dat
ErrorNumber	Varchar2(9)	Y	Číslo chyby dle katalogu chyb
ExtSystemName	Varchar2(10)	Y^{WS}	Identifikace (název) externího systému, který je partnerem při komunikaci pomocí webové služby. Externí z pohledu logujícího systému – při logování poskytovatele služby je o konzumenta (klienta), při logování klienta jde o poskytovatele služby (server)
ServiceURL	Varchar2(255)	Y^{WS}	Identifikace umístění služby (poskytované respektive konzumované)
MethodName	Varchar2(100)	Y^{WS}	Název aktivované metody

Vysvětlivky:

Y^{WS} – položka závazná pro logování komunikace pomocí webových služeb

Poznámka: Pokud komunikace s externím systémem pomocí webových služeb odpovídá modelu request – response, musí být proveden záznam o požadavku i záznam o odpovědi, a to každý samostatně. Odpověď musí být, pokud je to možné, zaznamenána i v případě chyby (timeout apod.).

V konfiguraci aplikace bude vytvořen Appender, který bude logy zapisovat do této tabulky, dále bude vytvořen appender, který bude zapisovat úroveň Fatal a Error pro Windows do EventLogu a pro Linux do lokálního syslogu.

Kódy událostí nemusí tvořit posloupnosti, musí však být v rámci aplikace unikátní

Logy budou stahovány pomocí Tivoli do úložiště Tivoli . **Upozornění je nutné zajistit archivaci příslušných dat** !

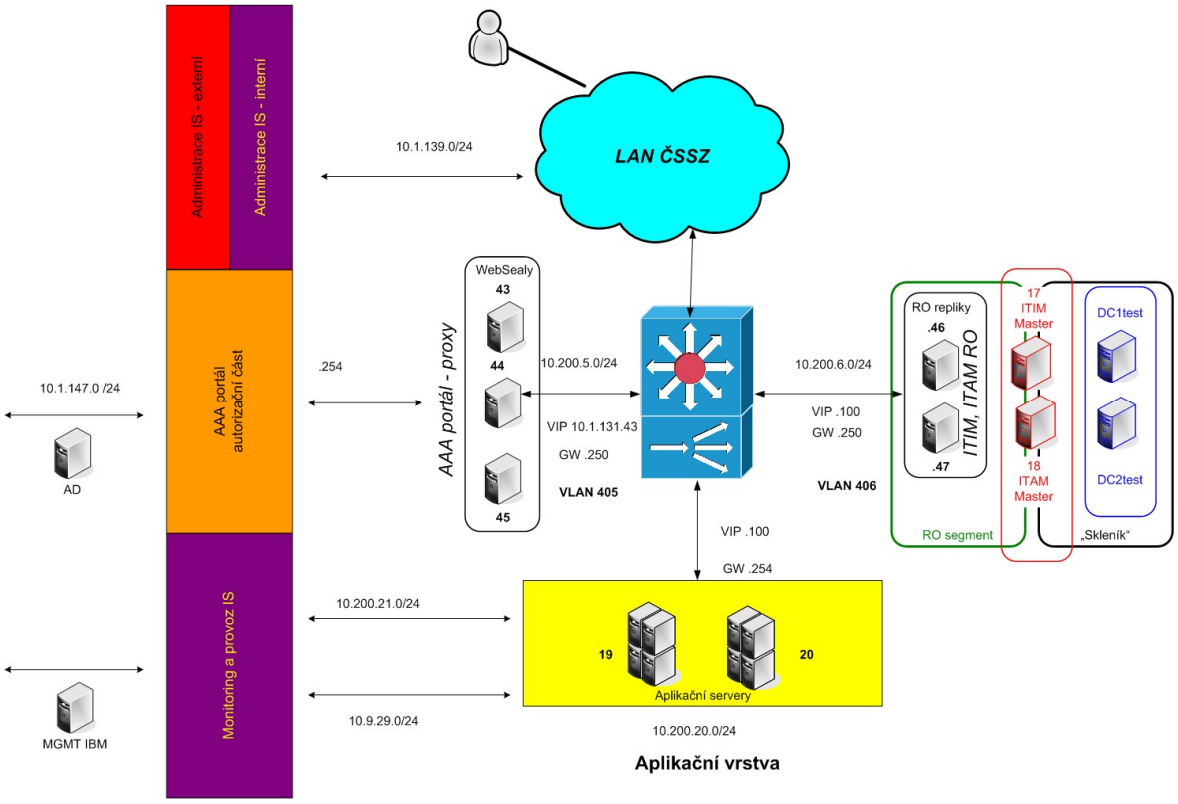
Windows klienti

Logy z aplikací provozovaných jako Windows klient, se ukládají do EventLogu Windows pomocí appenderu – ukládá se do úrovně Error stejné informace jako pro AS

Bezpečnost

Pro zajištění autorizace a autentizace požadavků musí být vždy využit AAA portál. Konkrétně je popsáno v dokumentu „Integrace aplikací do AAA portálu“.

