UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Simon Gotlib

Opis in uporaba strežnika Microsoft Team Foundation Server v projektnem delu

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Matjaž Kukar

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00498/2010
Datum: 15.03.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SIMON GOTLIB**

Naslov: **OPIS IN UPORABA STREŽNIKA MICROSOFT TEAM FOUNDATION SERVER V PROJEKTNEM DELU**

USING MICROSOFT TEAM FOUNDATION SERVER FOR PROJECT MANAGEMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Microsoft Team Foundation Server je skupek orodij, procesov in smernic, s katerimi omogoča boljšo komunikacijo med zaposlenimi in preglednost projektov ter na enem mestu zagotavlja vsa orodja za načrtovanje, razvoj in preskušanje programske kode. Kandidat naj opiše glavne funkcionalnosti in značilnosti orodij in prikaže njegovo uporabo na realnem primeru. Orodja naj primerja tudi z njihovimi odprtokodnimi ekvivalenti (npr. SVN) in opiše njihove morebitne prednosti ali slabosti.

Mentor:
doc. dr. Matjaž Kukar

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a **Simon Gotlib**,

z vpisno številko **63040210**,

sem avtor/-ica diplomskega dela z naslovom:

Opis in uporaba strežnika Microsoft Team Foundation Server v projektnem delu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc.dr. Matjaž Kukar

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju doc. dr. Matjažu Kukarju za podporo in nasvete pri izdelavi diplomske naloge ter Tomažu Mlakarju, nadrejenemu v podjetju, ki mi je pomagal pri odločitvi o vsebini diplomske naloge ter omogočil uporabo orodja TFS za namen izdelave diplomske naloge.

Posebna zahvala pa je namenjena staršem za vso podporo in finančno pomoč pri študiju.

Seznam uporabljenih kratic

TFS	Team Foundation Server
API	Application Programming Interface
ASP	Active Server Pages
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SP1	Service Pack 1
RUP	Rational Unified Process
MSF	Microsoft Solutions Framework
CMMI	Capability Maturity Model Integration
IDE	Integrated Development Environment
QoS	Quality of Service
CPU	Central Processing Unit
XML	Extensible Markup Language
OLAP	Online Analytical Processing
BIDS	Business Intelligence Designer Studio
VSIP	The Visual Studio Industry Partner
TFB	Team Foundation Build
VSTS	Visual Studio Team System

Kazalo vsebine

Povzetek

Diplomsko delo vsebuje predstavitev strežnika Team Foundation Server. Microsoft Team Foundation Server je skupek orodij, procesov in smernic, ki omogočajo boljšo komunikacijo med zaposlenimi in preglednost projektov, ter na enem mestu zagotavlja vsa orodja za načrtovanje, razvoj in preskušanje programske kode. Med ključne faze strežnika Team Foundation Server štejemo nadzor nad viri, sledenje delovnim nalogam, upravljanje projektov, upravljanje z graditvijo kode in poročanje. Te faze so opisane v teoretičnem delu ter prikazane na praktičnem primeru celotnega procesa razvijanja aplikacije *Pregledovalnik dogodkov*, ki iz komponente Event Viewer glede na izbrano možnost (področje dogodkov) prebere vse podatke in jih prikaže uporabniku. Za implementacijo aplikacije je bilo uporabljeno orodje Microsoft Visual Studio 2008, za programiranje programske kode pa jezik C#. V diplomskem delu je predstavljena tudi primerjava strežnika Team Foundation Server s konkurenčnim odprtokodnim sistemom Subversion. Opisane so prednosti in slabosti obeh sistemov, primerjava skupnih lastnosti ter lastno mnenje o strežniku Team Foundation Server in sistemu Subversion.

Pri raziskovanju strežnika Team Foundation Server se je izkazalo, da je strežnik v večini primerov najboljša izbira za razvoj aplikacij. Team Foundation Server namreč vsebuje vse potrebno za preprost in kvaliteten razvoj aplikacij.

Ključne besede: Team Foundation Server, Subversion, aplikacija, upravljanje projektov, nadzor nad viri, poročanje, delovna naloga

Abstract

The BA thesis includes a presentation of Team Foundation Server. Microsoft Team Foundation Server is a set of tools, processes and guidelines that enable a better communication between the employees and better project overview. It provides the tools needed for planning, developing and testing the program code in one place. Key phases of the Team Foundation Server are source control, work item tracking, project management, code building management and reporting. These phases are described in the theoretical part and presented using a practical example of the application building process from the beginning to the end. Based on the selected option (event area), the application *Pregledovalnik dogodkov* reads all the information from the Event Viewer component and displays it to the user. Microsoft Visual Studio 2008 was used for the implementation of the application, and C# for programming the code. The empirical part of the thesis also includes the comparison of Team Foundation Server and Subversion which is an open source system. Advantages and disadvantages of both systems, the comparison of the properties common to both systems as well as my own opinion on Team Foundation Server and Subversion are also presented.

The study of Team Foundation Server proved this server is the best choice for application development. Team Foundation Server has all that is needed for a simple, high-quality application development.

Keywords: Team Foundation Server, Subversion, application, project management, source control, reporting, work item

1. Uvod

Danes se večina podjetij ukvarja s težavo, kako izboljšati preglednost razvoja programske opreme, kakovost programske opreme in komunikacijo v podjetju, pri tem pa povečevati in deliti svoje znanje. To skušajo rešiti z uporabno različnih delnih rešitev, ki onemogočajo učinkovito integracijo dela in povečujejo stroške razvoja. Uporabljajo različne delne rešitve za različne naloge. Sem sodijo rešitve za vodenje različic, graditev kode ali vodenje delovnih nalog. Za vsako področje posebej uporabljajo ločeno rešitev, kar pomeni veliko dodatnega upravljanja in vodenja sistemov.

Namen diplomske naloge je predstaviti rešitev, ki združi vse te možnosti. Team Foundation Server je strežnik, ki vključuje orodja, procese in smernice, s katerimi omogoča boljšo komunikacijo med zaposlenimi in preglednost projektov ter na enem mestu zagotavlja vsa orodja za načrtovanje, razvoj in preizkušanje programske kode, med katere spadajo orodja za nadzor nad viri, sledenje delovnim nalogam, upravljanje projektov, upravljanje z graditvijo kode in poročanje. Ta orodja so namenjena predvsem zelo velikim skupinam. Strežnik Team Foundation Server sem pobližje spoznal v podjetju, kjer sem zaposlen. Na podlagi tega sem dobil tudi motivacijo za izdelavo diplomskega dela o strežniku.

Prvi del predstavlja podroben opis strežnika, ki je sestavljen iz nekaj ključnih faz, med katere štejemo upravljanje različic, sledenje delovnim nalogam, upravljanje projektov, upravljanje z graditvijo kode in poročanje.
V fazi upravljanja z različicami je predstavljeno upravljanje z izvorno kodo ter vodenje različic. Faza sledenja delovnim nalogam opisuje upravljanje delovnih nalog, kar omogoča učinkovitejšo komunikacijo med uporabniki, kot so arhitekti, razvijalci in preizkuševalci. Faza upravljanja projektov je namenjena upravljanju procesa razvoja programske opreme. V fazi upravljanja z graditvijo kode je opisano spajanje različnih delov projekta, prevajanje, razporejanje in preskušanje. Zadnja faza predstavlja poročanje in opisuje poročila, ki jih lahko uporabimo za analizo napredka postopka, ustreznosti projekta in učinkovitosti razvojne skupine in skupine za preskušanje.

V drugem delu je predstavljen podoben odprtokodni sistem Subversion ter njegova primerjava s strežnikom Team Foundation Server.

Zadnji del predstavlja praktičen primer uporabe strežnika pri razvijanju aplikacije Pregledovalnik dogodkov, ki iz komponente Event Viewer glede na izbrano opcijo (področje dogodkov) prebere vse podatke in jih prikaže. Prikazane so vse glavne lastnosti strežnika Team Foundation Server od začetka do konca nastajanja aplikacije Pregledovalnik dogodkov.

Glavni cilj diplomske naloge je preučiti Team Foundation Server, predstaviti njegove najpomembnejše lastnosti in prikazati njegovo uporabo na dejanskem primeru. Tako bi lahko uporabnikom pomagal pri odločitvi o izbiri rešitve, ki bi zagotovila dosledno kakovost programske opreme pri razvojnih projektih in s tem povečala zadovoljstvo strank.

2. O strežniku Team Foundation Server

Družba Microsoft je dolgo delala na področju ustvarjanja izpopolnjenega programa. Velike skupine neprestano ustvarjajo in vzdržujejo zapletene programske kode za večkratno izdajo. Za uspeh v proizvodnji programov morajo razviti učinkovit pristop za kontroliranje različic, napak, spremljanje delovnih nalog in upravljanje graditve (izgradnje).

S pomočjo ogrodja Microsoft Solutions Framework so združili bistvo vseh teh tehnik v skupek fleksibilnih elementov projektnega upravljanja.

Izkoristili so rezultate dolgoletnih izkušenj in raziskovanj pri ustvarjanju programa in metodologije za razvoj novih tehnologij in tehnik, ki so usmerjene proti optimizaciji procesa skupinskega razvoja programa. Rezultat tega dela je Microsoft Team Foundation Server.

Obstajata dve strani strežnika Team Foundation Server. Na eni strani je to zbirka funkcionalnosti, ki je v skupni rabi z različnimi člani projektne skupine, da bi jim omogočila še uspešnejše skupinsko delo (člani skupine lahko preprosto dajo v skupno rabo projektne načrte, proizvode dela in ocene napredka), na drugi strani pa je platforma TFS narejena za integriranje in razširitev. Odjemalci in partnerji lahko prilagodijo elemente TFS-ja in jih dopolnijo z novimi elementi. Razširitve se lahko raztezajo od najbolj enostavnih do zelo zapletenih, od zamenjave imen določenih polj delovnih nalog do integracije popolnoma novih orodij. [1]

Strežnik Team Foundation Server je ena izmed komponent sistema Microsoft Visual Studio Team System. Visual Studio Team System je produktivna, integrirana in razširljiva zbirka orodij, ki dopolnjuje družino izdelkov orodja Visual Studio in omogoča boljšo komunikacijo in sodelovanje med skupinami za razvoj programske opreme. S sistemom Visual Studio Team System lahko organizacije zagotovijo večjo predvidljivost in kakovost na začetku razvojnega procesa in pogosto tudi skozi celoten proces. Vsebuje tudi ogrodje Microsoft Solutions Framework, ki ponuja niz preizkušenih procesov za razvoj programske opreme, s katerimi organizacije lažje dostavijo rešitve za uporabo v podjetjih. [14]
Visual Studio Team System sestavljajo:

- **Visual Studio Team Suite.** Zbirka programov Visual Studio Team Architect Edition, Visual Studio Team Developer Edition in Visual Studio Team Test Edition.
- **Visual Studio Team Edition for Software Architects.** Vizualni oblikovalci, ki omogočajo arhitektom, upraviteljem operacij in razvijalcem, da oblikujejo storitvene rešitve, ki jih je mogoče preveriti v njihovem okolju delovanja.
- **Visual Studio Team Edition for Developers/Database Professionals.** Napredna razvojna orodja, ki omogočajo skupinam, da izdelajo zanesljive rešitve in aplikacije.
- **Visual Studio Team Edition for Software Tester.** Napredna orodja za preskušanje obremenitve, ki omogočajo skupinam, da preverijo učinkovitost delovanja aplikacij pred uvedbo.
- **Visual Studio Team Foundation Server (TFS).** [15]

Strežnik Team Foundation Server omogoča tudi odlično združljivost z ostalimi orodji. Eden izmed teh je Microsoft Project, ki ga na primer lahko uporabimo za delo z delovnimi nalogami strežnika Team Foundation Server. Microsoft Project omogoča nazoren pregled nad projekti. Z diagrami postanejo zasedenost delavcev in njihove naloge jasne v trenutku. Bistveno se skrajša čas načrtovanja, izvajanja in nadzora projektov. Za učinkovit nadzor nad potekom dogodkov je dovolj le pogled, tudi v primeru naknadnih sprememb. Z večanjem

obsega in števila projektov sicer narašča možnost za neporazdeljene obremenitve virov, vendar Microsoft Project omogoča njihov optimalen izkoristek ali skrajšanje časa izvedbe projekta.

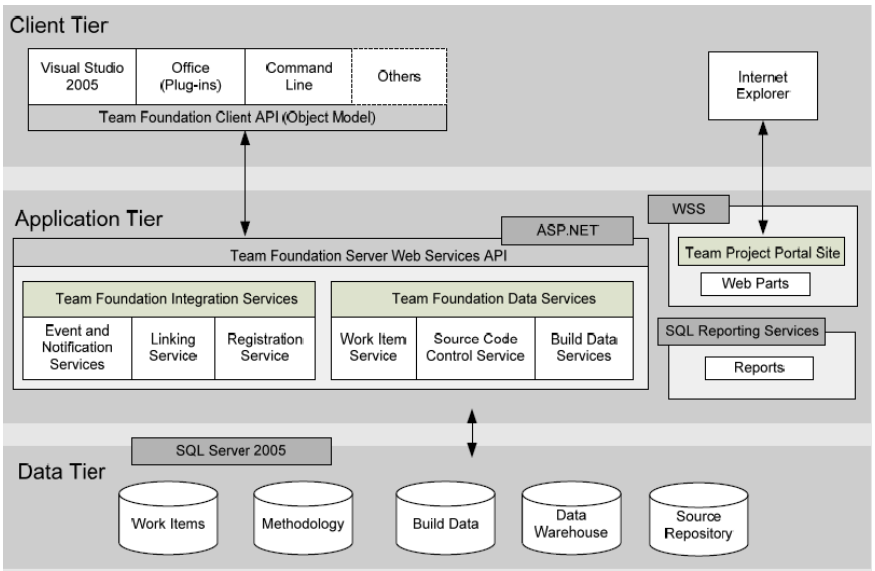
2.1 Funkcionalnosti

Team Foundation Server ima pet glavnih funkcionalnosti:

- **Nadzor nad viri** za upravljanje izvorne kode in ostalih proizvodov, ki zahtevajo različice.
- **Sledenje delovnim nalogam** za spremljanje poti različnih delovnih nalog, kot so napake, zahteve, opravila in scenariji.
- **Funkcije projektnega upravljanja** omogočajo oblikovanje skupinskega projekta, ki je načrtovan na koristnih in specifičnih programskih procesih, ter omogočajo načrtovanje in spremljanje z uporabo programov Microsoft Excel in Microsoft Project.
- **Skupinska graditev** zagotavlja funkcionalnost laboratorija, ki ga sestavljamo s skupinsko graditvijo. S skupinsko graditvijo lahko skrbniki sinhronizirajo vire, združujejo različne datoteke v eno, analizirajo kodo, gradijo izdajo, objavijo poročila prevajanja kode itd.
- **Zbiranje podatkov in poročanje** služi kot pomoč pri ocenjevanju stanja projektne skupine, ki temelji na podatkih, zbranih iz orodij strežnika Team Foundation Server. [1]

2.2 Arhitektura

TFS uporablja logično tri-nivojsko arhitekturo, ki vključuje odjemalski, aplikativni in podatkovni nivo. Odjemalci strežnika TFS vzajemno delujejo z aplikativnim nivojem prek različnih spletnih storitev; aplikativni nivo podpirajo različne podatkovne zbirke na podatkovnem nivoju. Slika 1 prikazuje sestavne dele posameznega nivoja TFS in njihovo vzajemno delovanje.



Slika 1: Arhitektura strežnika Team Foundation Server

a) Odjemalski nivo

Odjemalski nivo vsebuje naslednje pomembne sestavne dele:

- **Predmetni model strežnika Team Foundation Server.** To je javni API, ki se uporablja za komunikacijo s strežnikom TFS. Objektni model lahko uporabimo za ustvarjanje lastne odjemalske aplikacije za komunikacijo s strežnikom TFS.
- **Sestavni deli Visual Studio Industry Partners (VSIP).** To so orodja tretjih oseb, dodatki in jeziki, ki se uporabljajo znotraj integriranega razvojnega okolja Microsoft Visual Studio.
- **Integracija Microsoft Office.** Sestavljena je iz več dodatkov za Microsoft Office in Microsoft Office Project, ki omogočajo pisanje poizvedb in posodabljanje delovnih nalog v zbirki podatkov za sledenje delovnim nalogam. To je predvsem uporabno za projektne vodje, ki že obsežno uporabljajo ta orodja.
- **Orodja orodne vrstice.** To so orodja, ki omogočajo komunikacijo s strežnikom TFS prek orodne vrstice. Večina teh orodij ponuja funkcije za nadzor nad viri in je uporabna za avtomatizacijo ponavljajočih se opravil in časovne nastavitve opravil.
- **Ogrodje pravilnika sprostitve.** Podpira funkcijo pravilnika sprostitve, ki je razširljiv mehanizem, ki omogoča preverjanje kode med procesom sprostitve.

b) Aplikativni nivo

Aplikativni nivo razkriva naslednje spletne storitve ASP.NET, ki so dostopne z odjemalskega nivoja. Spletne storitve so razvrščene v naslednje zbirke:

- *Podatkovne storitve strežnika Team Foundation Server*
- *Integrirane storitve strežnika Team Foundation Server*

Podatkovne storitve strežnika Team Foundation Server

Primarna naloga teh spletnih storitev je upravljanje s podatki na podatkovnem nivoju. Te storitve so:

- **Spletne storitve nadzora nad viri.** Odjemalski nivo uporablja te spletne storitve za izvršitev različnih funkcij nadzora nad viri in za komunikacijo z zbirko podatkov nadzora nad viri.
- **Spletne storitve za sledenje delovnim nalogam.** Odjemalski nivo uporablja te spletne storitve za ustvarjanje, posodabljanje in pisanje poizvedb za delovne naloge v zbirki podatkov za sledenje delovnim nalogam.
- **Spletne storitve graditve Team Foundation.** Odjemalski nivo in ogrodje MSBuild uporabljata te spletne storitve za izvršitev procesa graditve.

Integrirane storitve strežnika Team Foundation Server

Skupek teh spletnih storitev ponuja dve funkciji, in sicer integracijo in avtomatizacijo. Te storitve ne komunicirajo s podatkovnim nivojem. Integrirane storitve strežnika Team Foundation vključujejo:

- **Spletno storitev za registracijo.** Storitev se uporablja za registracijo številnih drugih storitev TSF. Informacije ohranja v podatkovni zbirki za registracije.

Storitve uporabljajo te informacije, da odkrijejo in določijo, kako komunicirati med sabo.

- **Spletno storitev za varnost.** Storitev sestoji iz storitve za varnost skupine in storitve za odobritev. Storitev za varnost skupine se uporablja za upravljanje vseh uporabnikov in skupin TFS. Storitev za odobritev ponuja sistem za kontrolo dostopa.
- **Spletno storitev za povezovanje.** Storitev omogoča orodja za vzpostavljanje razmerij (povezav) med podatkovnimi elementi. TFS s povezavo na primer vzdržuje razmerje med napako delovne naloge in napako kode, ki je bila spremenjena, da popravi napako.
- **Spletno storitev za dogodek.** Ta spletna storitev omogoča orodje ali storitev za registracijo vrst dogodkov. Uporabniki se lahko naročijo na te dogodke in prejemajo opozorila po e-pošti ali s pozivom spletne storitve. Dogodek sprostitve lahko uporabimo na primer za sprožitev nepretrgane integrirane graditve.
- **Spletno storitev za razporejanje.** Storitev deluje skupaj s spletno storitvijo za povezovanje in tako omogoča razporejanje izdelkov TFS glede na sistematiko. To omogoča poročanje tudi za artifakte, ki si ne delijo skupnih sistematik za urejanje podatkov. Če so na primer delovne naloge običajno razporejene po skupini, testi pa po komponenti, lahko tudi teste razporedimo po skupini, tako da so zabeleženi poleg delovnih nalog.

c) Podatkovni nivo

TFS ne podpira neposrednega dostopa do podatkov, ki so shranjeni na podatkovnem nivoju, iz odjemalskih aplikacij. Vse zahteve za podatke se morajo ustvariti prek spletnih storitev na aplikativnem nivoju. Podatkovni nivo TFS je sestavljen iz naslednjih podatkovnih shramb, ki ustrezajo podatkovnim storitvam na aplikativnem nivoju:

- **Sledenje delovni nalogi**. Shrani vse podatke, povezane z delovnimi nalogami.
- **Nadzor nad viri.** Shrani vse podatke, povezane z nadzorom nad viri.
- **Graditev Team Foundation.** Shrani vse podatke, povezane s funkcijo skupinske graditve TFS.
- **Skladišče poročil.** Shrani vse podatke, povezane z vsemi orodji in funkcijami TFS. [2]

3. Nadzor nad viri

Sistem nadzora nad viri je popolnoma nov Microsoftov sistem in ne gre za izboljšano različico prejšnjega sistema nadzora nad viri Visual Source Safe. Sistem nadzora nad viri je bil zasnovan kot poslovni sistem z možnostjo obdelave več sto ali celo več tisoč uporabnikov hkrati. [3]

3.1.1 Naloge

Naloge sistema nadzora nad viri so:

- hraniti datoteke na varen in zanesljiv način;
- ponuditi način za združevanje nizov različic datotek v izdajo;
- omogočiti delo z isto datoteko več uporabnikom hkrati – s koncepti sprostitve, rezervacije in združevanja:
- hraniti informacije o spremembah datoteke, in sicer:
  - kdo jih je naredil
  - kdaj jih je naredil
  - zakaj jih je naredil. [3]

3.1.2 Cilji

Sistem za nadzor nad viri je bil zasnovan z nekaj temeljnimi cilji:

- zagotoviti prilagodljive rešitve za razvojne skupine podjetij;
- zagotoviti zanesljive rešitve;
- zagotoviti oddaljen dostop do sistema z uporabo spletnih protokolov HTTP/HTTPS;
- omogočiti delo z izvorno datoteko več razvijalcem hkrati;
- zagotoviti popolnoma integrirano uporabniško izkušnjo v razvojnem okolju Microsoft Visual Studio. [3]

3.1.3 Arhitektura

Ker je sistem nadzora nad viri le še ena storitev, ki jo ponuja TFS, deluje na isti trislojni arhitekturi. Osnovan je na strežniku Microsoft SQL Server kot podatkovni zbirki za hranjenje skladišča nadzora nad viri, izpostavlja dostop do skladišča prek niza storitev, ki jih gosti aplikacijski strežnik TFS, in uporablja razvojno okolje Microsoft Visual Studio kot odjemalca. [3]

Slika 2: Arhitektura sistema nadzora nad viri

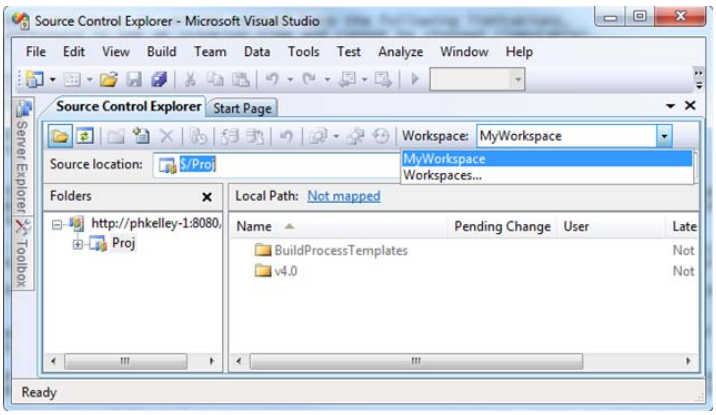
3.1.4 Varnostne pravice in dovoljenja

Nadzor nad viri uporablja isti sistem Windows za integrirano varnost kot aplikativni sloj strežnika TFS. To pomeni, da je uporabljen isti model uporabnika/skupine za določanje ravni dovoljenj, značilnih za operacije nadzora nad viri, in da se uporablja isti proces za dodajanje in odstranjevanje uporabnikov. Z drugimi besedami, nadzor nad viri ne hrani lastne specifične uporabniške podatkovne zbirke ali varnostnega sistema, ampak sodeluje v večji infrastrukturi strežnika TFS. Uporabnik ima v nadzoru nad viri vlogo uporabnika ali skrbnika.
Uporabniško skupino običajno sestavljajo razvijalci, preizkuševalci, sponzorji ali drugi. Ti posamezniki bodo opravljali osnovne skupne naloge, in sicer rezervacijo datotek za spreminjanje, sprostitev spremenjene datoteke, ogled datotek v skladišču in dodajanje ali brisanje datotek v skladišču.
Skrbniki so bolj osredotočeni na upravljanje z nadzorom nad viri kot celoto. Upravljajo dostop do skladišča in ohranjajo celovitost in varnost predmetov v skladišču. Naloga skrbnikov je tudi določanje, kdaj se mora ustvariti nova veja projekta v drevesu, in odpravljanje konfliktov pri spajanju projektov iz različnih vej. Članstvo v tej skupini je običajno omejeno na skrbnike TFS in skupinske vloge, npr. upravitelje projektov in projektne vodje.
Na ravni dovoljenj sistem nadzora nad viri podpira različne pravice, ki jih je mogoče dodeliti/odstraniti posameznemu uporabniku ali celotni skupini. [3]

3.1.5 Delovni prostor

Delovni prostor je področje v lokalnem datotečnem sistemu. Vse lokalne kopije datotek v sistemu nadzora nad viri so shranjene v delovnem prostoru. Lokalne kopije datotek se ustvarijo ob prvi povezavi s skladiščem. Takrat lokalne datoteke postanejo niz delovnih datotek. Ko spremenimo eno izmed lokalnih datotek, se spremembe, ki jih naredimo, označijo v delovnem prostoru in niso posredovane strežniku, dokler jih ne sprostimo. Z datotekami v delovnem prostoru lahko torej delamo, kar želimo, ne da bi to vplivalo na ostale člane v skupini in ne da bi poškodovali izvajanje projekta.

