**26.9.2011**

Uslijed testiranja inteligentnog prometnog sustava razvijenog unutar tvrtke, ustanovljeno je da u implementaciji komponente za ostvarivanje komunikacije između aplikacija sustava postoji greška kod rada bez priključka na mrežu te mi je postavljen zadatak pronaći i implementirati rješenje koje omogućava normalan rad aplikacija kada su odspojene sa mreže, čuvanje neposlanih podataka i automatsko slanje podataka kod ponovnog spajanja aplikacija.

Modul se zove komunikacijski servis (CommService), implementiran je u java programskom jeziku i za ostvarivanje komunikacije koristi redove poruka i tzv. publish-subscribe princip razmjene poruka. Programska podrška koja je korištena u implementaciji je ActiveMQ i WebsphereMQ poslužitelji redova poruka. Kod izrade je korišten Spring Integration okvir.

Prvi dan sam pristupio analizi implementacije komponente i došao do zaključka da prebacivanje na alternativni kanal komunikacije uslijed nemogućnosti spajanja na navedene poslužitelje traje jako dugo.

**27.9.2011**

Kako je komponenta implementirana korištenjem Spring Integration okvira, odlučio sam detaljnije proučiti implementaciju te dokumentaciju navedenog okvira. Okvir implementira uzorke za poslovnu integraciju (enterprise integration patterns) koja predstavlja skup klasa koje vrše razmjenu, usmjeravanje i obradu nad apstraktnim tokovima poruka.

Kod slanja poruka koriste se sekundarni kanali za slanje poruka gdje se u nemogućnosti dostave poruka navedenim poslužiteljima poruke zapisuju u datoteku. Komponenta koja odlučuje o preusmjeravanju poruke uslijed greške tek nakon dugo vremena zapisuje poruku u datoteku za razmjenu što je rezultat kasne prijave greške od strane klase okvira koja dostavlja poruku navedenim poslužiteljima.

Pokušao sam smanjiti vrijeme odustajanja od slanja i prijave greške no bez zadovoljavajućih rezultata.

**28.9.2011**

Današnji dan sam proveo u analizi klijentskih biblioteka za navedene poslužitelje i analizi konfiguracije samih poslužitelja. Zaključio sam kako u samoj konfiguraciji nije ništa sporno. Detaljnijim proučavanjem ActiveMQ poslužitelja sam ustanovio da ga je moguće konfigurirati na način da funkcionira kao lokalni ugrađeni poslužitelj koji je istovremeno most prema drugim poslužiteljima redova poruka.

Predložio sam mentoru i ostalim članovima tima u raspravi da se problem riješi na način da se u samu aplikaciju ugradi lokalni poslužitelj koji će se konfigurirati na gore navedeni način te da je za očekivati da gašenje bilo koje aplikacije, nestanak mreže ili centralnih poslužitelja neće usporiti rad aplikacija, te da će kod ponovnog uspostavljanja aplikacije sve neposlane poruke biti poslane u onom poretku u kojem su bile stavljene u red. Zaključeno je kako je takvo očekivano ponašanje poželjno te kako će uvesti veću toleranciju na prekide u komunikaciji i osigurati nesmetani i neusporeni rad aplikacija uslijed nestanka komunikacije.

Pretpostavka koja je nužna za ovakvu funkcionalnost je da ActiveMQ ugrađeni poslužitelj može sam ponovno uspostaviti komunikaciju i nastaviti dostavljati poruke.

Dobio sam zadatak napraviti primjer u kojemu će se ta pretpostavka pokazati ispravnom.

**29.9.2011**

Danas sam pristupio ugrađivanju ActiveMQ poslužitelja u apstraktne primjere aplikacija. Dodao sam java arhive u projekte te konfigurirao poslužitelje da se spajaju na centralne poslužitelje i implementirao slanje primjera poruka.

Zatim sam iskopčavao prežu te palio i gasio poslužitelje. Ustanovio sam da se proizvođači poruka uspiju ponovno spojiti ali ne i potrošači te sam ostatak dana proveo u istraživanju kako riješiti navedeni problem.

**30.9.2011**

Danas sam ustanovio da je problem bio u klasi koja implementira tvornicu konekcija koja u privremenom spremištu čuva već uspostavljene konekcije koje su u odspojenom stanju te kod svakog ponovnog stanja, sama konekcija se ne uspostavlja.

Problem sam riješio na način da sam isključio privremeno spremanje postojećih „mrtvih“ konekcija te uspješno testirao rješenje i dokazao gore spomenutu pretpostavku. Dalje sam pristupio rekonfiguriranju postojećih aplikacija i ugrađivanju lokalnih poslužitelja redova.