Zařazení AAA portálu a aplikací do síťové infrastruktury:



Kryptografické algoritmy

Pokud jsou v aplikaci či systému použity kryptografické algoritmy (šifrování (enkrypce), dešifrování (dekrypce), hash, tvorba a nebo ověřování digitálního podpisu atd.), musí být použité algoritmy vybrány schválením Odborem rozvoje a údržby IS ČSSZ a dokumentovány v samostatné příloze dokumentace dodávaného řešení.

Níže uvedené doporučené algoritmy jsou řazeny sestupně dle pořadí, v jakém by měly být zvažovány.

Doporučené algoritmy šifrování symetrickým klíčem:

- AES (Rijndael)
- 3DES (TDES, tripple DES)

Doporučené algoritmy šifrování asymetrickým klíčem:

- RSA (PKCS)
- Diffie-Hellman
- DSS/DSA

Doporučené algoritmy výpočtu „message digest" (hash)

- SHA-512(384)

- SHA-256(224)
- RIPEMD-160
- SHA-1, MD5

Komprimační algoritmy

Pokud jsou v aplikaci či systému použity komprimační algoritmy (komprimace/komprese, dekomprimace), musí být použité algoritmy vybrány schválením Odborem rozvoje a údržby IS ČSSZ a dokumentovány v samostatné příloze dokumentace dodávaného řešení.

Níže uvedené doporučené algoritmy jsou řazeny sestupně dle pořadí, v jakém by měly být zvažovány.

- RAR
- GZIP (LZ77)
- TAR
- ZIP

Konfigurace

Následující údaje by měly být ukládány v konfiguračních úložištích, tak, aby bylo možné je změnit bez nutnosti zásahu do kódu aplikace:

- Přístupnové údaje
    - autorizační údaje
    - connection strings apod.
- URI
    - Adresy webových služeb
    - Adresy HTTP endpointů
    - Adresáře pro vstupně-výstupní souborové operace
        - Načítané soubory (číselníky, šablony apod.)
        - Dočasné soubory
        - Data vytvářených či zpracovávaných dávek

Senzitivní údaje by měly být ukládány zabezpečeně (šifrování, hash přístupových údajů ukládaných v databázích či konfiguračních souborech) či s možností řízení přístupu (access control listy konfiguračních souborů).

Pro konfiguraci URL musí být používána DNS jména, nikoliv jména serverů či IP adresy.

Konfigurace adresářů a cest k souborům musí umožňovat zadání relativní cesty (..\Data\), absolutní cesty (X:\TEMP\) i síťové cesty (\\server\share).

Všechny prvky konfigurace musí být dokumentovány v předávací dokumentaci.

Programátorská dokumentace

Pro všechny používané technologie je programátorská dokumentace udržována v kódu a z něj je generována předávaná dokumentace.

Pro aplikace .NET a ASP.NET je způsob dokumentování popsán v Programátorské konvence .NET 2.0 a popis generování dokumentace v Programátorská dokumentace

Použití XML (Extensible Markup Language)

- XML je preferovaný formát pro komunikaci a uchováváni dokumentů.
- Pro jednání nad podobou rozhraní je nutné využívat definice pomocí XSD.
  - V XSD musí být definován výchozí jmenný prostor (default namespace)
  - Elementy musí být kvalifikovány (jmenným prostorem)
- Vhodné pro budoucí tvorbu WSDL.

Bude doplněno

Tvorba instalací a pravidla releasu managementu

Je jednotná pro všechny typy klientů a použité technologie a je popsána v Pravidla release managementu

Práce s číselníky

Veškeré číselníky se kterými aplikace pracuje musí být uloženy v databázi, aplikační vrstva s nimi pracuje přímo, klientská vrstva si je natahuje přes rozhraní webových služeb. Aplikace s číselníky mohou pracovat dle potřeby, to znamená buď si je dotahují ve chvíli kdy je potřebují, případně si je mohou nahrát před začátkem práce a pracovat s tímto snímkem.