Delovni prostor se lahko sklicuje na več projektov. Lahko uporabimo tudi več delovnih prostorov za izolacijo datotek ali različic za lastno uporabo. Število delovnih prostorov se razlikuje glede na računalnik oz. uporabniški račun. Za vsak računalnik, ki ga uporabljamo, imamo lahko različne definicije delovnih prostorov. Kot edini uporabnik imamo lahko na enem računalniku več delovnih prostorov. [3]



Slika 3: Delovni prostor

3.1.6 Dodajanje datotek v sistem nadzora nad viri

Vsak obstoječi projekt razvojnega okolja Microsoft Visual Studio moramo dodati pod nadzor. Če tega ne storimo, so datoteke samo v lokalnem računalniku, ne pa na strežniku. V tem primeru sistem nadzora nad viri ne ve nič o projektu oz. datotekah in zato ni mogoče uporabljati lastnosti, ki jih omogoča sistem nadzora nad viri. Datoteke moramo dodati na strežnik. [3]

3.2 Upravljanje z datotekami v sistemu nadzora nad viri

Do sedaj smo pregledali osnove ustvarjanja delovnega prostora odjemalca, prikazali nalaganje datotek iz strežnika z uporabo sistema nadzora nad viri ter dodajanje datotek na strežnik. Sedaj pa se bomo posvetili najpomembnejši stvari v sistemu nadzora nad viri – upravljanju s spremembami. [3]

3.2.1 Nalaganje datotek iz skladišča virov

Obstajata dva načina za prenos datoteke iz skladišča virov strežnika in shranjevanje v lokalni delovni prostor: z ukazom *Get Latest (Pridobi najnovejšo različico)* ali ukazom *Check-out (Rezerviraj)*. Ukaz *Get Latest* naloži najnovejšo različico datoteke, ki obstaja na strežniku, in jo kopira v delovni prostor, ukaz *Check-in* pa sporoči strežniku, da ne želimo samo najnovejše različice datoteke, ampak da jo želimo tudi urejati. [3]

3.2.2 Rezervacija datotek

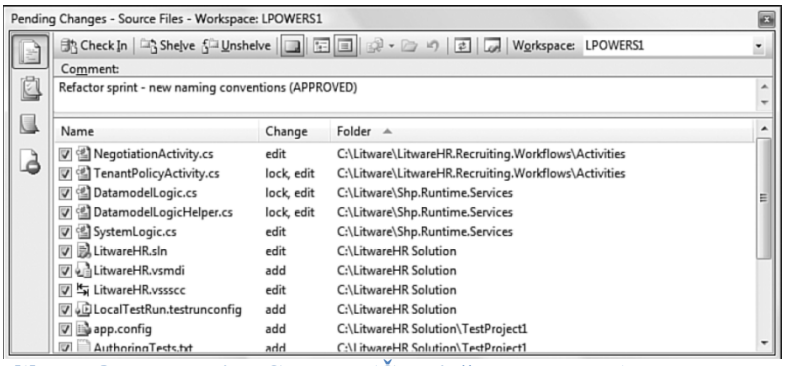
Rezervacija elementa kopira datoteko iz strežnika v lokalni računalnik in odstrani atribut »samo za branje«. Dodatno lahko uporabimo tudi ukaz *Check out for edit (Rezerviraj za urejanje)*, ki rezervira samo lokalno različico. Če ni nastavljena lastnost, da ima več uporabnikov lahko rezervirano isto datoteko, se ta datoteka zaklene in jo lahko drugi uporabniki samo berejo. Ko je datoteka rezervirana, jo lahko spreminjamo.

Poznamo dve vrsti rezervacij:

- **Izključna rezervacija.** Datoteko lahko spreminjate samo vi.
- **Rezervacija za skupno rabo**. Datoteko lahko spreminja več ljudi naenkrat. [3]

3.2.3 Sprostitev datotek

Ko končamo s spreminjanjem, je čas, da te spremembe sprostimo nazaj na strežnik. To lahko naredimo na tri načine: v oknu *Solution Explorer, Source Control Explorer* ali *Pending Changes (Čakajoče spremembe)*. [3]



Slika 4: Okno Pending Changes (Čakajoče spremembe)

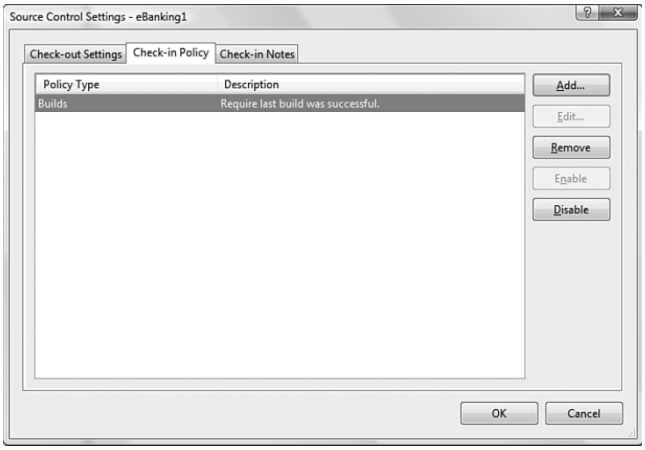
3.2.3.1 Razumevanje pravilnika sprostitve

Projektne skupine imajo različa pravila, ki jim morajo uporabniki slediti pri ugotavljanju, ali je sprostitev primerna. Sprostitev datotek, ki se ne prevedejo, verjetno ni dobra ideja. Ko bi nekdo drug izvedel postopek *Get Latest (Pridobi najnovejšo različico)* ali *Check-out (Rezerviraj),* bi bil projekt uničen zaradi teh sprememb. Sistem nadzora nad viri prepozna pomen preverjanja sprostitve in ponuja način uveljavljanja določenih pravil o sprostitvi z uporabo pravilnika sprostitve.

Na voljo so trije pravilniki sprostitve:

- **pravilnik analiziranja kode:** zagotavlja, da so bila izvedena določena preskušanja kode, preden je bila dovoljena sprostitev.
- **pravilnik preskušanja:** zagotavlja, da so bila izvedena določena preskušanja (izbrana s seznama vseh poznanih preskusov), preden je bila dovoljena sprostitev.
- **pravilnik delovnih nalog:** zahteva, da je s sprostitvijo povezana ena ali več delovnih nalog.

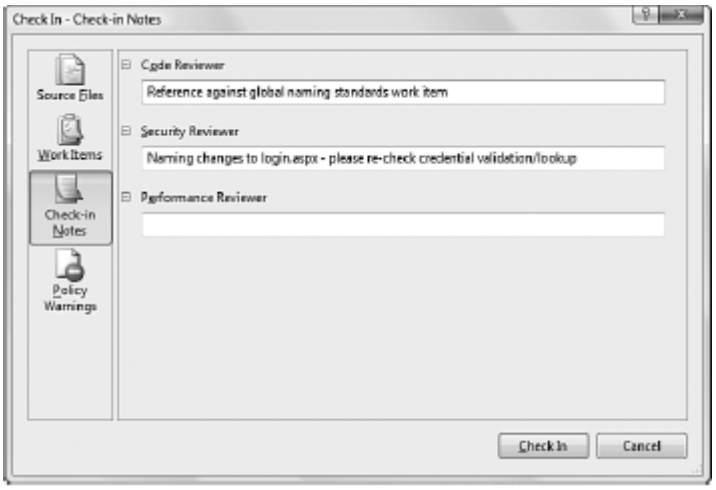
Pravilnike sprostitve običajno nastavijo skrbniki, pogosto pa privzet niz pravilnikov vključi predloga procesa, ki se trenutno uporablja. [3]



Slika 5: Zavihek Check-in Policy (Pravilnik sprostitve)

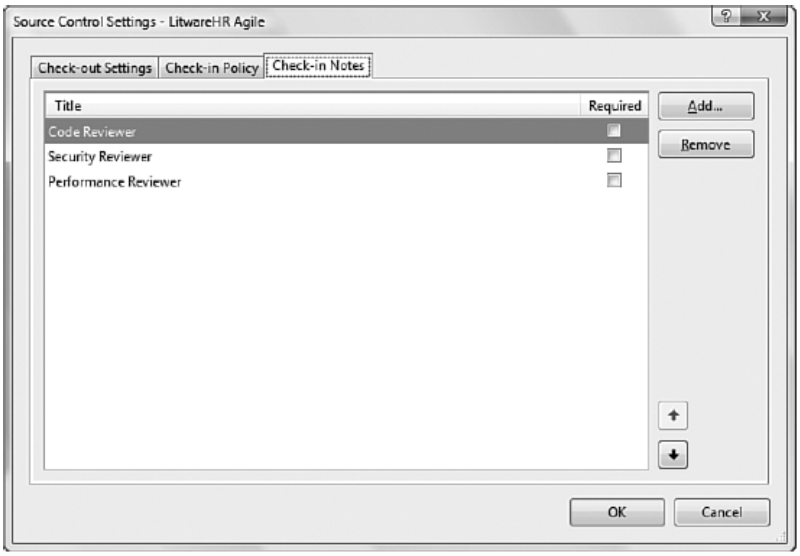
3.2.3.2 Opombe sprostitve

Opombe sprostitve so kratki deli besedila, ki ga lahko pripnemo elementom sprostitve med procesom sprostitve. Opombe sprostitve postanejo del zgodovinskih metapodatkov datoteke, shranjene so v skladišču virov in si jih lahko ogledamo pozneje, da dobimo vtis o zgodovini sprememb elementa. Opomba sprostitve je sestavljena iz imena kategorije in dejanskega besedila opombe. Privzeto so na voljo tri kategorije opomb: pregledovalnik varnosti, pregledovalnik kode in pregledovalnik uspešnosti. Med sprostitvijo kliknemo gumb *Check-in Notes (Opombe sprostitve)*, da vnesemo opombe. Posamezna kategorija opombe bo prikazana s pripadajočim besedilnim poljem, ki vsebuje besedilo opombe.



Slika 6: Okno Check-in Notes (Opombe sprostitve) za dodajanje novih opomb

Opombe sprostitve so lahko obvezne za projekt, lahko pa dodamo tudi svoje kategorije opomb sprostitve. Te nastavitve se upravljajo v pogovornem oknu *Source Control Settings (Nastavitve sistema nadzora nad viri)*. Zavihek *Check-in Notes (Opombe sprostitve)* omogoča dodajanje in odstranjevanje kategorije opomb.

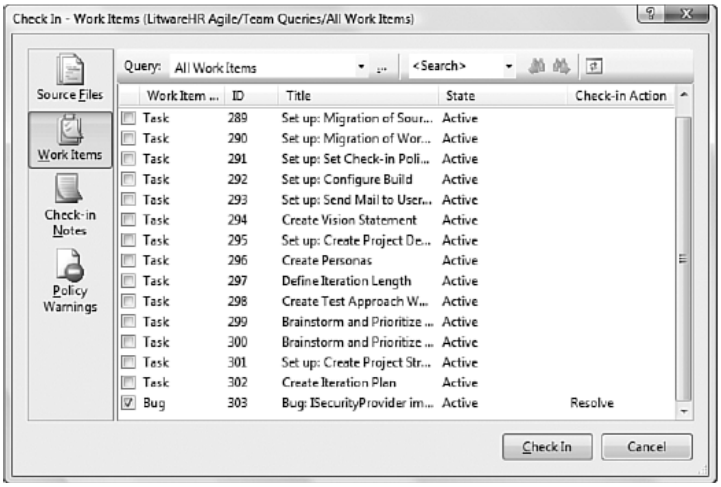


Slika 7: Zavihek Check-in Notes za dodajanje novih kategorij opomb

3.2.3.3 Uporaba delovnih nalog

Delovne naloge se uporabljajo za zastopanje opravil v okviru projekta, od poročil o napakah do tradicionalnih zadolžitev. Delovne naloge so lahko povezane z različnimi artefakti v sistemu Team Foundation; sprostitve so le eden od teh elementov.

S povezovanjem delovnih nalog s sprostitvami je integracija različnih nizov delovnih nalog po vsem projektu v eno povezano predstavitev napredka projekta preprostejša. Razvijalec na primer ustvari knjižnico razredov kot del projektne naloge. Po preskušanju ugotovi, da se eden od razredov ne odziva na preskusni primer, kot bi se moral. Namesto da bi poročal o izjemi, jo razred prekliče. Napaka se v skupinskem sistemu shrani kot delovna naloga. Če želi razvijalec popraviti to težavo, rezervira datoteko, popravi napako in nato sprosti datoteko. V oknu sprostitve lahko enostavno poveže delovno nalogo s sprostitvijo ali pa gre še korak dlje in označi, da ta sprostitev pravzaprav popravi napako, zabeleženo v delovni nalogi.



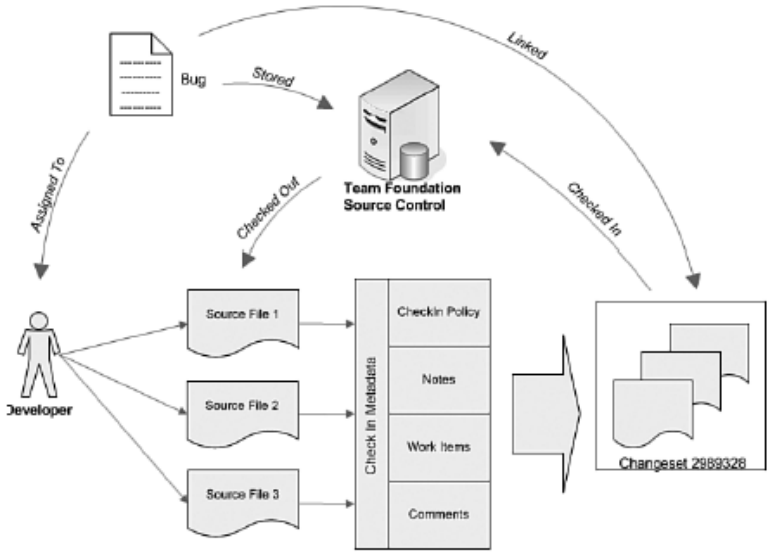
Slika 8: Zavihek Work Items (delovne naloge)

Delovne naloge, ki se prikažejo na seznamu, so rezultat poizvedbe v bazi podatkov. Poizvedbo lahko spremenimo z izbiro spustnega seznama na vrhu okna za delovno nalogo ali pa celo izvedemo iskanje po vseh delovnih nalogah. [3]

3.2.4 Niz sprememb

Niz sprememb je zbirka vseh informacij, povezanih z delovanjem sprostitve. Sem spadajo pregled datotek in map, povezave do sorodnih delovnih nalog, informacije o sprostitvah, komentarji, pravilnik o skladnosti in sistem metapodatkov, kot so ime uporabnika, datum in čas sprostitve.

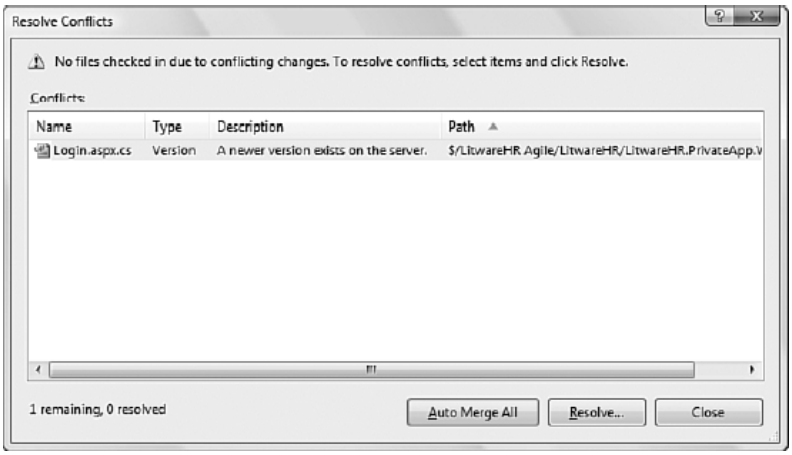
Ko sprostimo niz čakajočih sprememb, TFS ustvari nov niz sprememb v sistemu nadzora nad viri in mu dodeli edinstveno številko niza sprememb. Številke nizov sprememb se povečujejo zaporedno (nizu sprememb #2 na primer sledi niz sprememb #3 itd.). Dva niza sprememb ne moreta imeti istega časa in datuma sprostitve. [5]



Slika 9: Proces nastanka niza sprememb

3.3.5 Spajanje sprememb

Kot je bilo že omenjeno, je rezervacija lahko izključna ali za skupno rabo. Če je za skupno rabo, kjer več ljudi hkrati spreminja isto datoteko, TFS ponuja način, kako spojiti spremembe v nov niz sprememb, ki bo zamenjal trenutno različico datoteke na strežniku.
Primer: Razvijalec A rezervira datoteko in spremeni eno izmed metod v datoteki. Medtem ko je datoteka rezervirana, se razvijalcu B dodeli delovna naloga, ki pravi, da mora spremeniti neko drugo metodo. Razvijalec B rezervira isto datoteko in spremeni metodo. Razvijalec A sprosti datoteko, dan kasneje to stori tudi razvijalec B in tako pride do spora. Ker razvijalec B v delovnem prostoru nikoli ni imel kopije izvorne datoteke s spremembami, ki jih je naredil razvijalec A, je treba ti dve datoteki zdaj združiti. To naredimo z orodjem za spajanje.
Ko razvijalec B sprosti datoteko, sistem za nadzor virov samodejno zazna spor in prikaže se okno *Resolve Conflicts (Reševanje sporov).*

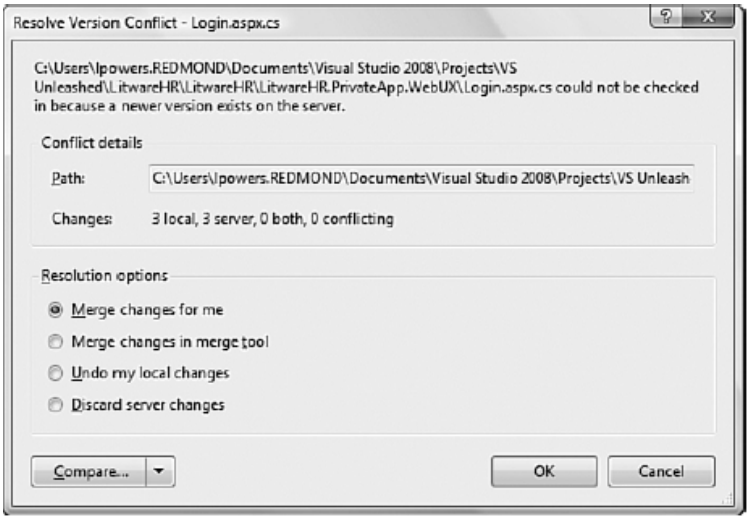


Slika 10: Okno Resolve Conflicts (Reševanje sporov)

Če želimo rešiti spor v opredeljeni datoteki, z gumbom *Resolve (Razreši)* odpremo novo okno *Resolve Version Conflict (Razreši spor različic)*, ki ponuja več podrobnosti o sporu in možnosti za razrešitev spora.

Te možnosti so:

- Dovoliti razvojnemu okolju Microsoft Visual Studio, da samodejno spoji spremembe.
- Spojiti spremembe dveh datotek z orodjem za spajanje.
- Razveljaviti spremembe v lokalni kopiji datoteke.
- Razveljaviti spremembe v strežniški kopiji datoteke.



Slika 11: Okno Resolve Version Conflict (Razreši spor različic)

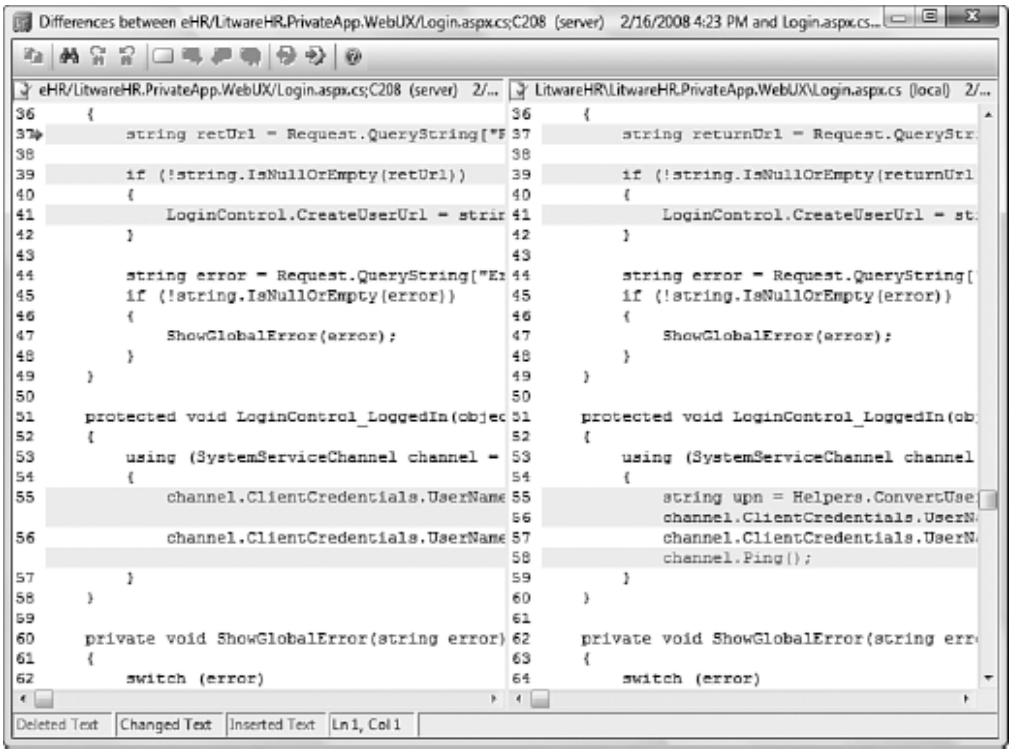
V večini primerov (razen najenostavnejših) moramo uporabiti orodje za spajanje in orodju Visual Studio natančno povedati, kako razrešiti spor. Za pomoč pri razumevanju narave spora lahko uporabimo tudi orodje za primerjavo datotek. [3]

3.2.5.1 Orodje za primerjavo datotek

Orodje za primerjavo datotek omogoča enostaven pregled dveh datotek – strežniške in lokalne datoteke – ter vizualno poudarja besedilne razlike med dvema shemama. Modra barva predstavlja spremenjen tekst, zelena vstavljeno besedilo in rdeča izbrisano besedilo. S tem orodjem ne moremo urejati datoteke. Urejanje datoteke se izvede z orodjem za spajanje. [3]

3.2.5.2 Orodje za spajanje

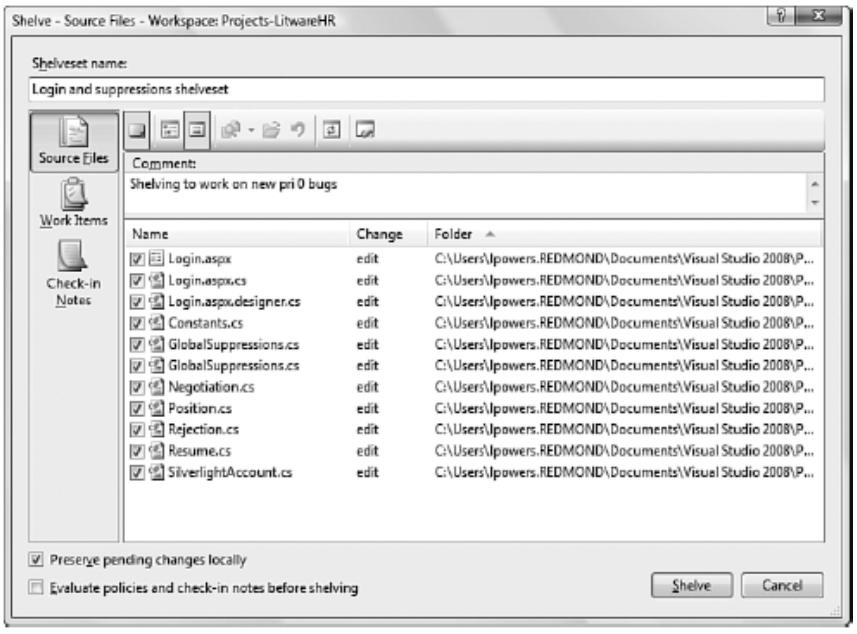
Orodje za spajanje omogoča podoben pogled kot pri primerjavi datotek: poudari besedilne razlike med dvema datotekama. Vendar to orodje vsebuje tudi tretji pogled – rezultat spojene datoteke. [3]



Slika 12: Orodje za spajanje

3.2.6 Odlaganje kode

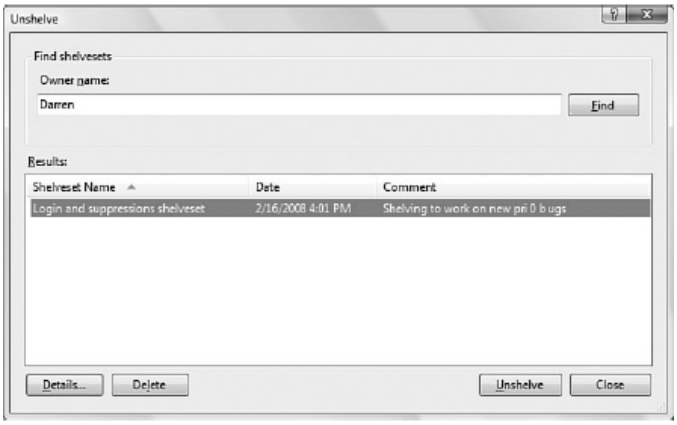
Včasih morajo razvijalci dati na stran trenutno delo ali začeti drugo delovno nalogo, preden so datoteke pripravljene za sprostitev. V takih primerih odlaganje kode omogoča, da vzamemo nekaj čakajočih sprememb ali pa kar vse in jih shranimo v skladišče TFS, ne da bi jih sprostili. Odlaganje kode deluje podobno kot sprostitev in se obravnava v oknu *Shelve (Odloži)*. Ko odložimo kodo, ustvarimo niz odlaganj, ki je enak kot niz sprememb, le da se uporablja samo za odlaganje kode. Niz odlaganj moramo tudi poimenovati, da lahko pozneje dostopamo do njega. [3]



Slika 13: Okno Shelve (Odlaganje kode)

3.2.7 Pridobivanje kode

Ko ustvarimo niz odlaganj, lahko dostopamo do njega, kadar koli želimo. Vse odložene datoteke se pri tem vrnejo v delovni prostor, kjer lahko nadaljujemo z delom. Ta postopek imenujemo pridobivanje kode. [1]



Slika 14: Okno Unshelve (Pridobivanje kode)

3.3 Vejanje

Običajno so veje potrebne za podporo izdajam ali vzporednemu razvoju. Pri številnih preprostih scenarijih ne potrebujemo vejanja, saj je označevanje graditev dovolj. Z uporabo oznak lahko znova ustvarimo graditev na kateri koli točki v prihodnosti ali ugotovimo, katere različice izvorne datoteke so bile uporabljene za ustvarjanje določene graditve. O uporabi vejanja razmišljamo, ko potrebujemo osamitev za vzporedne skupine, in sicer za delo na ločenih, a prekrivajočih se funkcijah, ali za podporo izdaji. [2]

3.3.1 Scenariji za vejanje

To so primeri scenarijev, pri katerih je morda treba ustvariti veje in izvesti spajanja:

- Če imamo pogoste težave s poškodovanimi graditvami, ustvarimo razvojno vejo, da osamimo vzporedna razvojna prizadevanja.
- Če imamo funkcije, ki povzročajo težave s stabilnostjo, oz. skupine, ki druga drugi povzročajo težave s stabilnostjo, ustvarimo ločeno funkcijo ali veje skupine pod mapo razvojnega vsebnika v sistemu nadzora nad viri. [2]

3.3.2 Vzorčne mape in njihov namen

Naslednje mape so primeri map, ki jih moramo ustvariti v sistemu nadzora virov v TFS, ko oblikujemo izvorno drevo za scenarije vejanja.