**3.10.2011**

Danas sam testirao napravljene izmjene i izrađivao zapisnik testa. Sljedeći zadatak mi je bio izmijeniti generator konfiguracijskih datoteka koji je implementiran u Excel datoteci i koristi VBA makro funkcije za generiranje XML datoteka koje predstavljaju konfiguraciju komunikacijskog servisa. U generator konfiguracije je bilo potrebno ugraditi funkcionalnost koja omogućava generiranje konfiguracije mosta prema centralnim poslužiteljima redova poruka. Za to je bilo potrebno implementirati dodatne funkcije i modificirati postojeće makro funkcije.

Nakon testiranja i provjere generiranih konfiguracijskih datoteka, uključio sam ih u aplikacije umjesto ručno napisanih i testirao sustav koji se pokazao da radi ispravno.

Ove izmjene su pokazale potrebu za drugim izmjenama, konkretno način generiranja datoteka za razmjenu prema zahtjevu klijenta.

**4.10.2011**

Na zahtjev klijenta, svaka aplikacija sustava mora moći funkcionirati autonomno određeni period. Uslijed autonomnog funkcioniranja aplikacije mora biti moguće ručno prenijeti poruke na nekom od vanjskih medija (USB memorija). Do sada je sustav funkcionirao da su se poruke uslijed prekida komunikacije spremale na disk. Sada sustav funkcionira na način da se poruke čuvaju na lokalnom redu poruka uslijed prekida komunikacije a šalju se automatsku nakon oporavka konekcije.

Kako bi se ostvarila funkcionalnost ručnog prijenosa poruka u datotekama vanjskim medijem, bilo je potrebno implementirati funkcionalnost u komunikacijskom servisu za pražnjenje reda poruka i njihovo zapisivanje u datoteke.

To je ostvareno implementacijom posebnog servisa koji je implementiran unutar svežnja komunikacijskog servisa a na poziv export metode na servisu za izvoz se prazni red poruka do određenog trenutka u vremenu i te poruke se zapisuju u datoteke na vanjski medij.

Slijedilo je testiranje navedene funkcionalnosti koje je prošlo sa očekivanim rezultatima.

**5.10.2011**

Na zahtjev klijenta je bilo potrebno implementirati način prevođenja poruka za razmjenu redom poruka i datotekama koji nije u ljudima nečitljivom obliku.

Do sada su se poruke prevodile u XML zapis prije slanja ili zapisivanja u datoteku što je rezultiralo nepotrebno velikim prometom u odnosu na korisne podatke u poruci, kao i relativno razumljivim formatom čovjeku.

Klijent je na taj format zapisa imao primjedbu pa sam dobio zadatak implementirati prevođenje poruka u binarni oblik prije slanja i spremanja u datoteke.

Pristupio sam proučavanju postojećih biblioteka za pretvaranje Java objekata u nizove bajtova te sam se odlučio testirati dva načina pretvorbe tj. serijalizacije, Hessian i Google ProtoBuffers.

Izradom male testne aplikacije i mjerenjem vremena ova dva načina serijalizacije u usporedbi sa XML serijalizacijom pomoću XStream datoteke došao sam do sljedećih rezultata :

XStream : 2,6 minuta

Hessian : 34 sekunde

Google ProtoBuffers : 2,7 sekundi

Iz ovih rezultata sam odličio uz odobrenje mentora implementirati prevođenje poruka korištenjem Google ProtoBuffers bibliotebe.

To sam napravio na način da sam u klasi za prevođenje zamijenio pozive biblioteke XStrema na par mjesta sa pozivima ProtoBuffers biblioteke.

Dodatno je bilo potrebno u sve objekte koji se prevode za slanje implementirati prazni konstruktor bez argumenata, pošto bez takvog konstruktora nije moguće instancirati klasu pomoću refleksije.

Slijedilo je testiranje rješenja.

**6.10.2011**

Danas sam nastavio sa testiranjem prevođenja poruka kod slanja i zapisivanja u datoteke, te kod primitka datoteka i čitanja datoteka.

Slijedilo je ugrađivanje rješenja u postojeće aplikacije i testiranje aplikacija.

Nakon testiranja sam napravio zapisnik, te pripremao i dopunjavao testne procedure za integracijsko testiranje svih aplikacija nakon ugradnje izmjena od prošla dva tjedna.

**7.10.2011**

Danas sam prema testnim procedurama radio ispitivanje ugrađenih izmjena u aplikacije.

Prilikom testiranja ustanovio sam grešku koja je rezultat krivog načina instanciranja klasa pomoću refleksije u OSGi okolini. Nakon kraćeg uvida u dokumentaciju okoline sam izmijenio način instanciranja klasa na način da se umjesto klasičnog java dobavljača klasa koristi OSGi dobavljač klasa. Do problema je također došlo zato što je apstraktna klasa koja se nasljeđuje uvezena iz drugog svežnja, a taj svežanj ne uvozi paket klase koja se de-serijalizira.

Nakon ispravaka sam ponovio testove koji su sada završili uspješno te nastavio sa testiranjem. Nakon testiranja sam popunio zapisnik i uskladio stabilnu granu na repozitoriju izvornog kôda sa učinjenim izmjenama.