Postup odběru číselníku

Zaslání požadavku k odběru katalogu na PM KCC (nyní na adresu karel.kriz.ext@siemens.com, monika.razova@cssz.cz) s kontaktním údaji: jméno a příjmení, email, případně UNC a seznam názvů číselníků z katalogu číselníku, které jsou cílem odběru.

Po přijetí a registraci odběratele je automaticky distribuován katalog číselníků v okamžiku aktualizace vybraného číselníku (uvedeného v seznamu k odběru)

Postup při zřízení číselníku

Zaslání navrhovaného číselníku ve formě obsahu číselníku, včetně názvů položek a pojmenování číselníku, případně použitého zdroje pro obsah číselníku.

Je možné v tomto návrhu zaslat seznam odběratelů a správců navrhovaného číselníku.

Návrh obsahu, struktury a pojmenování projde technickou kontrolou a kontrolou shody s již evidovanými číselníky. V případě jakýchkoliv neshod bude provedena modifikace návrhu, jež bude poslán zpět ke schválení zadavateli. V okamžiku shody zadavatele a správce KCC je číselník zahrnut do katalogu číselníků a katalog je ihned distribuován všem uvedeným odběratelům nově evidovaného číselníku.

Postup při aktualizaci

Zasláním návrhu změny v konkrétním číselníku na PM KCC (nyní na adresu karel.kriz.ext@siemens.com, monika.razova@cssz.cz) zahajuje zadavatel změnové řízení nad konkrétním číselníkem (zadavatelem může být kdokoliv; osoba/automat/vyhláška/norma/...atd.) O navrhované změně jsou informování všichni správci příslušného číselníku (forma: eMail) a od všech správců je požadován souhlas s navrhovanou změnou. V okamžiku souhlasu od všech správců, je fyzicky provedena aktualizace číselníku a změnové řízení je ukončeno. Všem správcům a odběratelům příslušného číselníku je rozeslán kompletní katalog číselníků.

V případě nesouhlasu alespoň jednoho správce s navrhovanou změnou, je změnové řízení ukončeno a zadavatel je o zamítnutí požadavku informován.

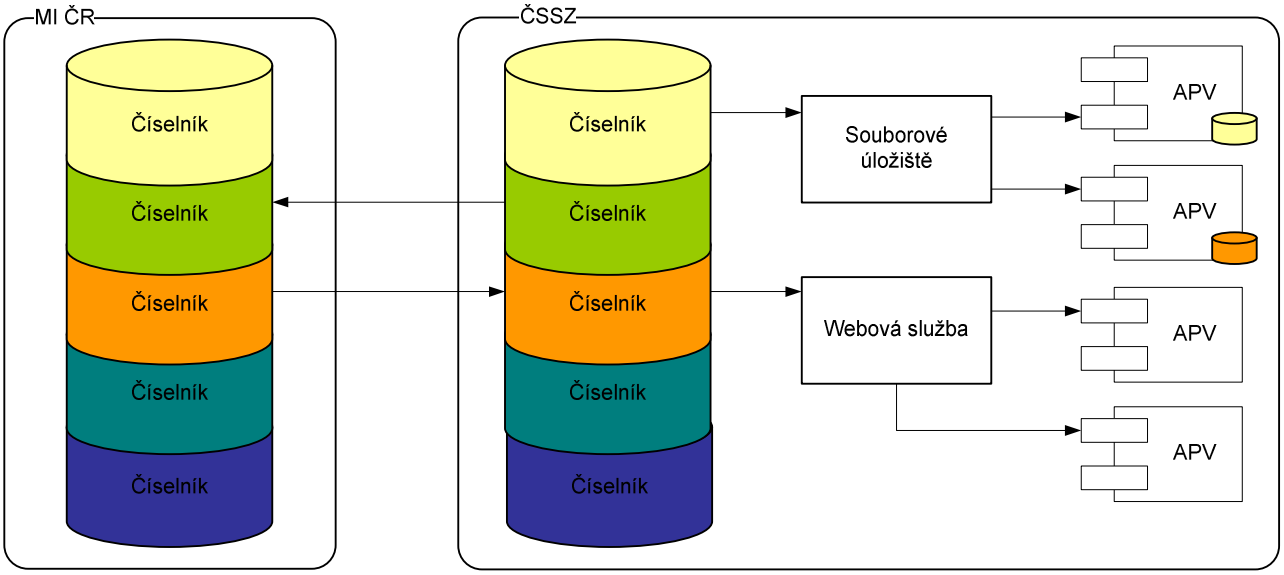
V takovém případě navrhovatel změny může navrhnout zřízení nového číselníku, jež navazuje na již existující číselník, ale bude obsahovat požadované modifikace (viz. Postup při zřízení číselníku).