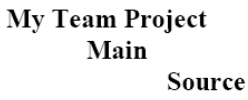
- Mapa *Development (Razvoj)* se razveja iz mape *Main (Glavno)* in se uporablja za osamitev aktivnega razvoja. Razvojne veje so lahko začasne, in sicer za razvijanje tveganih sprememb brez vpliva na mapo *Main.*
- Mapa *Main (Glavno)* vsebuje glavno izvorno drevo. Sem se vdelajo spremembe iz drugih vej.

- Mapa *Releases (Izdaje)* vsebuje veje, ki smo jih že dostavili, vendar jih moramo vzdrževati za stranke. To omogoča osamitev od aktivnega razvoja, ki poteka v razvojni veji. Vsebuje tudi vejo trenutne različice, ki je razvejana iz mape *Main (Glavno)* in vsebuje različico, ki jo trenutno zaklepamo pred izdajo. V tej veji pripravljamo programsko opremo za izdajo, medtem ko ostali nadaljujejo z delom v veji *Development (Razvoj),* kjer delajo na novih funkcijah. [2]

3.3.3 Pogosti scenariji vejanj v praksi

a) Scenarij 1 – Brez vej

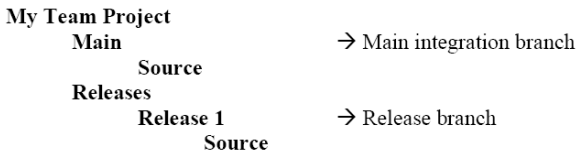
Ta scenarij običajno velja za manjše skupine, ki jih ločeno/izolirano okolje ne zadeva. Z označevanjem graditev pridobimo vir, ki ustreza določeni graditvi. Vejanje ni potrebno, ker lahko delamo neposredno iz mape *Main (Glavno).* Spodaj je prikazan scenarij brez vej. [2



Slika 15: Vejanje - brez vej

b) Scenarij 2 - Veja za izdajo

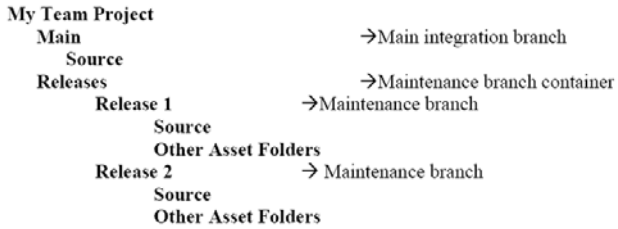
V tem scenariju skupina ustvari vejo za stabilizacijo izdaje in nato spoji vejo za izdajo nazaj v glavno drevo vira, ko je programska oprema izdana. Spodaj je prikazano vejanje za izdaje. [2]



Slika 16: Vejanje - veja za izdajo

c) Scenarij 3 – Veja za vzdrževanje

V tem scenariju ustvarimo vejo za vzdrževanje, tako da ne škodujemo trenutnim proizvodnim graditvam. Spodaj so prikazane veje za vzdrževanje. To je zelo podobno vejam za izdajo, le da se na tej točki veja vzdržuje čez nekaj časa za podporo izdaji. [2]



Slika 17: Vejanje - veja za vzdrževanje

d) Scenarij 4 – Veja za funkcijo

V tem scenariju ustvarimo razvojno vejo, opravimo delo v tej veji in nato spojimo svoje delo nazaj v glavno izvorno drevo. Razvojne veje razvrstimo glede na funkcije izdelka. Spodaj je prikazano vejanje za razvoj funkcij. [2]

My Team Project
Development → Isolated development branch container
Feature A → Feature branch
Source
Feature B → Feature branch
Source
Feature C → Feature branch
Source
Main → Main Integration branch
Source

Slika 18: Vejanje - veja za funkcijo

e) Scenarij 5 – Veja za skupino

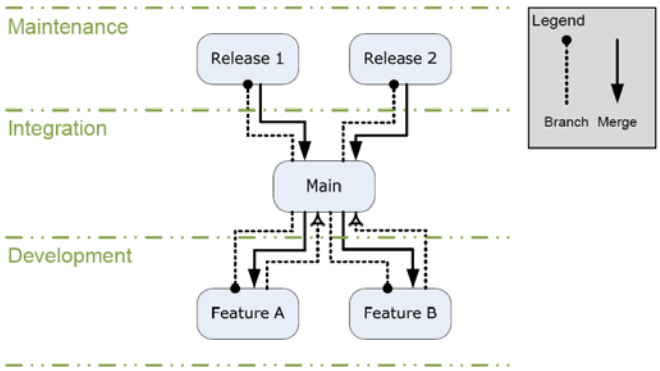
To scenarij je podoben prejšnjemu scenariju vejanja po funkciji, vendar pa tukaj razvrščamo razvojne veje glede na podskupino in ne funkcijo izdelka. Med skupino in funkcijo lahko obstaja korespondenca 'ena na ena', vendar lahko v nekaterih primerih skupina dela na več funkcijah. Spodaj je prikazano vejanje za razvoj podskupin. [2]

My Team Project
Development → Isolated development branch container
Team 1 → Team branch
Feature A → Isolated branch for development
Source
Feature B → Isolated branch for development
Source
Team 2 → Team branch
Feature A → Isolated branch for development
Source
Feature B → Isolated branch for development
Source
Main → Main Integration branch
Source

Slika 19: Vejanje - veja za skupino

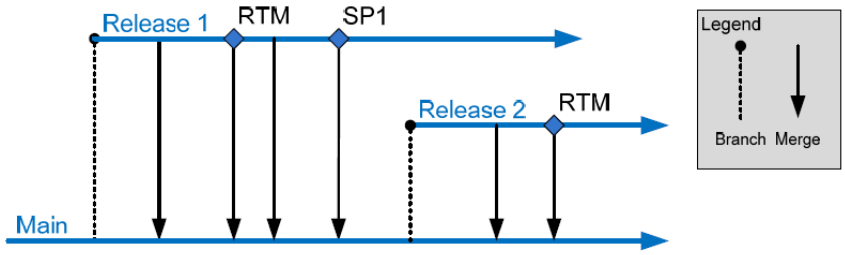
3.3.4 Logična struktura

Logična struktura je za posamezno vejo sestavljena iz razmerja nadrejeno/podrejeno. To se lahko razlikuje od fizične strukture, ki jo vidimo v Source Control Explorerju. V predhodni fizični strukturi sta mapi *Development (Razvoj)* in *Main (Glavno)* videti enakovredni, vendar je mapa *Development* podrejena mapi *Main.*



Slika 20: Logično razmerje ter potek vej in spajanj v strukturi.

3.3.5 Scenarij izdaje



Slika 21: Časovni trak pri vejanju za izdajo

Zaporedje dogodkov je naslednje:
1. Veja *Release 1 (Izdaja 1)* se ustvari iz veje *Main (Glavno)*, ko je izdaja pripravljena za zaklepanje.
2. Periodična spajanja v vejo *Main (Glavno)* omogočajo, da se nekateri popravki napak iz izdaje premaknejo v glavno vejo za integracijo.
3. Graditev izdaje je označena v veji *RTM* in se nato spoji nazaj v vejo *Main (Glavno).*
4. Izda se servisni paket SP1. Graditev je označena, spremembe se spojijo v vejo *Main (Glavno*)**.**
5. Veja *Release 1 (Izdaja 1)* ostaja v podpori servisnega paketa SP1 in omogoča prihodnje servisne pakete.

Ta postopek se ponavlja za prihodnje izdaje. [2]

4. Sledenje delovnim nalogam

Proces razvoja programske opreme v skupini je lahko tako težak kot pisanje kode. Razvijalci se želijo osredotočiti samo na pisanje kode. Vendar pa stranke, vlagatelje v projekt, vodje projektov, preizkuševalce in ostale prav tako zanima spremljanje napredka kode in določitev splošnega stanja projekta. Sodelovanje vseh zahteva zamudna srečanja. Na teh srečanjih se ustvarijo poročila, ki pogosto ne povzamejo dejanskega dogajanja in so že zastarela, ko ima ciljna publika priložnost, da jih pregleda.
Podoben izziv je ustvarjanje dejanskih, smiselnih matrik. Vodje projektov so pogosto prepuščeni interpretaciji nejasnih poročil članov skupine, ki na tedenskih sestankih izmenično poročajo o napredku. Ti podatki predstavljajo le majhen del dejanskega stanja projekta. Zaradi nerealnih statičnih meritev nastaja vrzel med tem, kaj se res dogaja na projektu in poročili. Mnogo podjetij za razvoj programske opreme uspešno zmanjšuje to vrzel. Postajajo vedno boljši pri ocenjevanju, poročanju in spremljanju napredka, in sicer z uporabo tehnologij, kot so Agile, SCRUM, RUP, MSF, CMMI in Extreme Programming. Team Foundation Server ima na voljo metodologijo Agile in CMMI. Omogoča tudi dodajanje novih. Če pa ne želimo dodati novih, pa lahko spreminjamo obstoječe. Te tehnologije so jim v veliko pomoč pri izpopolnjevanju razvoja programske opreme, vendar so se orodja, ki podpirajo te metodologije v razvojni platformi, šele zdaj začela dobro uporabljati.
Z orodjem Team Foundation Server in sledenjem delovnim nalogam lahko napišete odlično kodo, medtem ko še vedno posredujete informacije o stanju in napredku kode. Pomembno je tudi, da vam lahko ostali člani skupine posredujejo pomembne informacije v zvezi z IDE. Te informacije so rezultati testiranj, scenariji, upravljanje zahtev in ostali rezultati.
To poglavje se osredotoča na delovne naloge, ki se jim sledi v programski opremi in zagotavljajo potrebne meritve za proces. [3]

4.1. Delovna naloga

Delovne naloge so primarni način, da lahko vodje projektov in vodje skupin sledijo delu, ki mora še biti narejeno na projektu, kot tudi že narejenemu delu. Člani skupin uporabljajo delovne naloge za sledenje osebni delovni vrsti in za dodeljevanje nalog drug drugemu v obliki napak ali opravil.

Običajna uporaba delovnih nalog v skupinskih projektih:

- ustvarjanje zahteve uporabnikom ali zahteve kakovostnih storitev za aplikacijo
- sledenje razvoju in testiranje glede na zahteve
- ustvarjanje razvojne naloge za predstavitev dela, ki ga je treba izpolniti za izvajanje sestavnih delov aplikacije in funkcij.
- ustvarjanje poročil o napakah v kodi za predstavitev slabosti pri uporabi sestavnih delov in funkcij
- ocenjevanje o težavnosti delovnih nalog in poročil o napakah, da so lahko ustrezno razporejeni v skupini
- sledenje razvojnim nalogam za ugotavljanje napredka glede kode
- sledenje poročilom o napakah v kodi, skupaj z ostalimi kvalitetnimi meritvami, za določitev kakovosti aplikacije in njegovi pripravljenosti za uporabo. [2]

4.1.1 Struktura delovne naloge

Vsaka delovna naloga je lahko opredeljena na naslednji način:

- Je smiselna in ima namen uporabe. Hrošči se na primer uporabljajo za spremljanje napak v kakovosti, naloge se uporabljajo za spremljanje načrtovanega dela, zahteve QoS se uporabljajo za zajemanje kritičnih nefunkcionalnih vidikov, kot so varnost in zahteve glede učinkovitosti.
- Ima potek dela, ki je opredeljen s stanji in prehodi. Na primer koraki od stanja *Open (Odprto)* do *Resolved (Razrešeno*) in *Closed (Zaprto).*
- Ima niz polj, ki jih mogoče nastaviti, zanje izvesti poizvedbe in o njih poročati. Na primer *Priority (Prednost), Status (Stanje)* in *Iteration (Ponovitev).* [2]

4.1.2 Tipi delovnih nalog

Opredelitve delovnih nalog so shranjene v predlogi procesa, ki se izbere, ko se projekt prvič ustvari. Izbiramo lahko med dvema predlogama:

- MSF (Microsoft Solution Framework) for Agile Software Dewelopment(MSF Agile);
- MSF for CMMI Process Improvement(MSF CMMI).

Posamezna predloga določa sklop delovnih nalog, ki vsebujejo vloge in dejavnosti, določene v navodilih. [2]

4.1.2.1 MSF for Agile Software Dewelopment

MSF Agile vsebuje naslednje tipe delovnih nalog:

- ***Hrošč***. Uporablja se za poročilo o težavi s sistemom. Običajno o hroščih poročajo preizkuševalci in uporabniki. Hrošči so zabeleženi in nato dodeljeni za popravilo. Hrošči omogočajo upravljanje z napakami in sledenje.
- ***Tveganje***. Ekipi omogoča sledenje in upravljanje s tveganji projekta. Tveganja projekta predstavljajo vse, kar bi lahko imelo negativen vpliv na projekt v smislu kakovosti, stroškov, rokov itd. Tveganje se označi z izbiro polj za opis. Nekaj dodatnih polj je opredeljenih z natančnostjo in lestvico. Natančnost kaže na verjetnost tveganja, da se pojavi skupaj z vplivom tega tveganja. Natančnost je določena kot kritična, visoka, srednja ali nizka.
- ***Scenarij***. Opredeljuje interakcijo uporabnika s sistemom za dokončanje posebnega cilja ali naloge. Običajno bo scenarij opredelil skupno uspešno pot za dosego cilja uporabnika. Poleg tega se lahko nanaša na nadomestne scenarije, ki določajo alternativne in včasih neuspešne poti skozi sistem. MSF smernice procesa kažejo, da ekipa na začetku zavira širitev seznama možnih scenarijev za sistem. Seveda je treba te scenarije povezati s skupno vizijo projekta. Vsak scenarij se potem dodeli poslovnemu analitiku ali strokovnjaku za opredelitev in opis. Nato so scenariji razčlenjeni na naloge, ki jih bodo člani skupine opravili za realizacijo danega scenarija in s tem projekta.
- ***Opravilo***. Je projektna naloga, ki sporoča članu ekipe, kaj mora narediti pri projektu. Kot ostale delovne naloge, so opravila dodeljena članom ekipe. Vendar pa so opravila običajno delovne naloge, ki določajo razpored projekta. Opravilo je na primer ustvarjanje novega scenarija. To opravilo se lahko dodeli poslovnemu analitiku v ekipi. Ko določimo opravilo, izberemo disciplino, ki ji pripada opravilo. Discipline so

podobne vlogam na projektu; vključujejo arhitekturo, razvoj, projektno upravljanje, upravljanje izdaj, zahteve in testiranje.

- ***Kakovost storitve***. Določa, kako mora določen sistem delovati, ko je končan. Te vrste delovne naloge prihajajo v obliki učinkovitosti delovanja, platforme, stresa, varnostnih zahtev in drugih kategorij. Njihov namen je pojasniti ekipi, kaj se na splošno pričakuje od sistema. Kakovost storitve lahko na primer določa, da se celotna interakcija uporabnikov z uporabniškim vmesnikom izvede v manj kot sekundi. Ta zahteva lahko narekuje, da ekipa namesto odjemalca z brskalnikom ustvari obogateno odjemalsko aplikacijo. Področja, ki se uporabljajo za določanje zahteve QoS, so skoraj enaka tistim za scenarij. Smernice procesa MSF Agile kažejo, da je zahteva QoS v celoti opredeljena znotraj področij delovnih nalog. Zahtevam QoS je mogoče priložiti dodatne dokumente, vendar za to ne bi smelo biti posebne potrebe (za razliko od scenarijev). [3]

4.1.2.2 MSF for CMMI Process Improvement

Vrste delovnih nalog MSF CMMI:

- ***Hrošč***. Predstavlja problem ali potencialni problem v aplikaciji. Podrobneje je opisan že pri predlogi MSF Agile.
- ***Zahteva za spremembo***. Omogoča sledenje spremembam in upravljanje le-teh. Nekateri projekti se lahko preprosto zlijejo s spremembami. Drugi, bolj formalni projekti, pa zahtevajo dokumentacijo o spremembah in oceno njihovega učinka. Ta sprememba je nato predstavljena odboru za nadzor sprememb, ki sprejme odločitev. Če se odloči, da sprejme spremembe, se ustvari nov seznam delovnih nalog, ki povzroči spremembe pri razporedu in stroškov.
- ***Težava***. Omogoča članom skupine, da prijavijo položaj, ki blokira napredek na projektu. To niso tveganja. Težava je resnična stvar, ki zahteva ukrepanje (ne gre za potencialna tveganja). Pri tej vrsti delovnih nalog je treba določiti prioriteto, vpliv na projekt, stopnjevanje in korektivne ukrepe. Prioriteta predstavlja pomembnost reševanja te težave. Vpliv na projekt je opredeljen kot kritičen, visok, srednji in nizek. Polje stopnjevanja predstavlja, ali je potrebno stopnjevati položaj, ker na nek način blokira projekt.
- ***Zahteva***. MSF Agile opredeljuje dve delovni nalogi za opredelitev zahtev (scenarij in kakovost storitve). Na drugi strani pa MSF CMMI določa eno samo zahtevo delovne naloge. Ko ustvarimo novo zahtevo za delovno nalogo pod MSF CMMI, s tem določimo tip zahteve. Nato lahko razlikujemo med scenarijem in zahtevo QoS. Pravzaprav ima delovna naloga zahteve privzeto sedem vrst: funkcionalnost, vmesnik, operativnost, kakovost storitev, varnost, scenarij in varnost.
- ***Pregled***. Omogoča pregled rezultata kode ali pregled dokumenta. Pri večini razvojnih projektov je najbolje, da naredimo pregled scenarijev, zahtev, načrtovanj in kode. Taki pregledi se lahko naredijo v načinu brez povezave ali v obliki sestanka. Poleg tega se pregledi lahko ponavljajo, dokler element ne ustreza pregledu. Pregled ima polje, ki označuje, ali gre pri pregledu za način brez povezave ali za sestanek. Poleg tega vsebuje tudi polja, ki označujejo avtorja pregleda. Pri tem polju je na voljo osem možnosti. Rezultati pregleda se spremljajo s poljem za določanje minut.
- ***Tveganje***. Predstavlja možen dogodek ali pogoj, ki bi lahko negativno vplival na projekt. Podrobneje je opisan že pri predlogi MSF Agile.

- ***Opravilo***. Predstavlja delo, ki ga mora opravit član ekipe. Podrobneje je opisan že pri predlogi MSF Agile. [3]

4.2. Spoznavanje skupnih lastnosti delovnih nalog

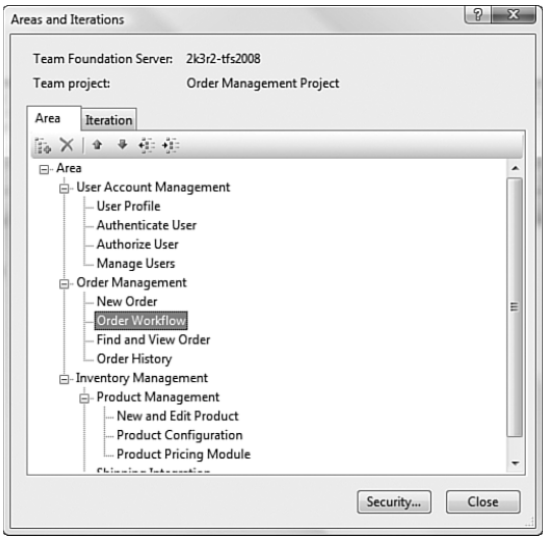
Skupne lastnosti delovnih nalog niso naključje. Sam pojem *delovna naloga* je splošna abstrakcija, njegova dejanska izvedba pa so različne že omenjene delovne naloge (opravila, hrošči, tveganja itd.). Delovne naloge ustvari in upravlja ogrodje delovne naloge. Zato moramo najprej razumeti, kako delovna naloga deluje v temu ogrodju, da lahko razumemo, kako delujejo vse delovne naloge. [3]

4.2.1 Področja in ponovitve delovnih nalog

S področji in ponovitvami delovnih nalog lahko kategoriziramo in razvrščamo delo v skupine ter določamo, kdaj se bo to delo izvedlo. Ta informacija je določena za celoten projekt (običajno jo določi projektni vodja). Področja in ponavljanja so običajno specifična za vsak projekt posebej in jih zato metodologija ne določa vnaprej. Posamezna delovna naloga je nato razdeljena glede na področje in ponovitev. [3]

4.2.1.1 Področja delovnih nalog

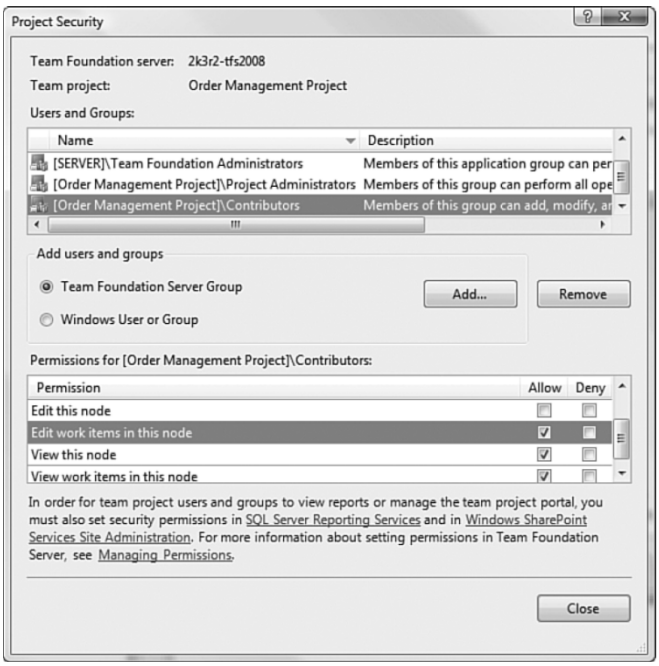
Področje predstavlja kategorijo, ki se uporablja za razvrščanje dela v skupine. Področja so pogosto opredeljena kot moduli ali nabori funkcij. Projekt lahko na primer porazdeli delo tako, da določi modul za profil uporabniškega računa, modul za vnos naročila, modul za zgodovino naročil ali modul za zalogo. Ti moduli ali področja se uporabljajo za razvrščanje opravil in ostalih delovnih nalog v sistemu v skupine. Ta možnost je uporabna za poročanje in sledenje. Področja so lahko definirana tudi v hierarhični strukturi. To omogoča opredelitev podpodročij in podobnega.