Technický popis

- číselníky se dělí na interní a externí
- APV používají pouze interní číselníky
- externí číselníky slouží za zdroj pro primární číselníky
- aktualizace interních číselníků z externích číselníků se provádí po revizi obsahu a po kladném potvrzení změnového řízení správci příslušného číselníku

- interní číselníky mohou být aktualizovány z externího číselníku nebo správcem číselníku
- APV může a nemusí mít uloženou kopii interního číselníku ve svém datovém úložišti
- způsob a čas aktualizace kopie interního číselníku v datovém úložišti APV z interního číselníku KCC si spravuje samo APV dle vlastním metodických postupů pro příslušné APV
- každý interní číselník obsahuje minimálně jednoho správce číselníku
- každý správce číselníku je současně i odběratelem číselníku
- ke každému číselníku se může přihlásit libovolný počet odběratelů
- distribuce číselníku je prováděna v souboru ve formátu CSV a XML
- forma distribuce číselníku je možná technologií UNC, FTP nebo POP3
- rozsahem distribuce bude vždy kompletní katalog číselníků obsahující všechny aktuálně platné interní číselníky včetně složených interních číselníků ve výchozím formátu (pokud není u odběratele specifikován jiný formát výstupu)
- interní aktuálně platný číselník obsahuje vždy dvě položky (dva sloupce) [KOD] a [NAZEV], které jsou jedinečné v rámci aktuálně platného číselníku, jehož jsou součástí
- každý interní číselník obsahuje další systémové položky (ve výchozím výstupu skryté) [RID], [OID], [PARENTID], [VALIDFROM] , [VALIDTO], které jsou plněny aplikací KCC a uživatel číselníku je nemůže přímo upravovat

- systémová položka RID je jedinečná identifikace záznamu napříč všemi číselníky (tj. neexistují dva číselníky, jež by obsahovali shodné číslo RID)
- systémová položka OID je jedinečná identifikace kolekce záznamů, jež jsou v časové ose jedním záznamem (vazba aktuálního záznamu s historii změn nad tímto záznamem), přičemž současně je OID jedinečné napříč všemi číselníky (tj. neexistují dva číselníky, jež by obsahovali shodné číslo OID)
- k internímu číselníku mohou být přidány další položky (sloupce), potom tedy z interního číselníku vzniká složeny interní číselník při zachování původního "základního číselníku"
- název interního číselníku musí být složen ze znaků BASE 36 (Regulární vyraz: [A-Z0-9])
- názvy položek interního číselníku musí být složeny ze znaků BASE 36 (Regulární vyraz: [A-Z0-9])
- obsah v položce KOD musí být složen ze znaků BASE 36; doporučeno je však použití znaků BASE 32 (Regulární vyraz: [A-FHJ-NP-Z0-9])
- obsah v položce NAZEV musí být složen alfanumerických znaků bez řídících znaků (Regulární vyraz: [A-Ža-ž0-9])
- číselníkem je dvourozměrné pole obsahující jedinečný klíč v jedné položce (položka KOD) a jedinečný popis v položce druhé (položka NAZEV)
- každý interní číselník eviduje metadata o zdroji, času poslední aktualizace, správce, odběratele
- každý záznam v číselníku eviduje metadata o času platnosti záznamu (položka: VALIDFROM, VALIDTO), vazbu nadřízeného záznamu (viz.hierarchie) (položka: PARENTOID), ID záznamu (položka: RID) i ID objektu záznamu (položka: OID)
- metadata nejsou součástí výchozí distribuce katalogu číselníku
- každý číselník musí splňovat podmínky atestace ČSSZ podle zákona č. 365/2000 Sb., ve znění jeho novely č. 81/2006
- každý číselník státní organizace je povinen být zveřejňován v databázi MIČR v časovém období maximálně 3 dnů po aktualizaci



Sdílené komponenty

Pro tvorbu sdílených komponent jsou připraveny konvence a seznamy jednotlivých komponent v dokumentu Znovupoužitelné sdílené komponenty

Test driven developement

Veškerá nevizuální funkčnost aplikace musí být automaticky testovatelná na správnou funkčnost. Dodavatel je povinen vytvářet tyto testy (unit testy) jako součást své standardní práce na projektu.

Unit testy jsou vytvářeny na základě analýzy a návrhu jednotlivých rozhraní a tříd, a to před anebo v průběhu implementace (nikoliv tedy až po vlastním vývoji).

Počet a procento úspěšnosti unit testů jsou základním měřítkem postupu práce na projektu.

Pokrytí kódu unit testy (Code Coverage) je základním měřítkem úplnosti napsaných testů. V každém projektu by měl být stanoven cíl na pokrytí kódu unit testy blízký 100% (100% však není reálné).

Klíčové funkce aplikace musí mít definované výkonnostní cíle (např. počet definovaných transakcí za hodinu při zachování stanovené doby odezvy v 90% případů). Výkonnostní testování na splnění těchto cílů musí probíhat co nejdříve.

Klíčové uživatelské scénáře musí mít vytvořeny automatické (robotizované) testovací skripty, kterými lze aplikaci automaticky testovat.

Veškeré testy jsou součástí projektu a musí být odevzdány v takové formě, aby umožňovaly automatické znovuspuštění kdykoliv v budoucnu.

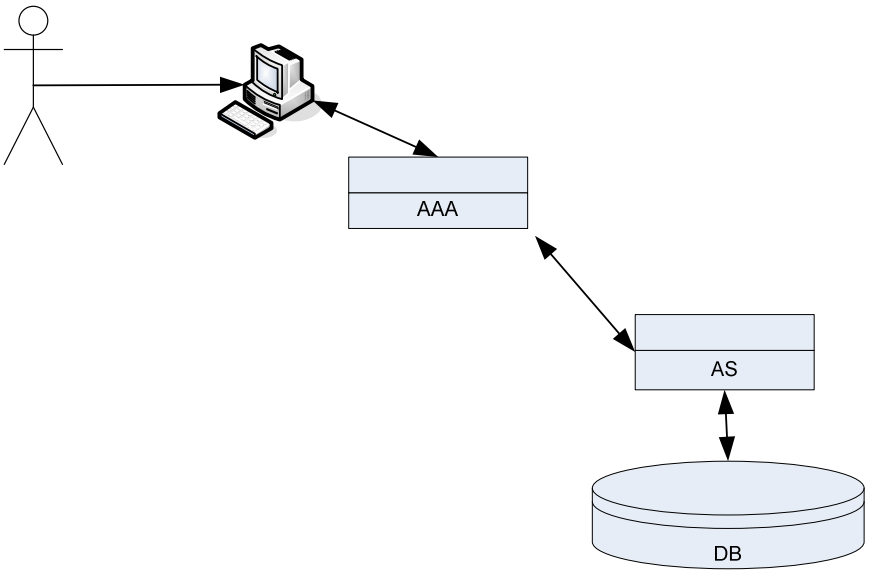
Pravidla pro práci v síti WAN

Kapitola popisuje základní požadavky na chování aplikací v síti WAN, tak aby byla zajištěna co nejefektivnější komunikace aplikací jak uvnitř aplikace, tak v rámci komunikace aplikací mezi sebou a okolními systémy.

Aplikace jsou tvořeny na třívrsvé architektuře s jasně oddělenými vrstvami, tyto vrstvy jsou využívány pouze pro účely dané vrstvy (DB – ukládání dat, aplikační vrstva – zpracování business logiky, prezenční vrstva pouze prezentace dat). Pro návrh jednotlivých aplikací je využívána SOA (service-oriented architekture) s využitím OOP.

Autorizace a autentifikace

Pro zajištění autentifikace uživatelů a autorizace požadavků pro přístup k částím aplikace a k datům musí být využíván AAA portál



Chování aplikací v síti dle typu klienta

Veškeré aplikace jsou vytvářeny na třívrstvé architektuře s oddělenými vrsvami. Pro prezentaci jsou využívány dva druhy klientů:

1. **WEB klient** – aplikace spouštěná ve WEB browseru, vlastní aplikace (prezenční logika) je umístěna na aplikačním serveru – generuje HTML kód a na klientech jsou renderovány jednotlivé obrazovky.
2. **Windows klient** – aplikace jsou nainstalovány na jednotlivých PC – business logika je však opět umístěna na aplikačním serveru – ten generuje pouze data, která předává jednotlivým klientům a ti je prezentují v již předpřipravených obrazovkách.