Slika 22: Področja delovnih nalog

Področja in ponovitve je mogoče opredeliti v hierarhiji. To omogoča ustvarjanje področij in podpodročij. Orodna vrstica v tem oknu omogoča razvrščanje elementov v pravilnem vrstnem

redu in hierarhiji. Varnostni gumb na dnu okna omogoča nastavitev varnosti, povezane s področji. Označimo lahko na primer, kdo v skupini ima dovoljenje za ustvarjanje, urejanje in ogled delovnih nalog, povezanih z danim področjem. [3]



Slika 23: Varnost področij

4.2.1.2 Ponovitve delovnih nalog

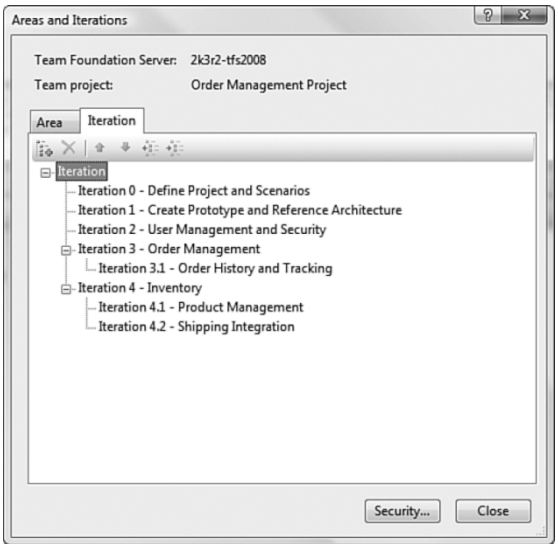
Ponovitev je časovno obdobje, v katerem se opravi del dela na projektu. Namen ponovitve je določiti časovno obdobje za niz delovnih nalog, ki bodo opravljene v določenem časovnem okvirju. Določimo lahko na primer 30-dnevni rok. Poleg tega se ponovitve lahko prekrivajo iz različnih razlogov. Običajno pride do tega zaradi prenosa med člani skupine ali skupinami na projektu.

Določimo na primer štiri ponovitve za projekt. Vsaka ponovitev je določena kot 30-dnevno časovno obdobje. Med vsako od teh ponovitev gremo skozi celoten življenjski krog razvoja za podmnožico funkcij. Med vsako ponovitvijo lahko določimo dva do štiri območja ali module za razvoj.

Recimo, da začnemo prvo ponovitev z arhitekturo in oblikovanjem za prva dva modula. Ko sta modula oblikovana, se ponovitev nadaljuje tako, da oblikovalci predajo specifikacijo razvojni skupini, nato pa začnejo delati na drugi ponovitvi. Ko je prva ponovitev razvita in preizkušena, se jo preda ekipi za zagotavljanje kakovosti. Nato pa lahko razvijalci začnejo kodirati obliko za drugo ponovitev. Ta postopek se nadaljuje skozi ponovitve in tako se ponovitve prekrivajo.

Ta postopek seveda zahteva izkušeno ekipo in napredna orodja za tekoče delo. Poleg tega se ti krogi lahko razlikujejo. TFS omogoča, da določimo ponovitve za svoj projekt in nato razvrstimo delovne naloge glede na njihovo ponovitev.

Pod dano ponovitvijo je mogoče določiti podponovitve. Ta možnost je uporabna za nekatere namene poročanja. Vendar pa večini ekip običajno zadostuje, da imajo za projekt štiri do šest ponovitev najvišje ravni. Za vsako se lahko določi 8-tedenski časovni okvir: dva tedna za oblikovanje, štirje tedni za razvoj ter dva tedna za integracijo in testiranje uporabnikove odobritve. [3]



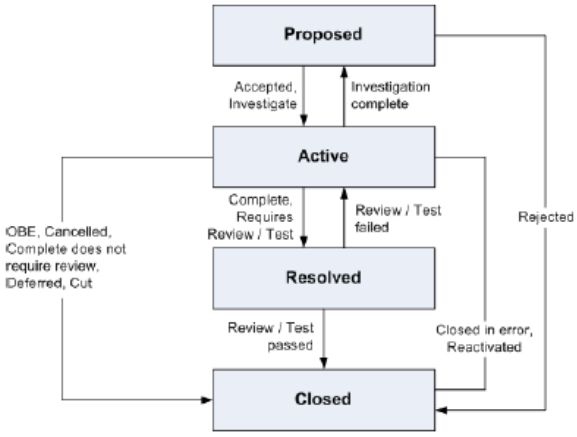
Slika 24: Ponovitve delovnih nalog

4.2.2 Stanja in prehodi delovnih nalog

Vsaka delovna naloga ima vnaprej določen potek dela, ki predstavlja posamezna stanja delovne naloge in prehode med stanji. Vsako stanje delovne naloge je seveda povezano z vlogo v orodju Team Foundation Server. Ko na primer preizkuševalec odpre novega hrošča v MSF Agile je stanje *Active (Aktivno)*. Ko razvijalec popravi hrošča, se stanje spremeni v *Resolved (Rešeno)*. Ko preizkuševalec preveri popravek hrošča, se stanje spremeni v *Closed (Zaprto)*. Naslednji primeri prikazujejo potek dela za dva tipa delovnih nalog.

Opravilo MSF CMMI:

- **Proposed (Predlagano).** Na primer predlagano s strani razvijalca, preizkuševalca ali arhitekta.
- **Active (Aktivno).** Na primer sprejeto od vodje.
- **Resolved (Rešeno).** Na primer rešeno s strani razvijalca.
- **Closed (Zaprto).** Na primer preizkušeno in zaprto s strani preizkuševalca. [2]

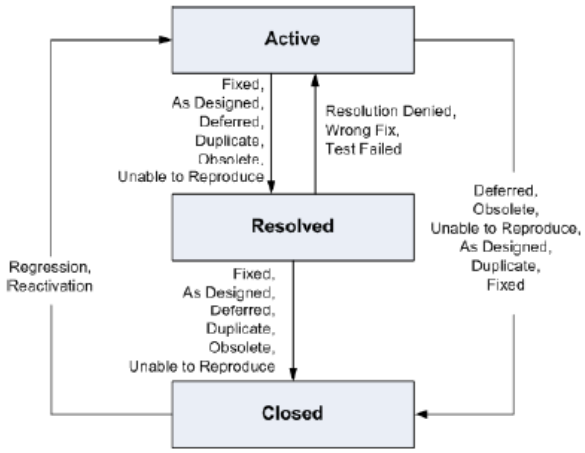


Slika 25: Stanja prehodov za MSF CMMI

Hrošč MSF Agile:

- **Active (Aktivno).** Na primer odprt s strani preizkuševalca.

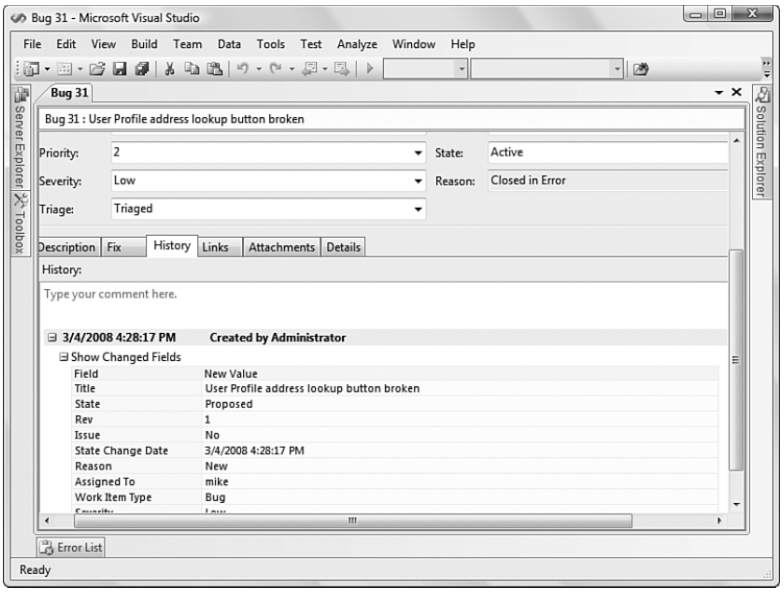
- **Resolved (Rešeno).** Na primer rešen s strani razvijalca.
- **Closed (Zaprto).** Na primer preizkušen in zaprt s strani preizkuševalca. [2]



Slika 26: Stanja prehodov za MSF Agile

4.2.3 Sledenje zgodovini delovne naloge

Ko se delovna naloga spremeni, TFS samodejno zapiše v dnevnik zgodovino te spremembe. Zgodovino naloge lahko pregledamo na zavihku *History (Zgodovina)* za dano delovno nalogo. Vsaka sprememba je shranjena v vnosu. Vnosi so razvrščeni po datumu in času spremembe, skupaj z imenom osebe, ki je spremenila nalogo.
Kot je prikazano na sliki 27, lahko vidimo, katera polja so se spremenila med zapisi zgodovine delovne naloge. Poleg tega lahko uporabnik vnese svoje opombe v vrstico *Type your comments here (Tukaj vnesite komentar)*. Te opombe so vdelane v zgodovino delovne naloge, kar omogoča zabeleženo razpravo za dano delovno nalogo.
Slika 28 prikazuje drug zapis zgodovine za isto nalogo. Ta zapis je rezultat premika napake iz stanja *Closed (Zaprto)* nazaj v stanje *Active (Aktivno)*. V zapisu zgodovine so tudi opombe, zabeležena pa so samo spremenjena polja. [3]

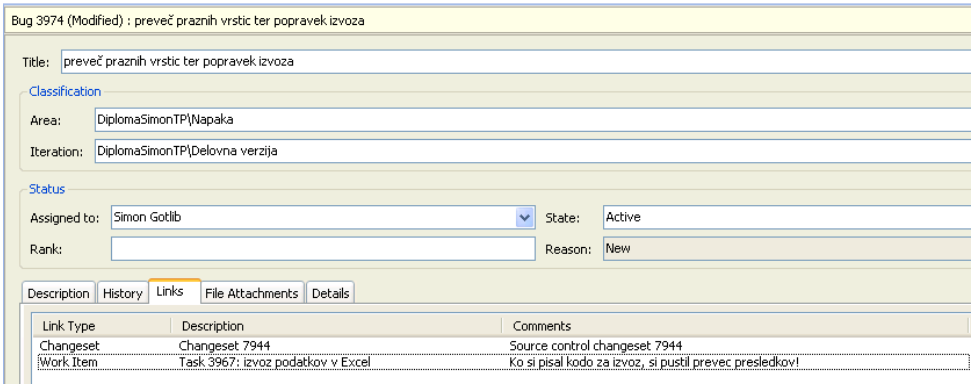


Slika 27: Zgodovina delovne naloge – hrošča

Slika 28: Zgodovina delovne naloge – opravila, kjer je zapis rezultat premika napake iz stanja Closed (Zaprto) nazaj v stanje Active (Aktivno).

4.2.4 Povezovanje delovnih nalog

Delovne naloge moramo pogosto povezati. Povezovanje nalog omogoča boljše razumevanje sistema in dela, ki nastaja. Ljudje se bolje razumejo, poročanje pa je hitrejše in boljše. Recimo, da pišemo scenarij uporabnika. Dobro bi bilo vedeti, katere zahteve so nastale kot rezultat scenarija in katera opravila so nastala iz posamezne zahteve.
Morda pa bi želeli celo povezati težave in spremeniti zahteve v opravila in/ali zahteve.
Na koncu pa lahko z metrikami določimo uspeh dane zahteve ali scenarija. Če scenarij na koncu nima zahtev po spremembi oz. ima manj težav, to pomeni, da je bil dobro napisan in razumljen, ekipa pa se je z njim strinjala. Če ima scenarij veliko težav in zahtev po spremembi, pa to lahko pomeni, da scenarij ni bil razumljen, se z njim niso strinjali ali pa je bil slabo napisan.
Če imamo na primer opravilo, ki je blokirano zaradi težave ali tveganja, lahko povežemo to opravilo z delovnimi nalogami, ki so bile ustvarjene, da odmaknejo blokado opravila. [3]

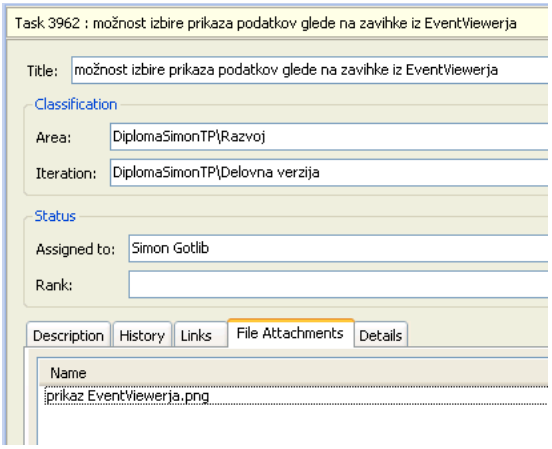


Slika 29: Povezava ene delovne naloge na drugo delovno nalogo.

4.2.5 Priloge delovnih nalog

Datoteke lahko priložimo neposredno v delovne naloge. Ta možnost je koristna, ko želimo na primer priložiti posnetek zaslona za napako ali dokument programa Word, ki je povezan s scenarijem.

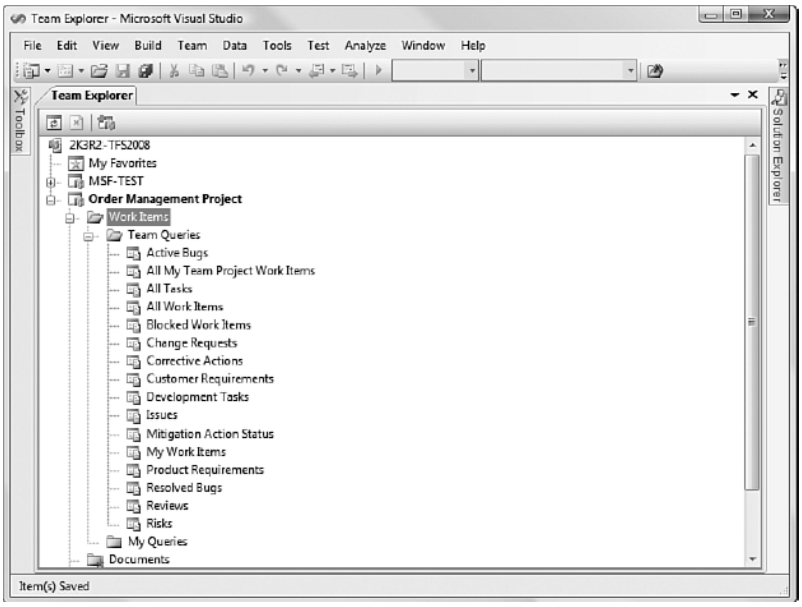
Vmesnik za prilaganje datotek je podoben tistemu za ustvarjanje povezav. Na zavihku *File Attachments (Datotečne priloge)* so navedene vse priloge. Tukaj lahko dodamo, izbrišemo, pregledamo in prenesemo datotečne priloge v delovne naloge. [3]



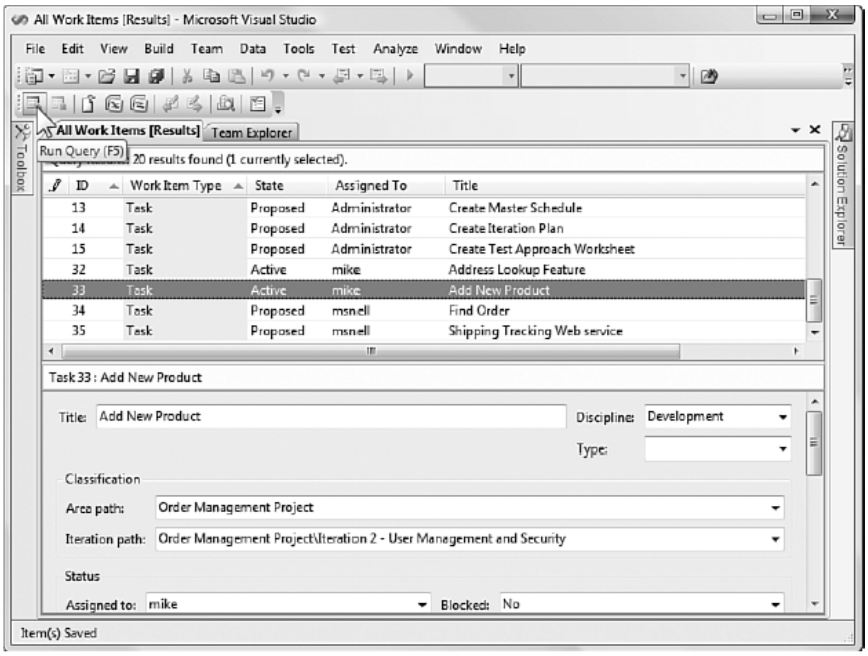
Slika 30: Priloga delovne naloge

4.3 Iskanje in razvrščanje delovnih nalog

Delovno nalogo ali seznam delovnih nalog, s katerimi bi želeli delati, lahko poiščemo z mehanizmom poizvedbe v sistemu Team Explorer. Posamezna poizvedba se izvede v zbirki podatkov delovnih nalog in vrne podmnožico podatkov. Na ta način lahko hitro preklopimo na drugo nalogo.
Številne privzete poizvedbe se premaknejo iz polja (glede na izbrano metodologijo) in jih je mogoče najti v mapi *Team Queries (Poizvedbe skupine).*
Poizvedbe skupine so v skupni rabi z vsemi članom skupine na projektu. To pomeni, da vsak član skupine vidi seznam poizvedb skupine.



Slika 31: Seznam poizvedb za MSF CMMI projekt



Slika 32: Rezultat poizvedbe

Z ustreznimi pravicami lahko določimo novo poizvedbo skupine. Tako lahko pišemo poizvedbe, ki koristijo vsem. Na voljo je tudi mapa *My Queries (Moje poizvedbe)*, v katero lahko shranimo lasten seznam poizvedb. Ustvarimo lahko na primer poizvedbe, povezane z uporabniškim ID-jem ali pa napišemo seznam poizvedb glede na delovno mesto. Rezultate poizvedbe lahko tudi poljubno razvrščamo. [3]

5. Upravljanje projektov

TFS zagotavlja orodja, predloge in poročila, ki nam pomagajo spremljati in podpirati procese razvoja programske opreme. Rezultat je lažja komunikacija v ekipi, prenosi med člani ekipe so samodejni, delovne naloge se lažje dodeljujejo ter jim je lažje slediti, poleg tega se lažje nadzorujejo tudi meritve, ki odražajo ustreznost projekta.

Povzetek upravljanja projektov
Načrtovanje projektov običajno sestoji iz različic naslednjih korakov:

1. **Ustvarjanje vizije.** V tem koraku se ustvari pogled želenega in končnega rezultata projekta, ki si ga delijo vsi zainteresirani pri projektu.
2. **Ustvarjanje scenarijev.** V tem koraku se določi začetni niz scenarijev za uporabo programske opreme. To vključuje uporabo vnosov strank. Vključuje tudi preverjanje scenarijev, s čimer se zagotovi, da so primerni za stranko.
3. **Ustvarjanje niza funkcij za podporo tem scenarijem.** V tem koraku se scenariji razdelijo v določene elemente, ki so zanimivi za stranko, tako da se s stranko lahko pogovorimo o njenih pričakovanjih v zvezi s temi elementi.
4. **Ustvarjanje niza delovnih nalog.** V tem koraku se scenariji in funkcije razdelijo v določena opravila. Če so delovne naloge zapletene, se uporabi ustrezna funkcija ali scenarij.
5. **Razdeljevanje opravil v območja.** V tem koraku se opravila razdelijo v območja. Ta območja so lahko funkcionalna ali pa osnovana na tem, kako je določena skupina organizirana.
6. **Načrtovanje dela.** V tem koraku se lahko vse delo načrtuje vnaprej, lahko pa se razdeli v ponovitve. [2]

5.1 Pogoste težave pri upravljanju projektov

Večina projektnih vodij pri upravljanju procesa razvoja programske opreme danes uporablja številna različna orodja, ki pa nimajo skoraj nič skupnega (če sploh) z orodji, ki jih za svoje delo uporabljajo razvijalci programske opreme. In ravno to predstavlja izziv za vodjo projekta. Pogoste težave vključujejo:

- **Delo z različnimi viri informacij.** Orodja za upravljanje projektov se običajno uporabljajo izolirano, kar vodi do ločenih virov informacij, ki jih ni lahko povezati. Ko želimo ustvariti pomembne matrike, je običajno tudi težko povezati podatke, ki se vzdržujejo z orodji za upravljanje projekta, s podatki, ki jih vzdržujejo drugi člani skupine, na primer napakami, ki se jim sledi z ločenimi sistemi za sledenje.
- **Težave pri zajemanju metrik, povezanih s projektom.** Pridobivanje metrik, povezanih s projektom, je izjemno pomembno pri sledenju stanja in sprejemanju odločitev z dovolj informacijami ter odgovarjanju na osnovna vprašanja, kot je »Ali bo projekt dokončan pravočasno in v okviru proračuna?«. Projektne vodje se pri odgovarjanju na ta vprašanja običajno zanašajo na podatke, ki jih pridobijo s sistemom Microsoft Office Project ali sistemom sledenja napakam, ki ga uporabljajo skupine za razvoj in preizkušanje. Usklajevanje podatkov iz različnih sistemov je težko, zamudno in nagnjeno k napakam. Večina metrik, ki jih ustvarijo orodja, se ne shranjuje oz. se do njih ne dostopa na enoten način. Ustvarjanje poročil je običajno zelo intenzivno ročno delo, ki zahteva veliko kopiranja in lepljena informacij med različnimi orodji.
- **Težave pri zagotavljanju izpolnjevanja zahtev.** Med načrtovanim delom za razvojno skupino in zahtevami stranke ter ključnimi nedelujočimi zahtevami pogosto

pride do razhajanj. Prihaja tudi do vrzeli med načrtovanim delom in dejanskim delom. Pomembne informacije se v teh vrzelih izgubijo, kar povzroči neizpolnjevanje zahtev.

- **Upravljanje procesov in spremembe v procesih.** Povezovanje procesov, ki bi jim skupina morala slediti, je lahko zahtevno opravilo. Uveljavljanje sprememb za reševanje sistematičnih težav, ne da bi vplivali na produktivnost skupine, pa je lahko še težja naloga.
- **Pomanjkljiva komunikacija in sledenje opravilom.** Sodelovanje in kohezija skupine se običajno rešujeta s skupinskimi sestanki in z dodeljevanjem seznamov opravil razvijalcem, tako da se lahko ti osredotočijo na prave stvari. Sledenje napredku posameznega opravila je lahko zahtevno. Poleg tega vodje projektov pogosto zapravijo veliko dragocenega časa z zbiranjem informacij o stanju z različnih načrtov in seznamov. Člani skupine porabijo veliko časa tudi z izpolnjevanjem poročil o stanju in posodabljanjem številnih različnih dokumentov in obrazcev.
- **Zagotavljanje kakovosti.** Napovedovanje števila in težavnosti napak v programski opremi je težko, zato so ocene urnika in stroškov običajno »ugibanja po najboljših močeh«. Te ocene običajno upoštevajo ukrepe ob nepredvidenih dogodkih, katerih število je odvisno od tega, koliko zaupanja ima projektni vodja v trenutno ustreznost projekta. [2]

5.2 Funkcije upravljanja projektov v TFS-ju

Ključne funkcije upravljanja projektov, ki jih ponuja TFS, vključujejo:

- **Upravljanje procesa.** Upravljanje procesa TFS vključuje tako usmeritev procesa MSF kot tudi predloge procesov, s katerimi se nastavijo novi skupinski projekti z vrstami delovnih nalog, poročili in portalom projekta SharePoint ter nastavitvami sistema nadzora nad viri.
- **Varnost in dovoljenja.** Novi projekti vsebujejo privzete skupine in dovoljenja, ki se preslikajo v pogoste vloge razvojne skupine.
- **Centralizirano upravljanje delovnih nalog.** Delovne naloge, vključno z napakami, tveganji, opravili, scenariji, in zahteve za kakovost storitve (QoS) se centralno beležijo, upravljajo in vzdržujejo v zbirki podatkov delovnih nalog TFS.
- **Integracija Microsoft Office Excel® in Microsoft Office Project.** Z uporabo integracijskih funkcij programov Office Excel in Office Project lahko projektne vodje še naprej dostopajo do skladišča delovnih nalog in informacij o načrtu z že poznanimi orodji.
- **Metrike in poročanje.** TFS ponuja storitev za poročanje, ki spreminja podatke za delovanje (na primer delovne naloge, rezultate graditev in rezultate preizkusa), v metrike, ki se shranjujejo v skladišče podatkov TFS. Vnaprej določena poročila omogočajo poizvedbe za številne metrike ustreznosti in kakovosti projekta.
- **Portal projekta.** Za vsak skupinski projekt strežnik TFS ustvari povezan portal projekta, ki uporablja storitve Microsoft Windows SharePoint®. Portal se uporablja za upravljanje dokumentacije, povezane s projektom, ter za hiter pregled ključnih poročil in ocenjevanje trenutnega stanja projekta.

Prednosti

Funkcije upravljanja projekta TFS imajo naslednje prednosti:

- centralizirano upravljanje;
- možnost sledenja;

- integrirano načrtovanje in razvrščanje projektov;
- izboljšan nadzor nad procesom;
- izboljšana komunikacija in povezanost skupine;
- natančno poročanje o napredku. [2]

5.2.1 Upravljanje procesa

V strežniku TFS je življenjski cikel razvoja programske opreme bistven del orodij za podporo pri razvoju programske opreme. Z uvajanjem procesov življenjskega cikla so procesi in dodeljevanja za sodelovanje razvojne skupine v celoti podprta z orodji in so lahko tako na številnih področjih avtomatizirani.