Webový klient

- WEB stránka obsahuje pouze dynamické informace, tzn. Pouze data a čisté HTML tagy. Nesmí obsahovat statické informace (css formátování, javascripty), tyto musí být vytaženy do samostatných souborů, tak aby je bylo možno ukládat do lokální cache.
- Komunikuje pouze s webovými servery na aplikačních serverech prostřednictvím rozhraní AAA portálu (Websealů) na portu 80
- Veškeré stránky musí být validní dle standardu W3C HTML (kontrola standardním validátorem (http://validator.w3.org/), případně pluginem (HTML Validator - https://addons.mozilla.org/firefox/249/)
- Aplikace musí být provozovatelé v prohlížeči dle „Standardu PC“
- Dle standardu PC – způsob přechodu bude popsán standardním změnovým požadavkem
- Hlavička musí obsahovat příslušné direktivy zajišťující opakované stahování pouze měnících se částí (info no cache, no expire, …)
- Jedna stránka je složena maximálně ze čtyř dotazů
- U interaktivních stránek (všechny mimo sestav) může mít jeden dotaz maximálně 20 kbytů, stránka pak 80 kbytů (dynamicky měnící se část)

Windows klient

- Komunikuje pouze s WEBovými službami na aplikačních serverech prostřednictvím rozhraní AAA portálu (Websealů) na portu 80
- Stahuje si pouze data, která uživatel potřebuje k práci s danou obrazovkou

Pravidla pro ověření živosti apliakcí

Pro zajištění monitoringu, zda jsou všechny aplikace „na živu“ je potřeba zajistit jednoduše ověřitelné řešení, které umožní oznámit stav aplikace. Současné požadavky jsou pouze na zjištění tohoto stavu, nikoli na zjištění okolí aplikace (například zda jsou na živu všechny webové služby, které ke svému běhu potřebuje nebo zda je naživu DB)

Aplikace

Tato kapitola popisuje co musí každá aplikace zajišťovat z pohledu zjišťování zda jsou aplikace na živu.

Každá aplikace bude obsahovat jednu Webovou službu **wsApplicationStatus** s metodou **getApplicationStatus**, tato metoda vrátí v textové návratové hodnotě „aplikace – OK“, jméno aplikace bude nahrazeno zkratkou aplikace dle seznamu projektů. Dále bude služba obsahovat metody pro podporu releasu managementu, např. metodu **getApplicationVersion**, která v textové návratové hodnotě vrátí verzi projektu.

Seznam všech metod webové služby:

Jméno metody	Návratová h.	Popis
getApplicationStatus	string	Zjištění stavu aplikace
getApplicationVersion	string	Zjištění verze aplikace na aplikačním serveru
getApplicationDatabaseVersion	string	Zjištění verze aplikace v databázi
getTestEnviroment	bool	Zjištění zda jsme v testovacím prostředí

Monitoring

Tato kapitola popisuje způsob jakým budou uvedené informace zjišťovány z pohledu monitorovacích nástrojů.

Monitorovací systém bude nakonfigurován tak, že v určeném časovém intervalu zavolá metodu getApplicationStatus z webové služby wsApplicationStatus . Webová služba odpoví standardním způsobem pomocí HTTP protokolu. V případě v HTTP response bude kód 200 webová služba běží bez problémů v ostatních případech Monitorovací systém zaloguje chybu, v chybě bude uložen kód chyby z HTTP protokolu a kompletní http response.

Definice konkrétních webových služeb je součástí administrátorské dokumentace každé aplikace.

Seznam kódů chyb HTTP protokolu dle standardu W3C:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Příklad volání metody getApplicationStatus:

http://localhost:4222/wsApplicationStatus/wsApplicationStatus.asmx/getApplicationStatus

Přílohy

Seznam dokumentů a jejich umístění

Název	Verze
Knihovna grafických prvků	1.8 6.5 1.0.1
Programátorské konvence .NET	1.0.13
Pravidla releasu managementu	
Konvence pro Sdílené komponenty	1.0.0
Integrace aplikací do AAA portálu Prováděcí pokyn vrchního ředitele úseku informačních a komunikačních technologií č. 7/2005 o pravidlech, zásadách a způsobu správy počítačových programů v ČSSZ Prováděcí pokyn vrchního ředitele úseku informačních a komunikačních technologií č. 22/2006 o vývoji, provozu a údržbě jednotné aplikační softwarové podpory v ČSSZ Datový katalog	1