Predloge procesa
TFS pri nastavljanju novih projektov uporablja predloge procesov za določanje niza navodil in artifaktov, kot so dokumenti za usmerjanje procesa, predloge dokumentov, privzete delovne naloge itd. Predloga procesa je niz navodil, ki razvojnim skupinam zagotavljajo metodologijo za razvoj programske opreme. Predloga procesa vsebuje naslednje elemente:

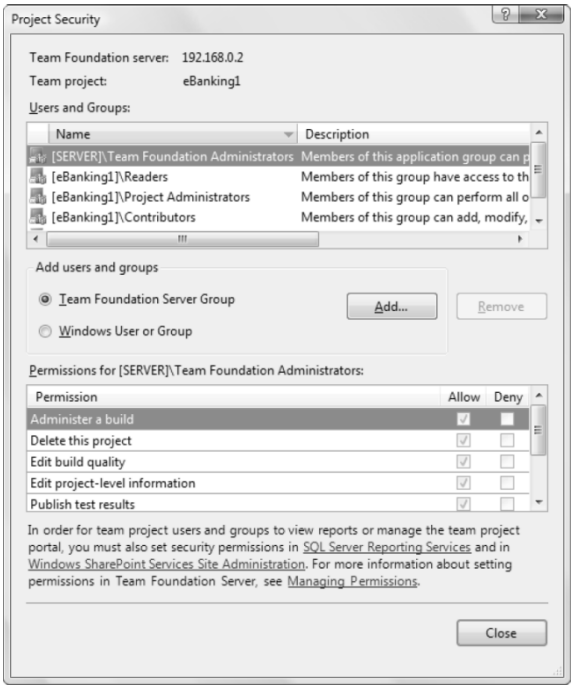
- **Usmeritev procesa.** To je na voljo za vse predloge in ponuja kontekstne informacije, pomoč in navodila za člane skupine, ki potrebujejo pomoč pri sledenju ali razumevanju določene dejavnosti. Usmeritev procesa je del sistema za pomoč v programu Visual Studio (Visual Studio Help System).
- **Predloge dokumentov.** Te predloge članom skupine omogočajo konsistentno ustvarjanje novih dokumentov, kot so specifikacije, ocene tveganj in načrti za projekt.
- **Delovne naloge in potek dela.** Delovne naloge imajo svoj niz polj in pravil, ki določajo potek dela delovne naloge in kako člani skupine dodeljujejo in opravljajo delo.
- **Varnostne skupine.** Te skupine se uporabljajo za določanje oseb, ki lahko nadzorujejo in spreminjajo poročila, delovne izdelke (na primer izvorne kode in dokumentacijo) in delovne naloge. Projektni vodje lahko upravljajo varnostne skupine, ne da bi morali biti skrbniki v sistemu Windows.
- **Pravilniki za sprostitve.** Ti pravilniki se uporabljajo za uvajanje pravil in vrat kakovosti za vse kode v sistemu nadzora nad viri. Uveljavimo lahko na primer, da sproščena koda ustreza določenim kriterijem, npr. da je v skladu s standardi za kodiranje podjetja ali da uspešno opravi preskuse enote.
- **Poročila.** Ta se uporabljajo za nadzor učinkovitosti delovanja in stanja projekta v teku. V TFS-ju so na voljo številna vnaprej določena poročila, vključno s poročili o kakovosti kode, poročili o napredku urnika, poročili o učinkovitosti preskusov in drugimi. Ustvarimo lahko svoja poročila in prilagodimo obstoječa. [2]

5.2.2 Varnost in dovoljenja

Ko ustvarimo projekt v TFS-ju, se privzete skupine za ta projekt ustvarijo ne glede na izbiro predloge procesa. Posamezna skupina ima privzet niz dovoljenj, določenih zanjo, ki določajo, za kakšna opravila so pooblaščeni člani skupine:

- skrbnik sistema;
- sodelavec;
- bralec;
- storitve graditve.

Za projekt lahko ustvarimo varnostne skupine, da bi bolje izpolnjevali varnostne zahteve svoje organizacije. Ustvarjanje varnostnih skupin je učinkovit način za podelitev določenega niza dovoljenj skupini uporabnikov pri skupinskem projektu. Dodelimo le tista dovoljenja, ki so nujno potrebna za skupino, in dodamo le tiste uporabnike ali skupine, ki morajo pripadati tej novi skupini skupinskega projekta.
Ko ustvarimo skupino za skupinski projekt, moramo dodati novo skupino, ji dodeliti ustrezna dovoljenja in v skupino dodati člane. Privzeto se skupina za skupinski projekt ustvari brez podeljenih dovoljenj. [2]



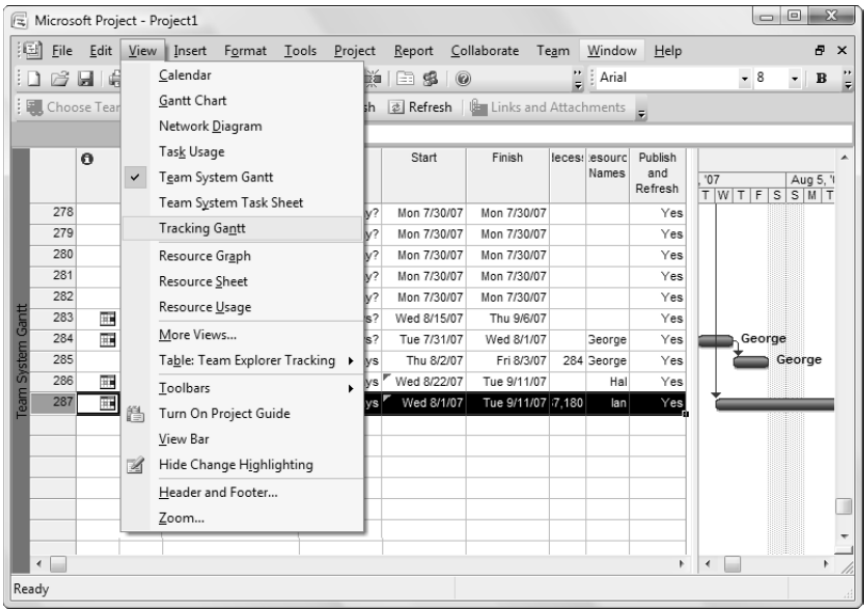
Slika 33: Varnost projekta

5.2.3 Upravljanje delovnih nalog

Delovne naloge se uporabljajo kot enote dela za komunikacijo in sodelovanje v skupini. Izbrana predloga procesa ponuja začetni niz delovnih nalog, projektni vodje pa potem ustvarijo in upravljajo dodatne delovne naloge, ki jih je treba dokončati na razvojnem projektu. Delovna naloga lahko določa opravilo, tveganje, scenarij, napako ali zahtevo za kakovost storitve (QoS). Delovne naloge lahko povežemo, da jim lažje sledimo. Povežemo lahko na primer določeno opravilo delovne naloge s povezanim scenarijem ali zahtevo za kakovost storitve, na katero se nanaša delovna naloga.
Predloga procesa ponuja definicije za vrste delovnih nalog, vključno z nizom polj, določenih za posamezno vrsto delovne naloge. Zato je izbira predloge procesa pomembna, saj je med projektom ni mogoče spremeniti. Po potrebi lahko prilagodimo predlogo procesa in vključimo dodatne vrste delovnih nalog, ki niso na voljo v osnovnih predlogah.
Številne vnaprej določene delovne naloge se ustvarijo v predlogah procesa MSF for Agile Software Development in MSF for CMMI Process Improvement, ko ustvarimo nov skupinski projekt. Te delovne naloge lahko uporabimo za hitrejši začetek uporabe procesa, saj vsebujejo opravila, ki jih treba dokončati, če želimo začeti proces razvoja programske opreme. [2]

5.2. 4 Integracija programa Microsoft Project

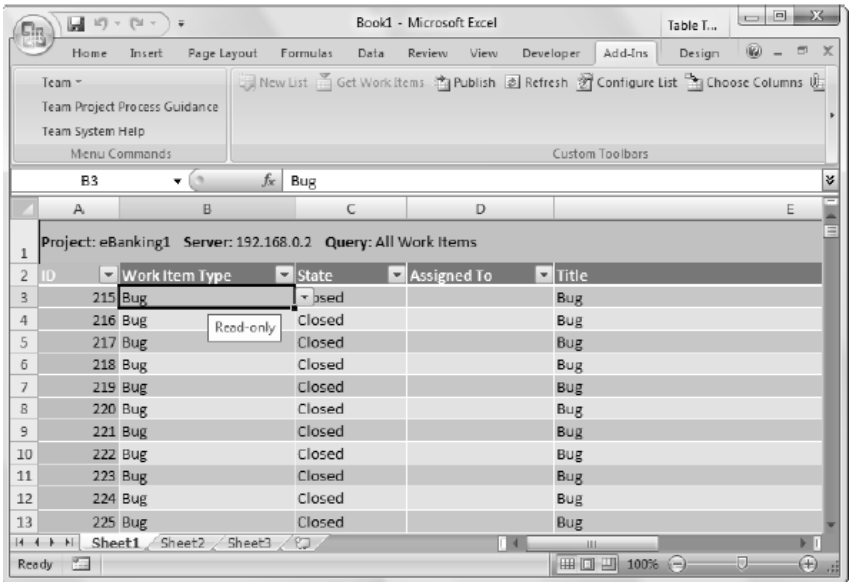
Namestitev TFS-ja ali aplikacije Team Explorer omogoča razširitve za Microsoft Project. Pri velikih projektih, ki vključujejo veliko virov, lahko uporabimo Microsoft Office Project za spreminjanje podatkov urnika v TFS-ju.
Microsoft Project lahko na primer uporabimo za upravljanje in načrtovanje, dodeljevanje, uravnavanje in sledenje delu, nato pa objavimo posodobitve v zbirko podatkov delovne naloge (ko so posodobitve pripravljene), da jih lahko uporabljajo tudi drugi člani skupine. [2]



Slika 34: Prikaz podatkov v Microsoft Projectu

5.2.5 Integracija programa Microsoft Excel

TFS ali aplikacija Team Explorer omogoča razširitve za Microsoft Excel. Pri projektih, ki vključujejo veliko delovnih nalog, lahko uporabimo funkcijo za integracijo programa Excel, da ustvarimo delovne naloge v razpredelnici programa Excel in prenesemo zbirko podatkov delovne naloge, tako da jo lahko uporabljajo tudi drugi člani skupine. [2]



Slika 35: Prikaz podatkov v Microsoft Excelu

5.2.6 Napredek in poročanje

Poročila v TSF-ju nam pomagajo hitro oceniti stanje skupinskega projekta, kakovost programske opreme v razvoju in napredek projekta. Ta poročila se ustvarijo iz podatkov v skladišču podatkov TFS in povzemajo metrike, ki nastanejo iz delovnih nalog, nadzora nad viri, rezultatov preskusa in graditev.

Poročila lahko na primer uporabimo, da izvemo, kako hitro skupina dela iz tedna v teden (glede na dejanske dejavnosti skupine). Cilj sistema za poročanje TFS je omogočanje enotnih podatkov iz komponent VSTS, s katerimi vodje projektov in člani skupine lažje razumejo stanje projekta za razvoj programske opreme in ustrezno ukrepajo, da zagotovijo njegov uspeh.

Predloga procesa, ki jo uporabljamo za ustvarjanje skupinskega projekta, določa, katera poročila so privzeto na voljo, dodamo pa lahko tudi svoja poročila po meri. Team Foundation Server je vdelan v Microsoft SQL Server™ 2005 in uporablja SQL Server za shranjevanje vseh podatkov, povezanih z delovnimi nalogami, atributi kakovosti, preskušanjem, rezultati preskusov in rezultati graditev. TFS nato uporabi storitve za analiziranje SQL Server Analysis Services, da zbere in analizira podatke in zažene poročila. Poročila, ki se ustvarijo s predlogo procesa oz. ki jih ustvarijo posamezni člani skupine s programom Report Designer za Microsoft Office Excel ali Visual Studio 2005, so na voljo v storitvah za poročanje SQL Server 2005 Reporting Services in na spletnem mestu portala SharePoint. [2]

6. Graditev

Team Foundation Build (TFB) je orodje, ki ga opredelimo kot del razvojnega okolja Team Foundation Server. Med ustvarjanjem projekta pridemo do točke, ko moramo spojiti vse različne dele projekta, jih prevesti, razporediti in jih preizkusiti. Ta celoten proces imenujemo graditev. Graditev programske opreme je veliko več kot enostavno prevajanje kode v izvršljive binarne datoteke, saj vključuje ogromno lastnega dela različnih članov projektne skupine. Ta proces predstavlja nekaj edinstvenih težav:

- Pomanjkanje celovitega, edinstvenega sklopa orodij graditve, ki se združujejo z projektnim okoljem, vodi do nepredvidljivih in neponovljivih začasnih procesov graditve.
- Spremljanje procesa graditev, izboljševanje ter razumevanje ustreznosti graditve je lahko težavno.
- Mehanizmi za odpravljanje težav, ki nastanejo v graditvi, niso na voljo.

Team Foundation Build je namenjen avtomatizaciji več vidikov procesa graditve in reševanju zgoraj navedenih težav, tako da se celotni projektni skupini zagotovijo orodja za ustvarjanje in analiziranje graditev.

Številne razvojne organizacije uporabljajo laboratorije graditve za ustvarjanje javnih in zasebnih graditev svojih lastnih različic programov. Laboratorij graditve je seznam virov strojne in programske opreme, ki vzamejo vse datoteke izvorne kode v skupinskem projektu, jih centralizirajo na strežniku graditve in nato sestavijo sistem z vsemi zadnjimi spremembami. Ta graditev se nato zapakira in kopira na centralno lokacijo, do katere lahko dostopajo vsi člani skupine. Viri za zagotavljanje kakovosti lahko nato izvedejo testiranja programske opreme in določijo, kaj je potrebno še spremeniti. Te ugotovitve nato pošljemo skupini za razvoj.
Cilj graditve je izločiti čim več ročnega dela pri procesu graditve, medtem pa članom skupine omogočiti zbiranje informacij o ustreznosti posameznih graditev. Platforma graditev predstavlja različne prednosti za člane skupine:

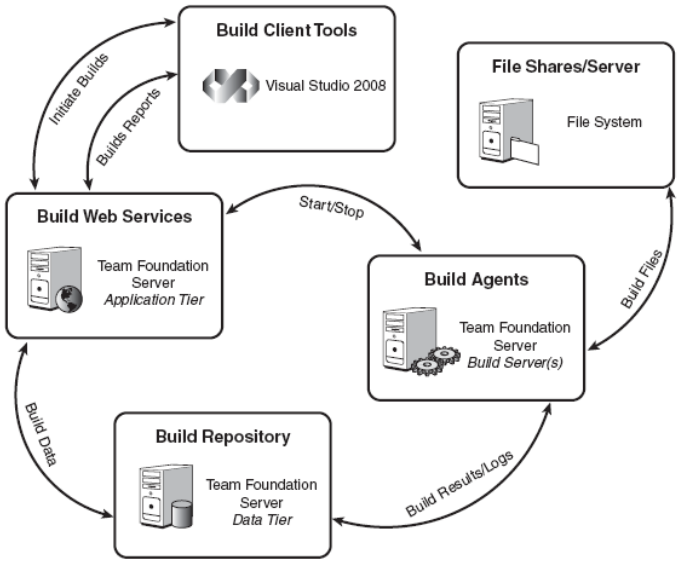
- omogoča hitro in enostavno grajenje procesa graditve z razvojnim okoljem Microsoft Visual Studio;
- zagotavlja ponavljajoč se in neprekinjen proces graditve programske opreme na osnovi določenih dejavnosti, kot so sproščanje ali časovni intervali;
- članom skupine zagotavlja potreba orodja za določanje ustreznosti graditve;
- omogoča primerjavo »od graditve do graditve« za pregled zgodovine napredka celotnega projekta med ustvarjanjem ali preizkušanjem posamezne graditve. [3]

6.1 Arhitektura graditve Team Foundation

Storitve graditve Team Foundation omogočajo štirje podsistemi v ekosistemu VSTS: orodja odjemalca, storitve graditve, spletne storitve graditve in skladišče graditve. Vsaka od teh štirih komponent igra ključno vlogo pri zagotavljanju, da graditev temelji na zanesljivi osnovi. Naslednji seznam opisuje vse štiri komponente:

- *Orodja odjemalca graditve* – Ta komponenta opredeli graditev (ki ji lahko določimo različico in jo shranimo v sistem nadzora nad različicami) in nato začne graditev.

- *Spletne storitve graditve* – Ta komponenta je del aplikativnega sloja Team Foundation Server. Njena naloga je »poslušati« zahteve za graditev, ki jih posredujejo orodja odjemalca graditve. Nato zahtevo za graditev posreduje storitvam graditve.
- *Agenti graditve* – Ti pravzaprav predstavljajo strežnike graditve v aplikaciji. Ti agenti skrbijo za to, da se storitve graditve izvajajo na teh strežnikih. Agenti tudi dejansko izvršijo graditev. To naredijo tako, da najprej pregledajo točno definicijo graditve. Nato vzamejo potrebne datoteke, zaženejo teste in zabeležijo rezultate v skladišče graditve. Obvestila o zaključku graditve se lahko nato pošljejo skupini.
- *Skladišče graditve* – Tipična vloga podatkovnega sloja strežnika Team Foundation Server je skladišče podatkov. Tukaj so shranjene informacije o graditvi in dogodkih. To odjemalcu graditve omogoča ogled in analiziranje dane graditve. [3]



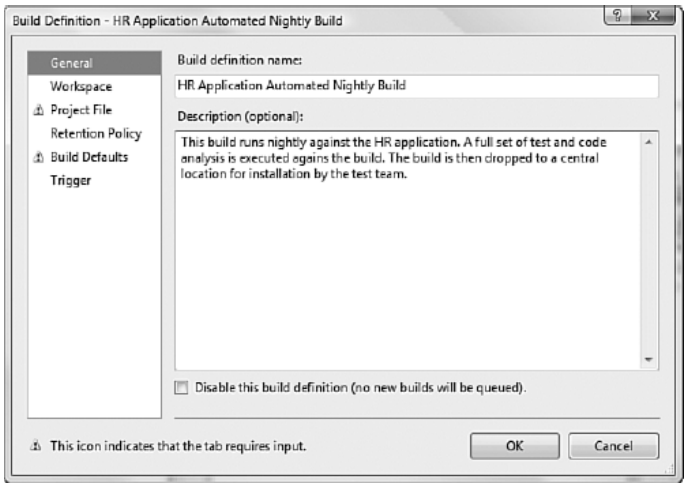
Slika 36: Prikaz, kako podsistemi delujejo eden z drugim

6.2 Ustvarjanje nove graditve

Graditev Team Foundation uporablja koncept definicij graditve. Definicija graditve je enostavna shramba za vse informacije konfiguracij, povezanih z graditvijo. V bistvu definira vse različne dele za vsako graditev posebej.
Pri ustvarjanju nove graditve se najprej odpre čarovnik za definicijo graditve, ki nas popelje skozi postopek določanja delovnega prostora, izbire datoteke za graditev, konfiguracije testiranja in analize, določanja agenta graditve in časovne opredelitve graditve. Ko končamo s čarovnikom, se konfiguracija graditve zapiše v posebno obliko datoteke, ki jo uporablja projekt graditve.

a) Poimenovanje graditve

Prvi korak je določitev imena graditve. To je isto ime, ki je določeno v vozlišču graditev znotraj okna Team Explorer.

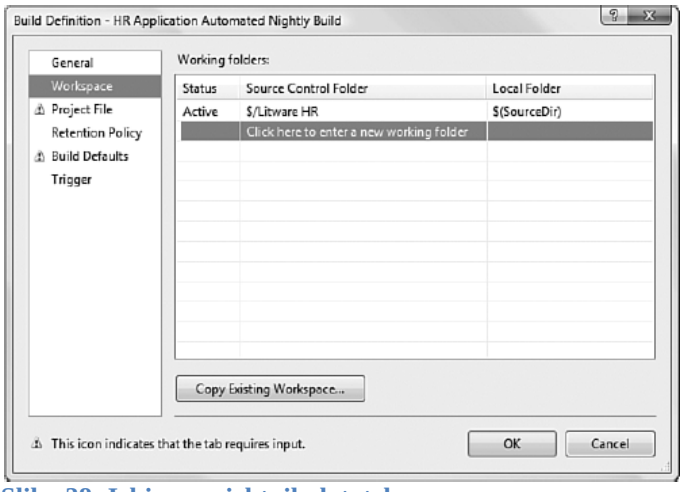


Slika 37: Izbira imena graditve

b) Izbira projektnih datotek za graditev

Drugi zaslon v oknu *Build Definition (Definicija graditve)* je zaslon *Workplace (Delovni prostor)*. Tukaj lahko konfiguriramo graditev tako, da temelji na eni ali več map sistema nadzora nad viri. Imamo lahko več primerkov (vej) iste kode v različnih mapah. Ali pa moramo za dokončanje graditve kodo pridobiti iz več projektov v okviru sistema nadzora nad viri.

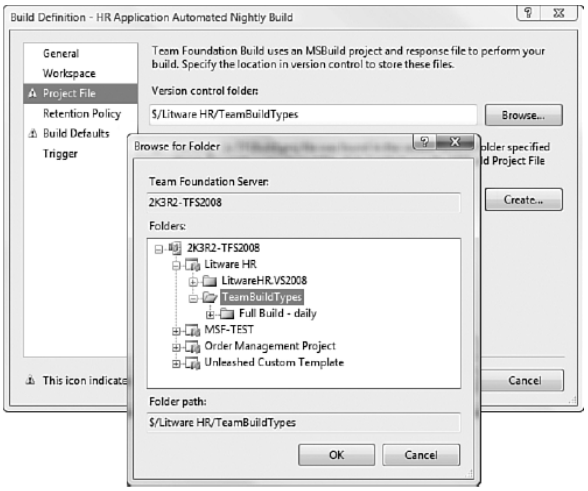
Vse mape sistema nadzora nad viri lahko preslikamo v mapo računalnika za graditev, ki se imenuje lokalna mapa. Spodnja slika prikazuje primer določanja delovnih map za graditev. Tu imamo samostojno mapo sistema za nadzor nad viri, ki je preslikana v eno lokalno mapo na strežniku graditve.



Slika 38: Izbira projektnih datotek

c) Določanje datoteke projekta graditve

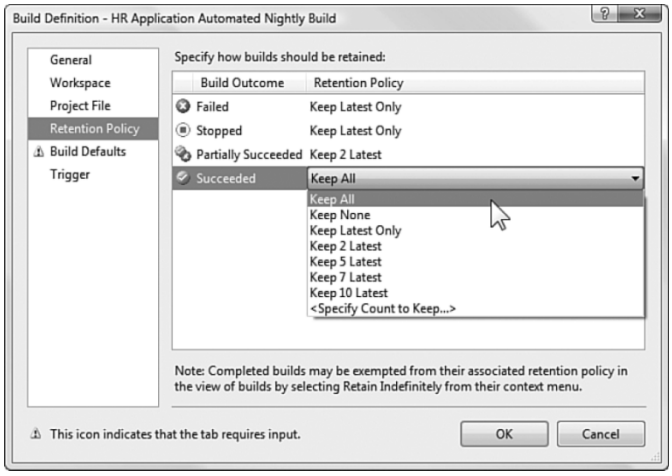
Izbrati moramo mesto za shranjevanje projekta graditve. Upoštevati moramo, da bo ta graditev imela svojo različico. Zato moramo izbrati mapo v sistemu nadzora nad viri, kamor želimo shraniti projekt.



Slika 39: Definicija datoteke graditve

d) Določanje pravilnika o hranjenju graditev

S pravilnikom o hranjenju graditev lahko označimo, koliko graditev naj se ohrani na strežniku ob nekem določenem času. Ta pravilnik omogoča sistemu graditve, da počisti starejše graditve in tako povrne prostor na strežniku. Pravilnik o hranjenju se lahko nastavi glede na rezultat graditve: *failed (neuspešno)*, *stopped (ustavljeno)*, *partially succeeded (delno uspešno)* ali *suceeded (uspešno)*. Za vsakega lahko določimo, koliko graditev želimo obdržati.

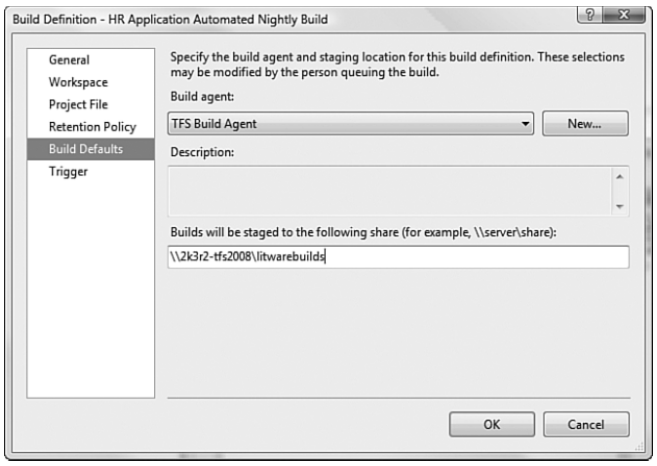


Slika 40: Definicija pravilnika hranjenja

e) Določanje konfiguracije za agenta graditve

Naslednji korak je, da povemo graditvi, kateri strežnik naj uporabi za agenta graditve. Ta strežnik moramo pripraviti kot strežnik graditve, tako da zaženemo ustrezne zagonske datoteke iz namestitvenega medija strežnika Team Foundation Server; tako se bo storitev graditve uvedla na strežnik, ki se mora seveda izvajati, preden lahko zaženemo graditev.
Mehanizem graditve mora vedeti, kam na lokalnem strežniku shraniti datoteke graditve in kam shraniti rezultat, da lahko do njega dostopa ekipa.

Agenti graditve so dodani strežniku Team Foundation Server in jih je mogoče znova uporabiti kadar koli pri graditvi.

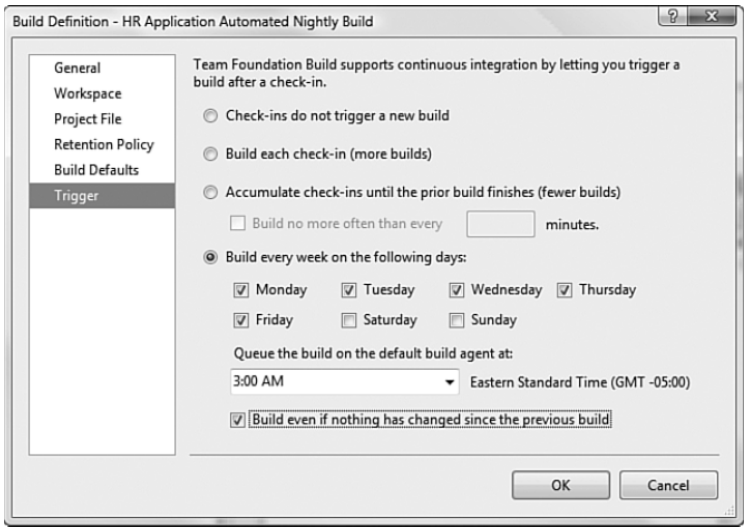


Slika 41: Določanje konfiguracije za agenta graditve

f) Načrtovanje urnika graditve ali nastavljanje sprožilcev

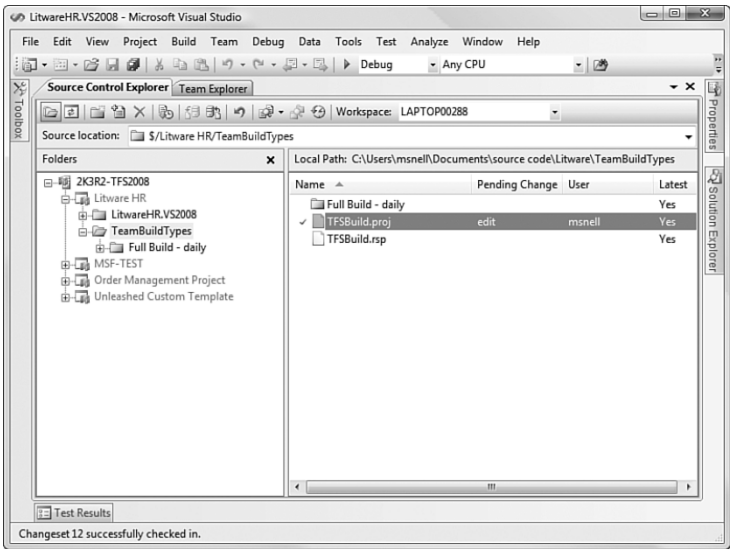
Zadnji zaslon v oknu *Build Definition (Definicija graditve)* je zaslon *Trigger (Sprožilec).* Tukaj določimo, kdaj želimo samodejno sprožiti graditev. Določimo lahko, da se graditev izvede vedno, ko se vir prijavi v strežnik. Določimo lahko tudi, da se graditve izvedejo ena za drugo, odvisno od sprostitev, ki se nanašajo na izvajanje graditev. Če se graditev že izvaja in pride do ene ali več sprostitev, se v tem primeru nova graditev ne izvede, dokler se trenutna ne konča.

Določimo lahko tudi, da se graditev izvede iz tega vmesnika. Tukaj lahko določimo dan v tednu in čas, ko naj se izvede graditev.



Slika 42: Načrtovanje urnika graditve ali nastavljanje sprožilcev

Ko končamo definicijo graditve, pride do dveh stvari: različne nastavitve graditve se zapišejo v datoteko XML, ki se imenuje TFSBuild.proj, in ta datoteka XML je dodana v sistem nadzora nad viri na lokacijo, ki jo določimo med postopkom definicije graditve. [3]



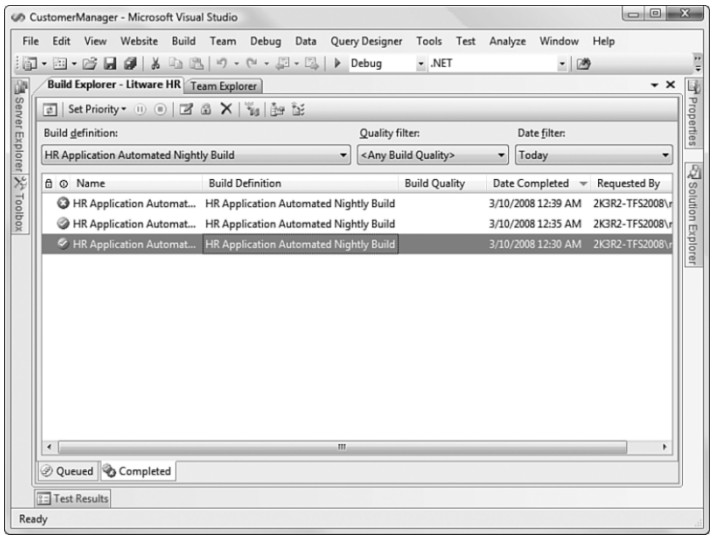
Slika 43: Prikaz projekta graditve (TFSBuild.proj)

6.3 Spremljanje in analiziranje graditev

Informacije graditve so na voljo v orodju Visual Studio prek Team Build Explorerja. To okno brskalnika ponuja seznam zaključenih graditev ali graditev v teku in deluje kot glavni mehanizem za ogled napredka graditve ali zaključenih kratkih poročil.

a) Predstavitev Team Build Explorerja

Team Build Explorer omogoča posnetek katere koli graditve, ki označuje, ali je graditev uspela oz. ni uspela oz. še poteka, ime graditve, kakovost graditve in datum, ko je bila graditev zaključena.



Slika 44: Team Build Explorer

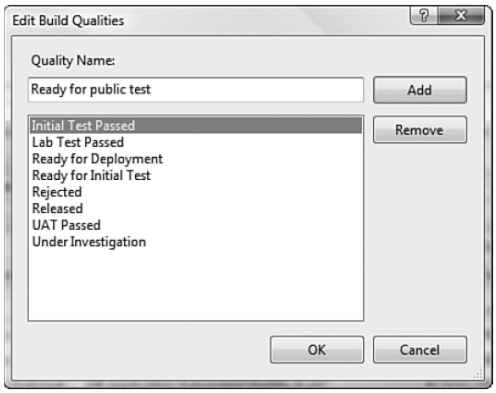
Brskalnik omogoča dostop do poročila za določeno graditev in nastavitev stanja kakovosti graditve.

b) Nastavitev stanja kakovosti graditve

Graditev vsebuje seznam stanj kakovosti, med katerimi lahko skupina za zagotavljanje kakovosti izbira, ko želi označiti kakovost dane graditve:

- Initial Test Passed (Začetni preizkus opravljen)
- Lab Test Passed (Laboratorijski preizkus opravljen)
- Ready For Deployment (Pripravljeno za uvedbo)
- Ready For Initial Testing (Pripravljeno za začetni preizkus)
- Rejected (Zavrnjeno)
- Released (Izdano)
- UAT Passed (UAT opravljen)
- Under Investigation (Se preiskuje)

Kot del procesa izgradnje skupina za zagotavljanje kakovosti odpre brskalnik Team Build in spremeni stanje kakovosti graditve, da ostalemu delu ekipe pokaže, kaj so našli preizkusi.

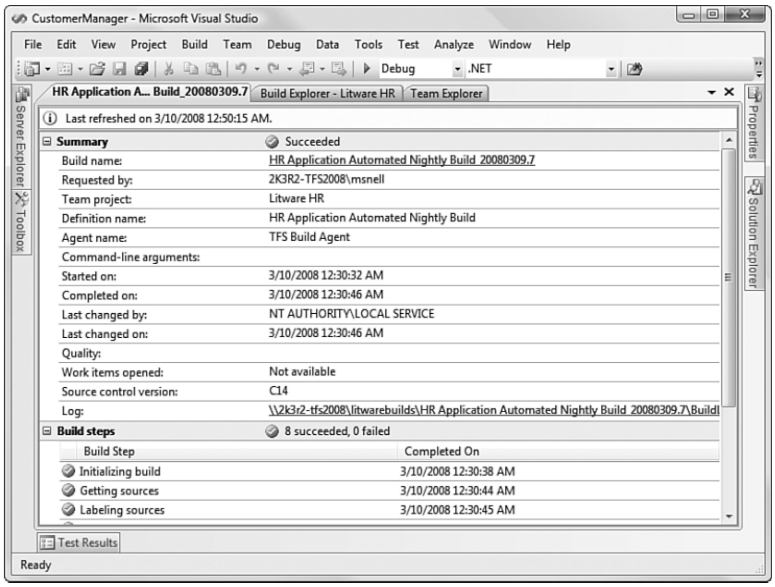


Slika 45: Stanje kakovosti graditve

c) Poročilo o graditvi

Za prikaz poročila o graditvi v teku in dokončani graditvi dvokliknemo graditev v brskalniku Team Build. Posamezno poročilo gostuje v odprtem oknu dokumenta v orodju Visual Studio in ima naslednje razdelke:

- *Summary (Povzetek)* – povzame podrobnosti graditve in vključuje podatkovne točke, kot so ime graditve, oseba, ki je zahtevala graditev, računalnik, ki je izvršil graditev, trenutno stanje kakovosti graditve in povezavo do dnevnika graditve.
- *Build Steps (Koraki graditve)* – vsebuje seznam (ki je dinamičen, če je graditev v teku), na katerem so datum in časovni žig za posamezno stopnjo graditve.
- *Result Details (Podrobnosti rezultatov)* – vsebuje napake in opozorila, ki jih je ustvarila graditev, rezultate preizkusov, ki so bili zagnani z graditvijo, in rezultate obsega delovanja kode.
- *Associated Chengesets (Povezani nizi sprememb)* – vsebuje seznam s hiperpovezavami za nize sprememb, ki so bili povezani z graditvijo.
- *Associated Work Items (Povezane delovne naloge)* – vsebuje seznam s hiperpovezavami za delovne naloge, ki so bili povezane z graditvijo. [3]



Slika 46: Poročilo o graditvi

7. Poročanje

To poglavje opisuje arhitekturo poročanja za TFS in pogosta poročila, ki jih lahko uporabljamo pri novih skupinskih projektih. Povezuje tudi pogoste scenarije poročanja s poročili, ki so na voljo v TFS-ju, in opisuje pogoste razloge za prilagajanje obstoječih ali ustvarjanje novih poročil. Te informacije lahko uporabimo za analizo napredka postopka, ustreznosti projekta in učinkovitosti skupine za razvoj in preizkušanje.
Poročanje TFS za ustvarjanje, upravljanje in zagon poročil uporablja storitve za poročanje Microsoft SQL Server™ 2005 Reporting Services. Posamezna predloga postopka vsebuje niz vnaprej določenih poročil, ki se uvedejo v mapo s poročili projekta, ko se projekt ustvari. S storitvami za poročanje lahko tudi dopolnimo ta poročila in ustvarimo sporočila po meri za svoj projekt. Dodamo lahko nova poročila v obstoječo predlogo postopka, tako da so na voljo za druge skupinske projekte.

Scenariji in rešitve
Poročila so osnovni način, s katerim vodje projektov in vodje skupine ostanejo na tekočem pri projektih v teku. Ko ustvarimo nov skupinski projekt, se na osnovi predloge postopka, ki jo izberemo, ustvari niz poročil. Ta poročila so na voljo na portalu projekta Microsoft Office SharePoint® ali v vozlišču *Reports (Poročila)* v Team Explorerju v orodju Visual Studio.
To so pogosta vprašanja, na katera lahko odgovorimo s poročili TFS:

- Kdaj bo moja aplikacija pripravljena za pošiljanje/dobavo?
- Ali delo poteka v skladu z načrtom?
- Kakšna je kakovost graditve?
- Kakšno je stanje razvoja glede na določene scenarije?
- Kako hitro bo razvojno delo končano?
- Ali se napake popravljajo?
- Ali se napake znova pojavljajo? [2]

7.1 Poročila TFS

Obe predlogi postopka, tako Microsoft Solution Framework (MSF) for Agile Software Development (MSF Agile) kot MSF for CMMI® Process Improvement (MSF CMMI), privzeto vsebujeta niz poročil:

a) ***Napake.*** S poročili o napakah v predlogah postopka vidimo, katere napake so ustvarjene in popravljene, ter prepoznamo trende. Na voljo so naslednja poročila o napakah:

- **Napake po prioriteti.** Ali so bile najdene prave napake? To poročilo prikaže število najdenih napak z visoko prioriteto v primerjavi z napakami z nizko prioriteto. Poročilo je na voljo v obeh predlogah postopka.
- **Stopnje napak.** Kako učinkovito se napake iščejo, popravljajo in zapirajo**?** To poročilo prikaže trende za nove napake, zaostanke pri napakah in rešitve za napake. Poročilo je na voljo v obeh predlogah postopka. [2]

b) ***Upravljanje izdaje***
Poročila za upravljanje izdaje omogočajo, da presodimo, kdaj bo programska oprema pripravljena za izdajo. Na voljo so naslednja poročila za upravljanje izdaje:

- **Dejanska kakovost v primerjavi z načrtovano hitrostjo.** Koliko scenarijev se lahko dokonča, preden je kakovost nesprejemljiva? To poročilo predstavlja razmerje za posamezno ponovitev, od ocenjene velikosti do splošne kakovosti. Poročilo je na voljo v obeh predlogah postopka.
- **Graditve.** Kakšna je kakovost graditve? To poročilo ponuja seznam razpoložljivih graditev, vključno s kakovostjo graditve in drugimi podrobnimi informacijami. Poročilo je na voljo v predlogi MSF for CMMI Process Improvement.
- **Indikatorji kakovosti.** Kakšna je kakovost programske opreme? To poročilo zbere rezultate preizkusov, napake in obseg delovanja kode v eno samo poročilo za sledenje ustreznosti projekta. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI.
- **Hitrost.** Kako hitro skupina dokončuje delo? To poročilo prikazuje, kako hitro skupina dokončuje načrtovano delo, prikazane pa so tudi stopnje sprememb iz dneva v dan. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI.
- **Podrobnosti scenarija.** Kakšne scenarije ustvarjamo za aplikacijo? To poročilo ponuja informacije za posamezen scenarij, vključno s stanjem dokončanja, tveganji in napredkom preizkušanja. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI. [2]

c) Preizkušanje

S poročili preizkušanja lahko nadzorujemo učinkovitost in napredek preizkušanja. Na voljo so naslednje poročila:

- **Regresija.** Kateri preizkusi so bili uspešno opravljeni, vendar pa jih zdaj ni mogoče opraviti? To poročilo prikazuje seznam vseh preizkusov, ki so bili prej uspešni, zdaj pa so neuspešni. Poročilo je na voljo v predlogi MSF CMMI.
- **Zgodovina preizkusa zahtev.** Kako dobro preizkušeni so moji scenariji in zahteve? To poročilo prikazuje napredek preizkušanja glede na določene scenarije in zahteve. Poročilo je na voljo v predlogi MSF CMMI.
- **Neuspešen preizkus brez aktivne napake.** Ali za sledenje vsake znane nepravilnosti obstaja delovna naloga tipa *napaka*? To poročilo prikazuje vse neuspešne preizkuse, ki niso povezani z odprto napako. Poročilo je na voljo v predlogi MSF CMMI.
- **Uspešen preizkus z odprto napako.** Ali je ta seznam posodobljen in se ujema s kakovostjo aplikacije? To poročilo prikazuje zastarele napake, pri katerih so preizkusi zdaj uspešno opravljeni. Poročilo je na voljo v predlogi MSF CMMI.
- **Povzetek preizkusa obremenitve.** Kakšni so rezultati preizkušanja obremenitve za učinkovitost delovanja aplikacije? To poročilo prikazuje rezultate preizkusa obremenitve aplikacije. Poročilo je na voljo v predlogi MSF CMMI. [2]

d) Delovne naloge

Poročila o delovnih nalogah omogočajo dostop do trenutnega stanja projekta in napredka trenutnega projekta. Na voljo so naslednja poročila o delovnih nalogah:

- **Odprte napake in trend blokiranih delovnih nalog.** Koliko odprtih napak ostaja? To poročilo prikazuje preostale odprte napake kot tudi trende za njihovo odpravljanje. To poročilo je na voljo v MSF CMMI.
- **Ponovne aktivacije.** Koliko delovnih nalog je ponovno aktiviranih? To poročilo prikazuje delovne naloge, ki so bile rešene ali prehitro zaprte. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI.

- **Povezane delovne naloge.** Katere delovne naloge so odvisne od drugih delovnih nalog? To poročilo prikazuje seznam delovnih nalog, ki so povezane z drugimi delovnimi nalogami, tako da lahko sledimo odvisnostim. Poročilo je na voljo v predlogi MSF CMMI.
- **Preostalo delo.** Koliko dela je ostalo in kdaj bo dokončano? To poročilo prikazuje preostalo, rešeno in zaprto delo. Projektiranje trendov preostalega dela naprej omogoča, da predvidimo točko, ko bo delo dokončano. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI.
- **Triaža.** Za katere delovne naloge je potrebna triaža? To poročilo prikazuje delovne naloge, ki so še vedno v predlaganem stanju. To poročilo je na voljo v MSF CMMI.
- **Nenačrtovano delo.** Koliko je nenačrtovanega dela? To poročilo s pomočjo razpredelnic prikazuje celotno delo v primerjavi s preostalim delom in razlikuje načrtovano od nenačrtovanega dela. Poročilo je na voljo v predlogah MSF Agile in MSF CMMI.
- **Delovne naloge.** Katere delovne naloge so aktivne? To poročilo navaja vse aktivne delovne naloge. To poročilo je na voljo v MSF CMMI.
- **Delovne naloge glede na lastnika.** Koliko dela je dodeljenega posameznemu članu skupine? To poročilo prikazuje delovne naloge na člana skupine. Poročilo je na voljo v predlogi MSF CMMI.
- **Delovne naloge glede na stanje.** Koliko je aktivnih, rešenih in zaprtih delovnih nalog? To poročilo navaja delovne naloge, ki so razvrščene po stanju. Poročilo je na voljo v predlogi MSF CMMI. [2]

7.2 Prilagajanje poročil

Včasih potrebujemo poročilo, ki ne obstaja v predlogah postopka MSF. Poročilo lahko prilagodimo na enega od treh načinov:

- **Uporabimo filter v obstoječem poročilu.** V poročilih so na voljo parametri, s katerimi lahko filtriramo poročilo. Na voljo so na primer filtri za podatke, področja, ponovitve in prioritete. Te filtre uporabimo za pregled podniza podatkov, ki je na voljo v poročili. Ti filtri so začasni in ne veljajo več, ko se premaknemo iz poročila.
- **Prilagodimo obstoječe poročilo.** Če je želeno poročilo podobno obstoječemu poročilu, je pogosto najlažje, da kopiramo obstoječe poročilo in ga nato spremenimo.
- **Ustvarimo novo poročilo.** Ustvarimo lahko povsem novo poročilo.

Če spremenimo obstoječe poročilo ali ustvarimo povsem novega, ga lahko objavimo v strežniku za poročanje, tako da je na voljo tudi drugim članom ekipe. Če želimo spremeniti obstoječe poročilo ali ustvariti novega, lahko uporabimo enega od naslednjih načinov:

- S programom Microsoft Office Excel® ustvarimo osrednje tabele iz podatkov v bazah podatkov za poročanje.
- V orodju Visual Studiu ustvarimo nov projekt v strežniku za poročanje in nato ustvarimo nova poročila ali uvozimo obstoječa. [2]

7.3 Dovoljenja za poročila

Nadzorujemo lahko, kateri uporabniki in skupine lahko berejo ali spreminjajo poročila skupine, ki so navedena v vozlišču Poročila v Team Explorerju in na spletnem mestu projekta

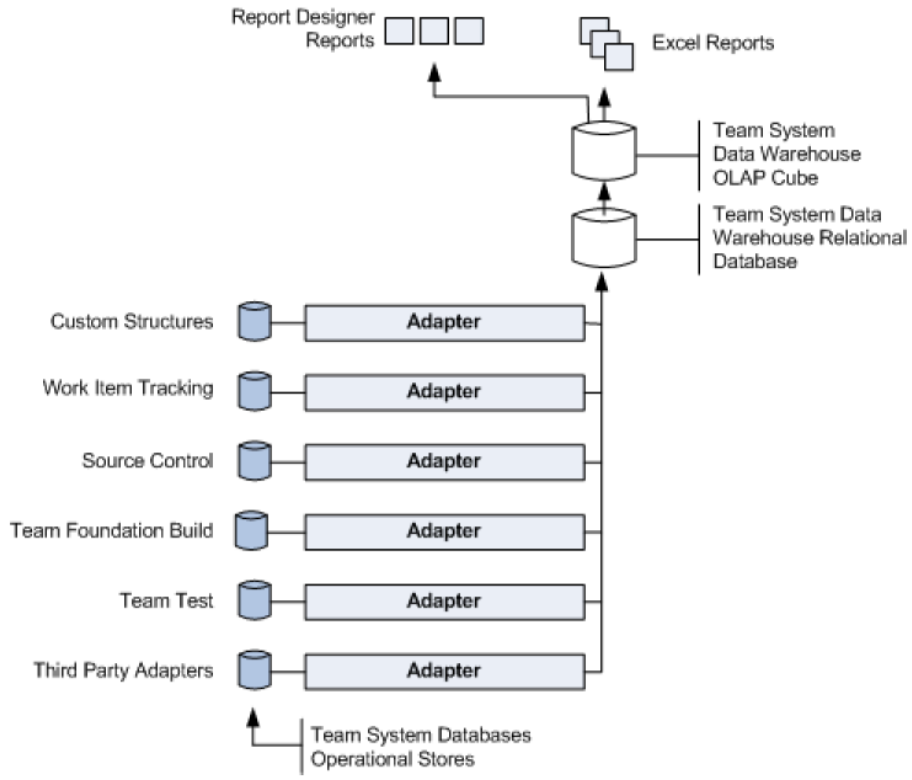
skupine v storitvah za poročanje SQL Server Reporting Services. Dovoljenja, ki so nastavljena za posamezno poročilo, so konfigurirana v storitvah za poročanje SQL Server Reporting Services. [6]

Poznamo tri skupine dovoljenj:

a) **Dovoljenja sodelavca**
   Sodelavec pri projektu Team Foundation običajno prispeva k projektu, vendar nima skrbniških pravic za projekt. Sodelavec pri projektu Team Foundation lahko bere in piše delovne naloge, dostopa do portala skupinskega projekta in vodenja procesa za skupinski projekt. Uporabnik, ki ustvari nov skupinski projekt, je samodejno dodan kot skrbnik projekta. Če želite dodeliti pravice drugim uporabnikom, jih morate dodati kot sodelavce pri projektu. [7]
b) **Dovoljenja vodje projekta**
   Vodja projekta v strežniku Team Foundation Server je običajno zadolžen za skupinski projekt in lahko vzdržuje zbirko podatkov za delovne naloge skupinskega projekta in portal skupinskega projekta ter skrbi za dovoljenja in varnost skupinskega projekta. Samo člani skupine *Team Foundation Administrators (Skrbniki TF)* lahko ustvarijo nov skupinski projekt. V večini primerov mora skrbnik TF v projekt dodati uporabnika ali skupino, če želi biti vodja skupinskega projekta. [8]
c) **Dovoljenja skrbnika**
   Člani skupine *TF Administrators* imajo najvišji niz dovoljenj od vseh uporabnikov strežnika Team Foundation Server. Skrbniki vzdržujejo TFS ter skrbijo za dovoljenja in varnost za druge vloge. V večini organizacij, ki uporabljajo TFS, je isti posameznik odgovoren za ustvarjanje projektov, njihovo upravljanje in prilagajanje vodenja procesa. [9]

7.4 Fizična arhitektura

TFS je vgrajen v SQL Server 2005 in uporablja storitve za analiziranje SQL Server Analysis Services za združevanje podatkov in zagon poročil. S programom Microsoft Excel ali Visual Studio 2005 Report Designer lahko ustvarimo nova poročila. Poročila gostujejo v storitvah za poročanje SQL Server 2005 Reporting Services in si jih lahko ogledamo na spletnem mestu strežnika za poročila, portalu skupinskega projekta SharePoint ali v vozlišču Poročila v Team Explorerju.



Slika 47: Fizična arhitektura poročanja

Posamezna komponenta TFS vsebuje svoj niz transakcijskih zbirk podatkov. To vključuje delovne naloge, nadzor nad viri, preizkuse, napake in Team Build. Ti podatki so združeni v relacijsko zbirko podatkov. Podatki so nato postavljeni v kocko OLAP (Online Analytical Processing) za podporo poročanju na osnovi trenda in bolj napredno analizo podatkov.
Relacijska zbirka podatkov TfsWarehouse je skladišče podatkov, oblikovano bolj z namenom uporabe za poizvedovanje po podatkih kot za njihove prenose. Podatki se prenesejo iz različnih zbirk podatkov TFS, ki so optimizirane za obdelavo transakcij, v to skladišče za namene poročanja. Skladišče ni glavna shramba za poročanje, vendar ga lahko uporabite za ustvarjanje poročil. Vir podatkov TfsReportDS kaže na relacijsko zbirko podatkov. Kocka Team System Data Warehouse OLAP Cube je zbirka podatkov, do katere dostopate prek storitev za analiziranje SQL Server Analysis Services. Kocka je uporabna pri poročilih, ki omogočajo analizo podatkov za trende, kot je »Koliko napak je bilo zaprtih ta mesec v primerjavi s prejšnjim?« Vir podatkov TfsOlapReportDS kaže na kocko OLAP skladišča podatkov za Team System v zbirki podatkov za analizo. [2]

Komponente sistema za poročanje

Sistem za poročanje je sestavljen iz naslednjih strežniških in odjemalskih komponent:

a) Strežniške komponente

Strežniške komponente vključujejo:

- *Zbirke podatkov strežnika za poročanje.* Te zbirke podatkov vsebujejo definicije poročil, zgodovinska poročila in konfiguracijske podatke.
- *Spletna storitev strežnika za poročanje*. Ta spletna storitev omogoča programski dostop do strežnika za poročanje.
- *Spletno mesto upravitelja poročila.* To mesto omogoča uporabniški dostop do strežnika za poročanje iz spletnega brskalnika.

- *Storitev Windows.* Ta storitev omogoča razporejanje in dostavo posnetkov poročila. [2]

b) Odjemalske komponente

Odjemalske komponente vključujejo:

- *Brskalnik***.** Ta komponenta omogoča dostop do spletnega mesta upravitelja poročil.
- *Team Explorer***.** Ta komponenta omogoča dostop do poročil iz programa Visual Studio. [2]

c) Orodja za razvoj poročil

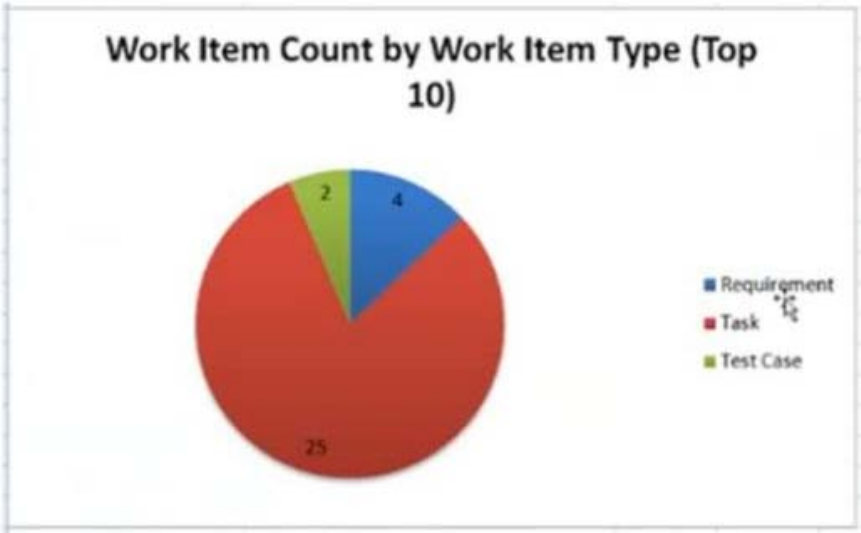
Orodja za razvoj vključujejo:

- *BIDS (Business Intelligence Designer Studio).* Ta komponenta omogoča razvijalcem, da oblikujejo in uvedejo poročila iz programa Visual Studio 2005.
- *Excel***.** V programu Excel lahko ustvarimo začasne poizvedbe in poročila ter si tako pomagamo pri upravljanju skupinskega projekta.

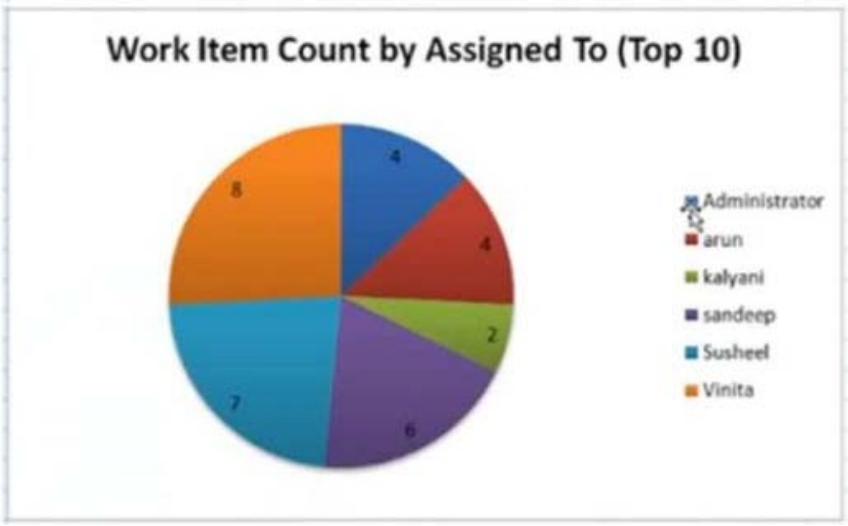
  - **Ustvarjanje poročila v programu Microsoft Excel**

    Microsoft Excel lahko uporabimo za ustvarjanje poročila na osnovi podatkov iz skladišča podatkov Team Foundation. Poročilo ustvarimo tako, da ustvarimo vrtilno tabelo v delovnem zvezku programa Microsoft Excel in povežemo vrtilno tabelo s skladiščem podatkov v strežniku Microsoft SQL Server. Ko ustvarimo poročilo vrtilne tabele, določimo, katera polja iz vira podatkov nas zanimajo, kako želimo razvrstiti tabelo in katere izračune želimo izvesti.

    Ko ustvarimo poročilo vrtilne tabele, jo lahko znova razporedimo in si ogledamo podatke iz različnih perspektiv. Ta možnost vrtenja dimenzij tabele (na primer za spreminjanje naslovov stolpcev v položaje vrstic) daje vrtilni tabeli njeno ime in nenavadno moč analiziranja. [10]



Slika 48: Primer poročila (število delovnih nalog različnih vrst) v Microsoft Excelu [12]



Slika 49: Primer poročila (število delovnih nalog glede na uporabnike) v Microsoft Excelu [12]

- **Urejanje poročila v programu Microsoft Excel**
  Ker se pogoji/okoliščine skupinskega projekta spreminjajo, je treba spremeniti poročila programa Microsoft Excel, ki jih uporabljamo za upravljanje projekta. Poročila moramo spremeniti tudi, ko postanejo na voljo nova polja v skladišču podatkov ali če želimo izbrati različne zapise za poročilo.
  Vrtilno tabelo ali vrtilni grafikon lahko posodobimo z uporabo novih podatkov, ki se prenesejo v originalno specifikacijo izvornih podatkov, ko posodobimo poročilo. Osveževanje zažene temeljno poizvedbo za pridobivanje novih ali spremenjenih podatkov. [11]

- *Report Designer* je nabor grafičnih orodij in oken, ki se izvajajo v programu Visual Studio 2005. Report Designer ponuja grafični vmesnik, v katerem lahko določimo vire podatkov in informacije poizvedb, umestimo področja in polja podatkov v poročilo, natančneje določimo postavitev poročila in nastavimo interaktivne funkcije.

  - **Ustvarjanje poročil v orodju Report Designer**
    Napredek skupine lažje spremljamo, če ustvarimo poročila, ki združujejo podatke iz strežnika Team Foundation Server v grafikone in tabele. Ustvarimo lahko na primer poročilo, ki prikazuje, koliko aktivnih delovnih nalog je dodeljenih posamezni osebi v skupini. Za takšno poročilo uporabimo orodje Report Designer v strežniku SQL Server 2005 in zbirko podatkov storitev za analiziranje v skladišču podatkov za Team System.
    Ko ustvarimo prvo poročilo, ga lahko spremenimo, tako da eksperimentiramo z različnimi merami in postavitvami. Spremenimo lahko na primer grafikon – iz enostavnega grafikona s stolpci v naložen palični grafikon.

  - **Urejanje poročil v orodju Report Designer**
    Ko ustvarimo in uporabimo poročilo, ga moramo morda spremeniti, ker se spremenijo zahteve skupinskega projekta. [13]

8. Primerjava strežnika Team Foundation Server z odprtokodnim sistemom

V tem poglavju se bom osredotočil na primerjavo strežnika Team Foundation Server in odprtokodnega sistema, ki je po funkcionalnostih najbolj podoben Team Foundation Server-ju in tudi najbolj konkurenčen. Odločil sem se za sistem Subversion (SVN), ki je tudi najbolj razširjen.

8.1 Predstavitev sistema Subversion

Subversion je brezplačen/odprtokodni sistem za nadzor nad viri. Subversion upravlja z datotekami, imeniki in spremembami v njih. To nam omogoča, da obnovimo starejše različice podatkov ali pregledamo zgodovino sprememb podatkov. Zaradi tega veliko ljudi misli, da je Subversion neke vrste »časovni stroj«.
Subversion lahko deluje po omrežjih, zato ga lahko uporabljajo ljudje na različnih računalnikih, poleg tega pa lahko več ljudi spreminja in upravlja isti niz podatkov s svojih lokacij, kar spodbuja sodelovanje. Napredek je hitrejši, če se spremembe izvajajo na več načinov. Ker je delo razdeljeno na različice, se kakovost ne zmanjša, če ostanemo brez enega načina. Če pride do napačne spremembe podatkov, lahko enostavno razveljavimo to spremembo.

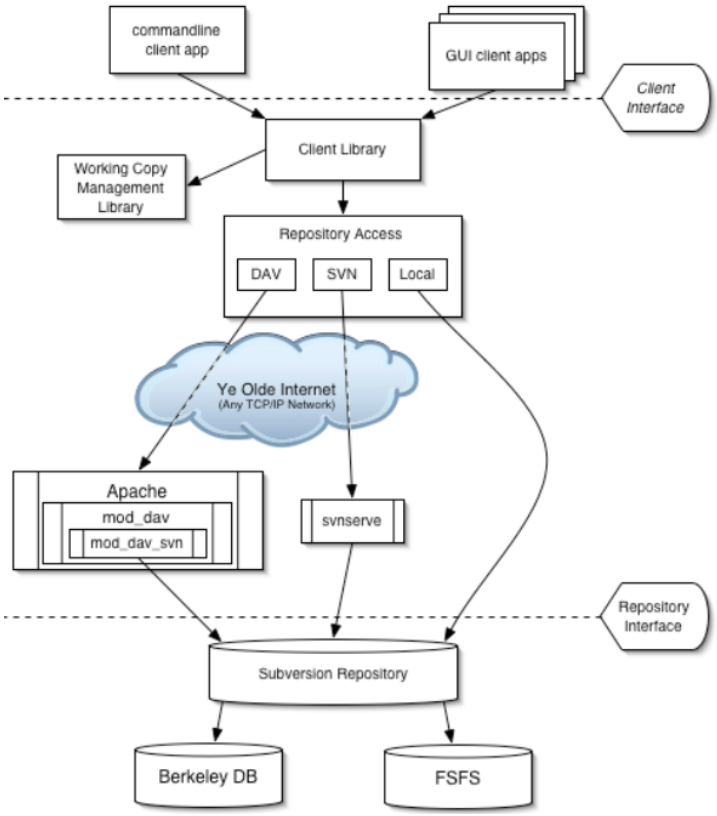
Ta sistem je posebej prilagojen za upravljanje dreves izvorne kode in vsebuje veliko funkcij, ki so značilne za razvoj programske opreme, na primer naravno razumevanje programskih jezikov in oskrba z orodji za grajenje programske opreme. Je splošen sistem, ki se lahko uporablja za upravljanje katere koli zbirke datotek. Te datoteke so lahko izvorna koda ali pa seznam trgovin z digitalnimi videi in podobno.

a) **Funkcije sistema Subversion**
   - **Imenik različic.** Subversion ima vdelan virtualni datotečni sistem različic, ki sledi spremembam celotnega imenika dreves v daljšem časovnem obdobju. Datoteke in imeniki imajo svojo različico.
   - **Pravilna zgodovina različice.** Subversion omogoča dodajanje, kopiranje in preimenovanje datotek in imenikov. Vsaka novo dodana datoteka se začne s svežo, čisto zgodovino.
   - **Atomska sprostitev.** Zbirka sprememb gre v skladišče v celoti ali pa sploh ne. To omogoča razvijalcem izgradnjo in objavo sprememb v enem kosu in preprečuje težave, ki lahko nastanejo, ko se samo en del sprememb pošlje v skladišče.
   - **Različice meta podatkov.** Vsaka datoteka v imeniku ima nabor lastnosti, ključev in vrednosti, povezanih z njimi. Ustvarimo in shranimo lahko vsak ključ/vrednost, ki jo želimo. Lastnosti imajo svoje različice, tako kot vsebine datotek.
   - **Izbira slojev omrežja.** Je preprost za uvedbo novih mehanizmov omrežja. Subversion lahko priključimo na strežnik Apache HTTP Server kot razširitveni modul. To daje sistemu Subversion veliko prednost pri stabilnosti in interoperabilnosti ter takojšen dostop do obstoječih funkcij, ki jih zagotavlja ta strežnik. Mogoč je tudi bolj enostaven proces strežnika Subversion. Ta strežnik komunicira prek lastnega protokola, s katerim je zelo enostavno zgraditi prehod čez SSH (Security Shell).
   - **Dosledna obdelava podatkov.** Subversion izraža razlike datotek z uporabo binarnega razlikovalnega mehanizma, ki deluje enako kot delujeta besedilna

(človeku berljivo) in binarna (človeku neberljivo) datoteka. Obe datoteki sta shranjeni v skladišču v enaki stisnjeni obliki, razlike pa se prenašajo v obeh smereh po omrežju.

- **Učinkovito vejanje in označevanje.** Cena vejanja in označevanja ni nujno sorazmerna z velikostjo projekta. Subversion ustvari veje in oznake z enostavnim kopiranjem projekta, uporabo mehanizma, ki je podoben trdi povezavi. Postopki tako vzamejo zelo malo časa.
- **Prilagodljivost.** Subversion nima skladišča zgodovine; implementiran je kot zbirka knjižnic C z natančno opredeljenim vmesnikom API (Application Programming Interface). Zaradi tega je Subversion zelo vzdržljiv in uporaben s strani ostalih aplikacij in jezikov.

b) Arhitektura sistema Subversion



Slika 50: Arhitektura sistema Subversion

Na eni strani je Subversion skladišče, ki hrani vse različice podatkov. Na drugi strani pa je program Subversion odjemalec, ki upravlja z lokalnimi odzivi deležev teh različic podatkov, ki jih imenujemo delovne kopije. Med tema dvema stranema potekajo poti skozi plasti *Dostopi do skladišča (Repository Acces)*. Nekatere od teh poti potekajo prek računalniških omrežij in prek omrežnih strežnikov, ki nato dostopajo do skladišča. Druge se v celoti izognejo omrežju in do skladišča dostopajo neposredno.

c) Komponente sistema Subversion

Ko je Subversion enkrat nameščen, ima veliko različnih delov. V nadaljevanju so opisani ti deli sistema Subversion.

- *Svn*. Ukazna vrstica odjemalca programa.
- *Svnversion*. Program za poročanje stanja delovne kopije.
- *Svnlook*. Orodje za neposredno pregledovanje skladišča.
- *Svnadmin*. Orodje za ustvarjanje ali popravljanje skladišča.
- *Svndumpfilter*. Program za razvrščanje tokov odlaganja v skladišče.
- *mod_dav_svn*. Vtični modul za strežnik Apache HTTP, ki omogoča drugim v omrežju dostop do skladišča.
- *Svnserve*. Navaden samostojni strežniški program. Drug način za omogočanje dostopa do skladišča.
- *Svnsync*. Program za postopno zrcaljenje enega skladišča v drugega prek omrežja.

8.5 Prednosti sistema Team Foundation Server

1) Odlaganje kode/pridobivanje kode

Je ena izmed zelo pomembnih stvari v strežniku TFS. Uporablja se za shranjevanje sprememb iz lokalnega razvojnega delovnega okolja v strežnik TFS, ne da bi te spremembe sprostili. Pozneje lahko uporabimo ukaz *Get (Pridobi)* in te spremembe posredujemo nazaj v lokalno delovno okolje.
Prednosti odlaganja kode so:

- Pregledovanje kode na drugih računalnikih.
- Preklapljanje med različnimi delovnimi nalogami. Začnemo na primer z eno delovno nalogo in moramo na polovici preklopiti na drugo. Za vse do sedaj narejene spremembe enostavno uporabimo odlaganje kode.
- Shranjevanje sprememb na strežnik, kar je lahko dodatna zaščita sprememb.

2) Spremljanje zgodovine spajanj datotek

TFS hrani zgodovino vseh spajanj datotek in omogoča njihovo iskanje.

3) Integracija delovnih nalog

Vsakemu izmed razvijalcev je lahko dodeljena delovna naloga. Razvijalec lahko tudi ustvari novo delovno nalogo, npr. hrošča, in sprosti točno določeno kodo, ki jo je uporabil, za popravilo tega hrošča. Ob sprostitvi se ustvari seznam sprememb, ki je povezan z delovno nalogo in ga najdemo na zavihku *Links (Povezave).* Ko pregledovalec pozneje pregleduje kodo, lahko z desno tipko miške klikne kodo in izbere *Version control* (*Nadzor nad viri) -> Annotate (Opomba).* Tako se pomakne do točno določene delovne naloge, ki je povezana s to vrstico kode. To je zelo koristno pri ugotavljanju avtorja ni časa nastanka te kode.

Včasih želimo pogledati nazaj in najti točno določeno delovno nalogo. TFS vsebuje vmesnik, ki omogoča pisanje poizvedb in iskanje točno določenih delovnih nalog.

4) Poročanje

Poročila so osnovni način, s katerim vodje projektov in vodje skupine ostanejo na tekočem pri projektih v teku. Poročanje nam pomaga pri odgovorih na različna vprašanja in sicer:

- Kdaj bo moja aplikacija pripravljena za pošiljanje/dobavo?
- Ali delo poteka v skladu z načrtom?

- Kakšna je kakovost graditve?
- Kakšno je stanje razvoja glede na določene scenarije?
- Kako hitro bo razvojno delo končano?
- Ali se napake popravljajo?
- Ali se napake znova pojavljajo?

5) Projektno upravljanje
Funkcije upravljanja projektov zagotavljajo naslednje prednosti:

- centralizirano upravljanje;
- večjo sledljivost;
- integrirano projektno načrtovanje in določanje termina;
- izboljšan kontrolnik procesov;
- izboljšana komunikacija skupine in povezanost med člani skupine;
- poročanje o natančnem napredku.

8.6 Prednosti sistema Subversion

1) Enostavnost uporabe
SVN je zelo preprost za uporabo, čeprav ponuja širok nabor funkcij.

2) Dostop do datotek
Vsak lahko uporabi datoteko, ki je samo za branje, in pogleda vsebino. To je velika prednost odprtokodnih razvojnih okolij.

3) Spajanje graditev
Subversion lahko primerja dve popolnoma različni graditvi in ustvari popravek za posodobitev ene različice v drugo. To zmore tudi TFS, vendar je to veliko lažje narediti v sistemu Subversion.

8.7 Primerjava skupnih lastnosti strežnika Team Foundation Server in sistema Subversion

1) Urejanje datotek z drugimi urejevalniki
Team Foundation Server: Datoteke na lokalnem disku so zaklenjene s strani operacijskega sistema. Ko odpremo datoteko z urejevalnikom datotek v imeniku vira, datoteke ne moremo spreminjati, dokler je ne rezerviramo v strežniku.

Subversion: Datoteke lahko urejamo in shranjujemo tudi izven razvojnega okolja.

2) Lahkotnost uporabe
Team Foundation Server: Za razvijalce, ki so že uporabljali Microsoft Visual SourceSafe (podobno kot sistem za nadzor nad viri v strežniku TFS) in Microsoftova orodja za razvoj, je uporaba strežnika TFS mnogo lažja kot za razvijalce, ki jim zgoraj naštete stvari niso znane.

Subversion: Veliko težje ga uporabljajo razvijalci, ki so že uporabljali Microsoftova orodja, in lažje tisti, ki ne poznajo strežnika TFS ali Subversion.

3) Integracija v razvojno okolje
Team Foundation Server: Najbolje se integrira in obnese v razvojnem okolju Microsoft Visual Studio in je najboljša izbira pri programiranju v programskih jezikih .NET. Ni najboljša izbira, če ne razvijamo v orodju Microsoft Visual Studiu.

Subversion: Integracija v Visual Studio je zadovoljiva, vendar ne tako dobra kot TFS. Subversion je boljša izbira, če ne programiramo v jezikih .NET, predvsem če ne uporabljamo orodja Microsoft Visual Studi. Bolj je primeren za odprtokodne programske jezike.

4) Področja in produktivnost
Team Foundation Server: Ni samo sistem za nadzor nad viri, ampak zajema tudi področja upravljanja projektov, upravljanja testnih skript, zaznavanja napak itd.

Subversion: Je samo sistem za nadzor nad viri.

5) Strežnik
Team Foundation Server: Zahteva spletni strežnik IIS (Internet Information Services).

Subversion: Je zasnovan za delovanje s strežnikom Apache. Lahko deluje tudi s storitvami IIS, vendar veliko rešitev še vedno zahteva namestitev strežnika Apache.

6) Programske zahteve
Team Foundation Server: Windows Server 2003 ali Windows Server 2008, IIS, Aktivni imenik, SQL Server.

Subversion: Apache strežnik, MySQL.

7) Cena
Team Foundation Server: Cena za TFS se giblje okoli 450 EUR.

Subversion: Je odprtokodni sistem in je brezplačen.

8) Delovanje brez internetne povezave
Team Foundation Server: Pri strežniku TFS nastane težava, ko je strežnik brez povezave in želimo spremeniti datoteko. TFS po približno 1 minuti javi napako in ne preide v stanje brez povezave. Projekt moramo zapreti in nato znova odpreti, če želimo, da TFS preide v stanje brez povezave.

Subversion: Uporabnikom omogoča delo, tudi ko nimajo vzpostavljene internetne povezave.

9) Preimenovanje datotek
Team Foundation Server: Podpira katero koli preimenovanje, tudi ciklično in odvisno.

Subversion: Nima vdelanega posebnega preimenovanja. Če torej nekdo preimenuje datoteko brez naše vednosti in nato izvedemo ukaz *svn update*, se datoteka ne bo preimenovala.

10) Prilagodljivost map/datotek
Team Foundation Server: Lažje se prilagodi večjim delovnim kopijam datotek in večjim velikostim skladišča.

Subversion: Težje se prilagodi večjim delovnim kopijam datotek in večjim velikostim skladišča.

11) Pridobivanje datotek iz strežnika
Team Foundation Server: Je hitrejši pri pridobivanju datotek iz strežnika, ker pozna različico datoteke, ki jo ima odjemalec.

Subversion: Je počasnejši pri pridobivanju datotek iz strežnika.

12) Namestitev
Team Foundation Server: Težja, časovno veliko bolj potratna, poleg tega pa zavzame več prostora.

Subversion: Enostavna, hitra in zavzame malo prostora.

8.8 Moje mnenje o strežniku Team Foundation Server in sistemu Subversion

Menim, da sta oba zelo zanesljiva. Če pogledamo njuno filozofijo, hoče TFS uvesti svoj način, ki naj ga uporabniki uporabijo pri razvijanju, pri sistemu SVN pa se uporabnik sam odloči, kako ga bo uporabljal, v katera orodja ga bo integriral itd. Težko je primerjati TFS in SVN kot celoto. Kar ponuja SVN, je le del tistega, kar ponuja TFS. Če izkoristimo vse tisto, kar ponuja TFS, je TFS zelo dobra rešitev.

Kdaj uporabiti Team Foundation Server:

- če razvijate v razvijalnem okolju .NET;
- če cena strežnika TFS ni težava ali če ste partner družbe Microsoft, kar omogoča cenejši nakup strežnika TFS;
- če ste veliko podjetje in razvijate zapletene, velike projekte;
- če poleg sistema za nadzor nad vidi potrebujete celoten sistem upravljanja razvoja, ki omogoča veliko dodatnih stvari, kot so delovne naloge za obvladovanje poročil o napakah s strani strank, sledenje delovnim nalogam, povezovanje spremenjenih datotek z delovnimi nalogami, spremljanje procesa razvoja projekta, upravljanje projektov, sledenje delovnim nalogam, analiziranje in poročanje, določanje načrtovanega časa, že izkoriščenega časa in preostalega časa itd.;
- če želite celotno delovno okolje na enem mestu, tako da se vam ni treba ukvarjati z implementacijo dodatnih orodij.

Kdaj uporabiti Subversion:

- če razvijate v odprtokodnem razvijalnem okolju;
- če si strežnika TFS ne morete privoščiti;
- če ste majhno podjetje in razvijate enostavne, majhne projekte;
- če želite samo sistem za nadzor nad viri;

- če imate čas za implementacijo celotnega delovnega okolja, ki poleg sistema SVN zajema tudi ostala orodja, kot so CruiseControl.Net, NUnit, NCover itd.

9. Primer uporabe strežnika Team Foundation Server

V tem poglavju želim prikazati nekaj najpomembnejših lastnosti strežnika Team Foundation Server, opisanih v prejšnjih poglavjih, od začetka do konca nastajanja aplikacije *Pregledovalnik dogodkov*.

9.1 Aplikacija *Pregledovalnik dogodkov*

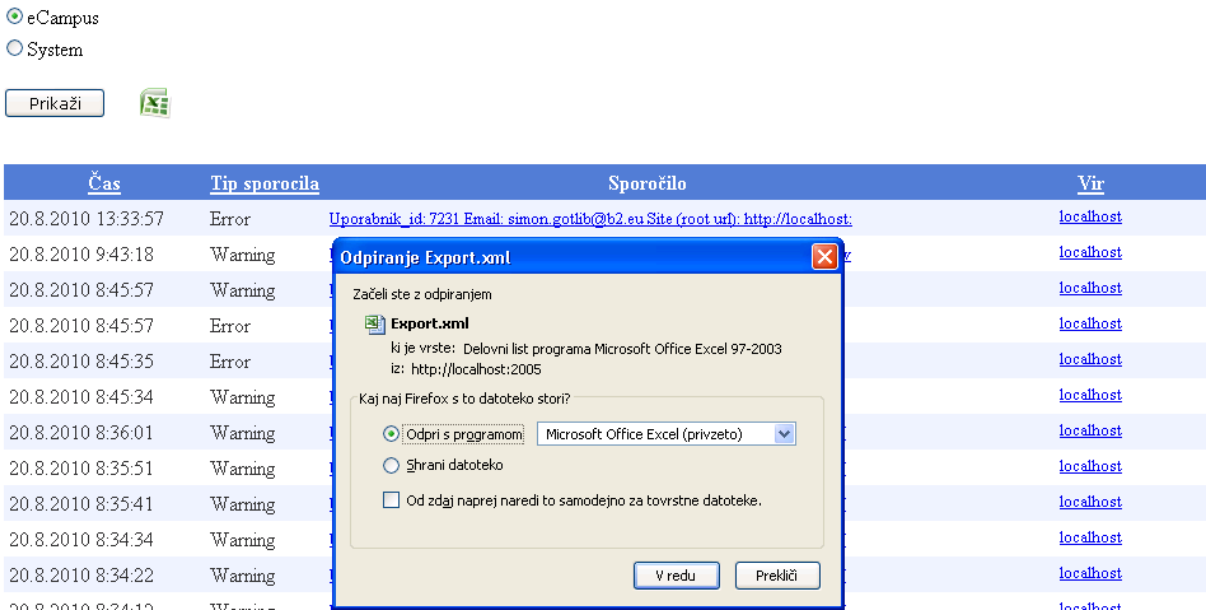
Za implementacijo programske opreme je bilo uporabljeno orodje Microsoft Visual Studio 2008, za programiranje programske kode pa jezik C# in tehnologija CCS za poenotenje sloga spletne aplikacije.

Spletne aplikacije se izvajajo na nekem strežniku, ki je običajno oddaljen računalnik. Spletna aplikacija omogoča zapis vsebine vseh ulovljenih napak v Microsoftovo komponento Event Viewer, ki omogoča uporabnikom pogled dnevnika dogodkov v lokalnem ali oddaljenem sistemu. Ker se aplikacija izvaja v oddaljenem računalniku, se je treba v primeru napake povezati z oddaljenim računalnikom in preveriti dogodek v komponenti Event Viewer, kjer je pojasnjen vzrok te napake. Zaradi lažje dostopnosti in ker ne uporabljamo vedno računalnika z možnostjo oddaljenega dostopa, sem se odločil narediti aplikacijo *Pregledovalnik dogodkov*, ki jo bo možno vključiti v že obstoječo spletno aplikacijo in tako kar na projektu preveriti vzrok napake.

Pregledovalnik dogodkov je spletna aplikacija, ki iz komponente Event Viewer glede na izbrano opcijo (področje dogodkov) prebere vse podatke in jih prikaže na strani. Prikaže le začetni del besedila dogodka, ob kliku na besedilo pa se le-to razširi. Vsak dogodek je sestavljen iz datuma, tipa sporočila, vsebine in vira. Tip sporočila sem omejil samo na napako in opozorilo. Omogočeno je tudi razvrščanje vsakega stolpca.

Slika 51: Prikaz podatkov v aplikaciji *Pregledovalnik dogodkov*

Te podatke lahko nato tudi izvozimo Microsoft Excel za lažji pregled.

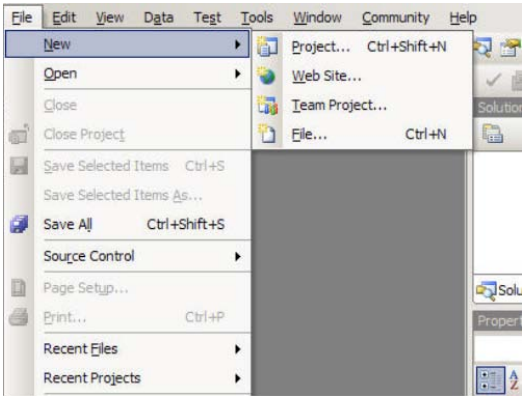


Slika 52: Izvoz podatkov v Microsoft Excel

9.2 Ustvarjanje novega skupinskega projekta

Ustvarjanje novega skupinskega projekta je dokaj enostavno, vendar moramo biti zelo pazljivi, saj ustvarjamo paradigmo za celotno organizacijo ali skupino. Če želimo ustvariti skupinski projekt in nadaljnje nastavitve, se moramo prepričati, da imamo ustrezne pravice. Pravice si zagotovimo, če smo člani skrbniške varnostne skupine *Team Foundation Administrators*.

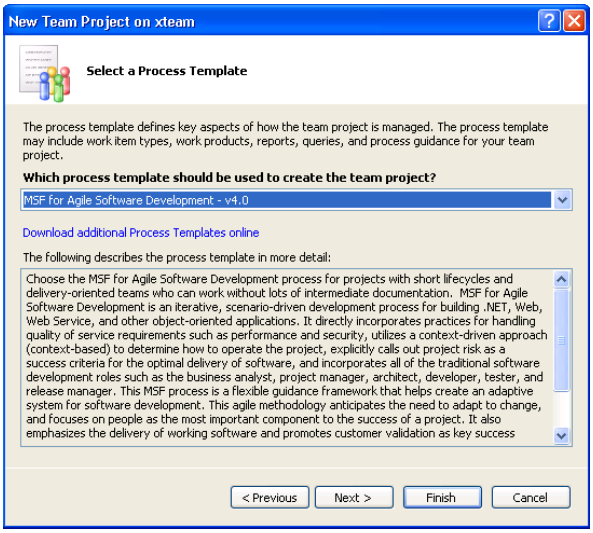
Team Foundation Server omogoča ustvarjanje skupinskega projekta s čarovnikom. Čarovnik zaženemo s klikom možnosti *File (Datoteka) -> New (Novo) -> Team Project (Skupinski projekt).*



Slika 53: Ustvarjanje novega skupinskega projekta

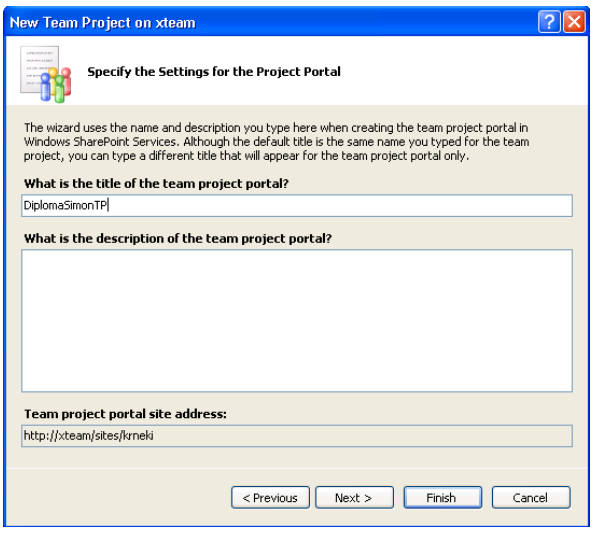
Odpre se potrditveno okno za vnos imena skupinskega projekta, v katerega vpišemo želeno ime. Sledi zelo pomemben korak. Izbrati moramo predlogo procesa, na podlagi katere nam bodo ponujene delovne naloge, poizvedbe, poročila itd. Na voljo imamo dve predlogi procesa, in sicer:
- MSF for Agile Software Development

- MSF for CMMI Process Improvement



Slika 54: Izbira predloge procesa

Ko izberemo predlogo procesa, vnesemo ime portala projekta. Portal projekta je skupina različnih komponent projekta. Obiščemo ga tako, da v spletni brskalnik vpišemo *http://<tfs app server>/sites/<project name>/default.aspx*. Portal se običajno uporablja za dostop do različnih dokumentov projekta, splošnih poročil o napredku itd.

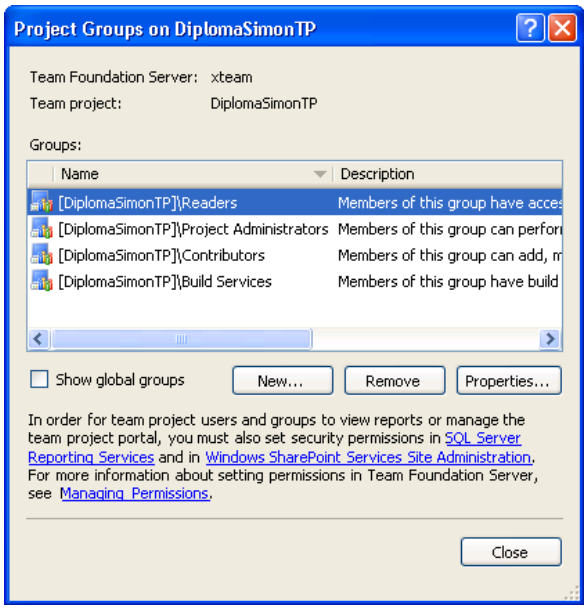


Slika 55: Vnos podatkov za portal projekta

Na koncu se prikaže potrditveno okno z vsemi podatki, ki smo jih izbrali med namestitvijo. Preverimo pravilnost podatkov in pritisnimo gumb *Finish (Dokončaj)*, da končamo ustvarjanje novega skupinskega projekta.

a) Dodajanje skupin in uporabnikov v skupinski projekt

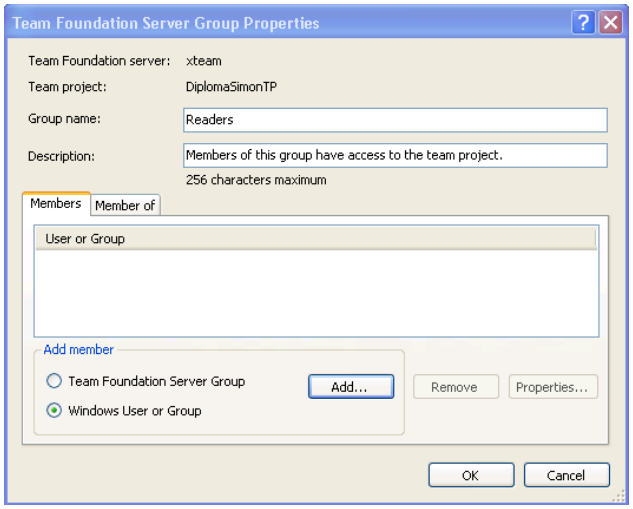
Kot člani skupine *Administrators* imamo dostop do varnosti skupin, ki nadzirajo dostop do naših projektov. Dostopamo lahko do nastavitev uporabnikov skupin s pomočjo okna *Team Explorer* ali prek menija *Team (Skupina)*. Prikaže se spodaj prikazano potrditveno okno.



Slika 56: Dodajanje skupin in uporabnikov v skupinski projekt

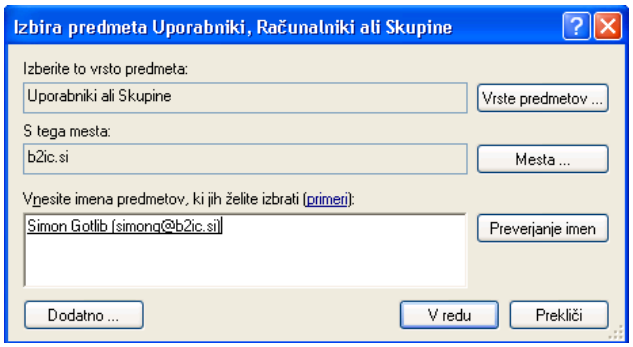
Pogovorno okno *Project Groups* (Skupine projekta) omogoča dodajanje in odstranjevanje skupin ter uporabnikov. Na voljo imamo štiri skupine. Če želimo dodati svojo skupino, kliknemo možnost *New (Novo)* ter v polje vpišemo ime skupine.

Druga možnost je dodajanje uporabnikov obstoječim skupinam. Uporabnike dodamo tako, da kliknemo eno od skupin, v katero želimo dodati uporabnike, in nato kliknemo *Properties (Lastnosti).* Odpre se novo pogovorno okno, v katerem moramo označiti možnost *Windows User or Group* (Uporabnik ali skupina sistema Windows) in klikniti *Add (Dodaj).*



Slika 57: Lastnosti skupin v skupinskem projektu

V novem pogovornem oknu izberemo želenega uporabnika in kliknemo *OK (V redu).* Tako dodamo novega uporabnika v skupini *Readers (Bralci).*

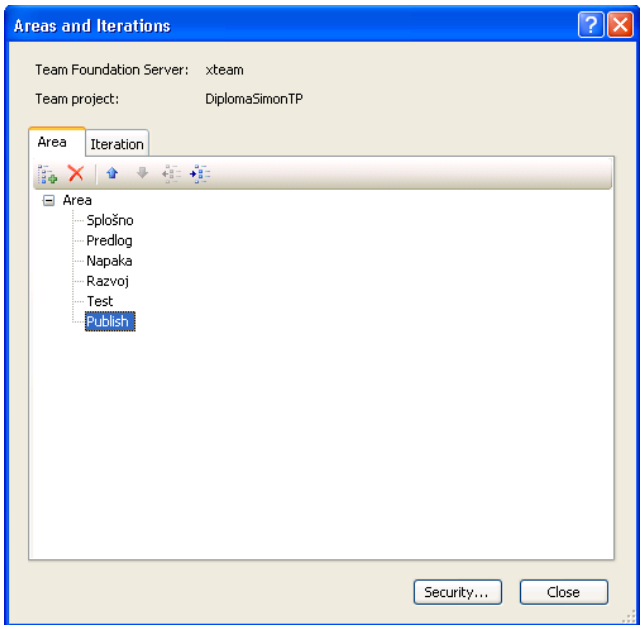


Slika 58: Dodajanje uporabnikov skupinam v skupinskem projektu

b) Področja in ponovitve skupinskega projekta

Naslednji korak je urejanje razvrstitve projektne skupine. Razvrstitve se uporabljajo za kategorizacijo delovnih predmetov na projektna področja in ponovitve. Struktura projekta (področja) je hierarhija vozlišč, ki predstavljajo področja funkcij ali skupine. Ponovitev je obdobje razvojnega prizadevanja in je namenjena doseganju posebnih mejnikov.

Strukturo projekta (področja) ustvarimo tako, da kliknemo *Team Project (Skupinski projekt) -> Team (Skupina) -> Areas and Iterations (Področja in ponovitve)* in nato zavihek *Areas (Področja)*. Rezultat je pogovorno okno, ki ga prikazuje spodnja slika. Nova področja ustvarimo tako, da kliknemo eno izmed vozlišč in kliknemo znak »plus« ter ga poljubno poimenujemo. Izbrišemo ga s klikom na »križec«.



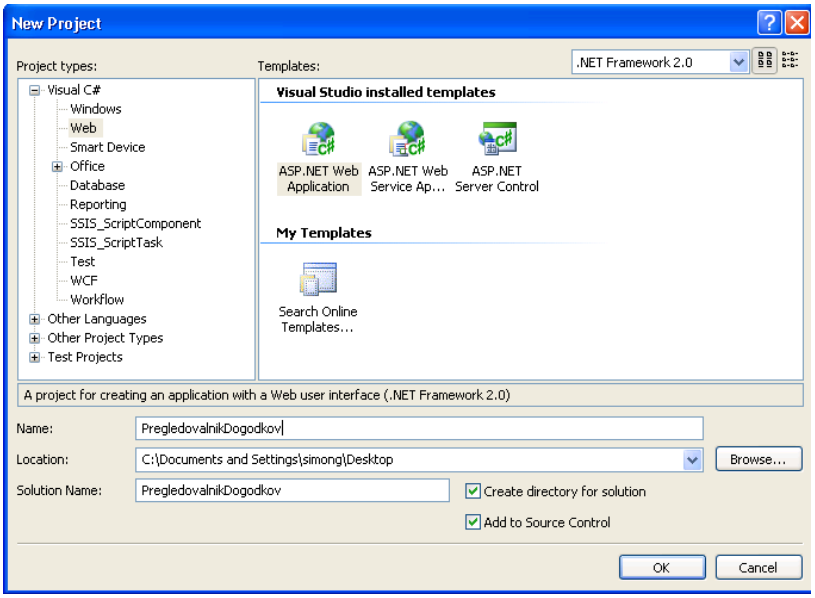
Slika 59: Upravljanje s področji in ponovitvami

Ponovitve dodajamo v istem pogovornem oknu, le da kliknemo zavihek *Iteration (Ponovitve)*. Postopek dodajanja/odstranjevanja pa je isti kot pri področjih.

9.3 Ustvarjanje novega projekta

Če želimo ustvariti novi projekt, moramo izbrati programski jezik, v katerem želimo ustvarjati ta projekt. Najpogostejša izbira sta programska jezika *Visual Basic* in *C#*. V našem primeru bomo izbrali C#, ki je razdeljen na več kategorij, med katerimi sta najbolj pogosto uporabljeni *Windows* in *Web (Splet)*. Izbrali bomo *ASP.NET Web Application (Spletna aplikacija ASP.NET)* iz kategorije *Web (Splet)*.

Kliknemo možnost *File (Datoteka) -> New (Novo) -> Project (Projekt)*, zavihek *Web (Splet)* ter izberemo *ASP.NET Web Application (Spletna aplikacija ASP.NET)*. V polje *Name (Ime)* vnesemo ime aplikacije, v polje *Location (Lokacija)* vnesemo pot, kamor želimo lokalno shraniti aplikacijo, ter označimo možnost *Add to Source Control (Dodaj v sistem za nadzor nad viri)*.



Slika 60: Ustvarjanje nove spletne aplikacije

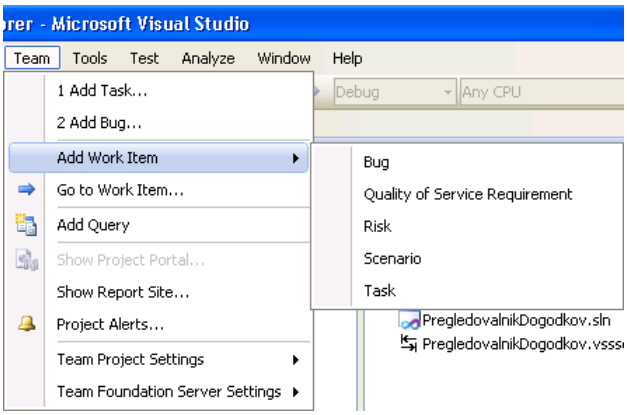
9.4 Delovne naloge

Eno najpomembnejših področij TFS so delovne naloge. Delovne naloge so opredelitev dela za razvojni projekt in so odgovorne za evidentiranje dela, omogočajo dodeljevanje dela, sledenje delu, povezovanje z drugimi delovnimi nalogami, poročila itd. Vsak del dela na projektu je delovna naloga. Pomembno pravilo je, da sledimo tem nalogam, dokler niso dokončane.

9.4.1 Ustvarjanje delovnih nalog

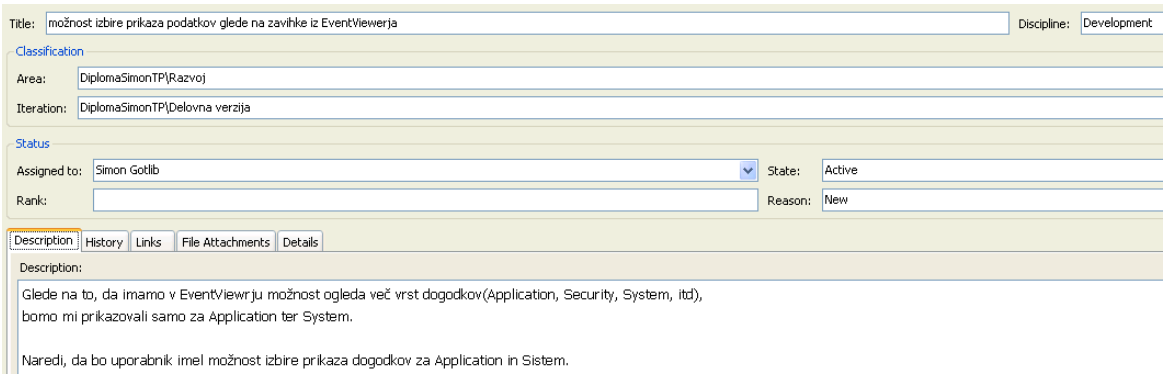
Na voljo imamo več vrst delovnih nalog. Te so odvisne od izbire predloge procesa. Pri ustvarjanju skupinskega projekta izberemo predlogo MSF for Agile Software Development in zato imamo na voljo naslednje tipe delovnih nalog: *Scenario (Scenarij)*, *Bug (Hrošč)*, *QoS (Kakovost storitve)*, *Task (Opravilo)* in *Risk (Tveganje)*. Ustvarili bomo delovno nalogo vrste *Task (Opravilo)*, ker gre za izvršitev nekega dela na projektu.

Kliknemo *Team (Skupina) -> Add Work Item (Dodaj delovno nalogo)* in izberemo *Task (Opravilo)*.



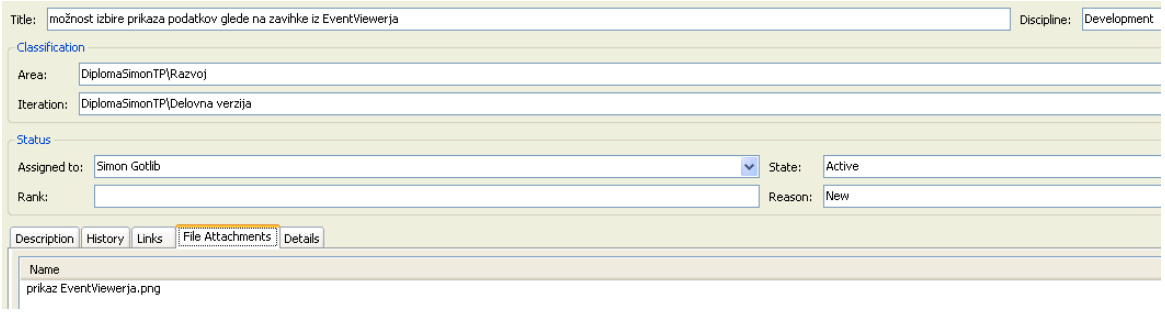
Slika 61: Ustvarjanje delovne naloge

Odpre se obrazec delovne naloge. Na obrazcu so polja za vnos podatkov, med katerimi so najpomembnejši *Title (Naslov)*, *Discipline (Stroko)*, *Area (Področje)*, *Iteration (Ponovitev)*, *Assigned to (Dodeljeno)*, *State (Stanje)* ter določeni zavihki. Osredotočili se bomo na zavihke *Description (Opis)*, *File Attachments (Datotečne priloge)* in *Details (Podrobnosti)*, ki so najpomembnejši za ustvarjanje delovne naloge. Na zavihku *Description (Opis)* vnesemo opis celotnega poteka delovne naloge.



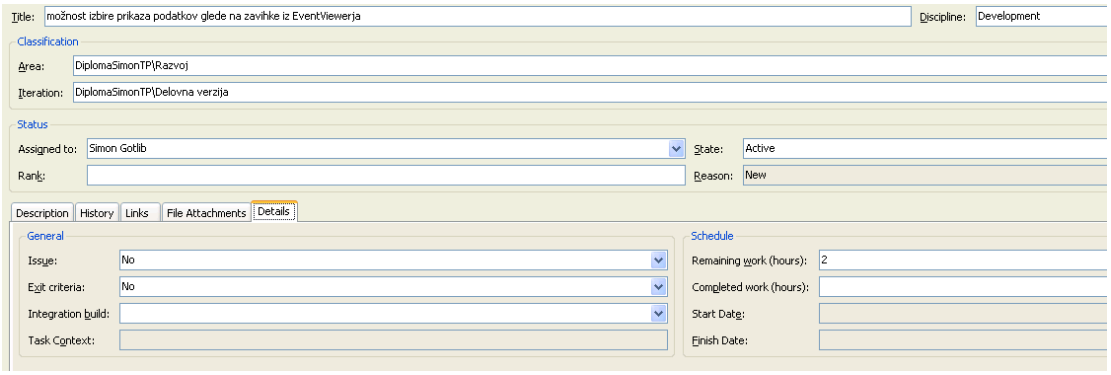
Slika 62: Zavihek *Description (Opis)* pri ustvarjanju delovne naloge

Zavihek *File Attachments (Datotečne priloge)* se uporablja za dodajanje datotek, ki bodo uporabniku, kateremu bo dodeljena delovna naloga, pomagale pri boljšem razumevanju delovne naloge.



Slika 63: Zavihek *File Attachments (Datotečne priloge)* pri ustvarjanju delovne naloge

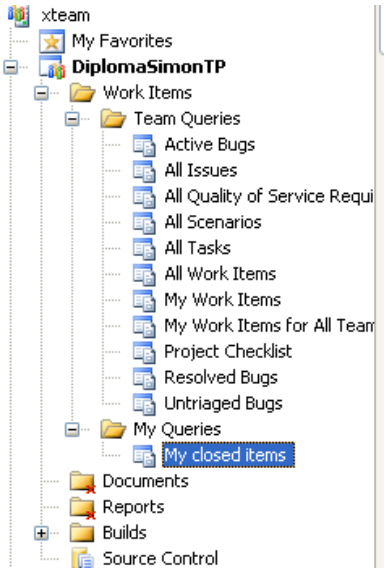
Na zavihku *Details (Podrobnosti)* vnesemo podatke za polja *Remaining work – hours (Preostalo delo - ure).*



Slika 64: Zavihek *Details (Podrobnosti)* pri ustvarjanju delovne naloge

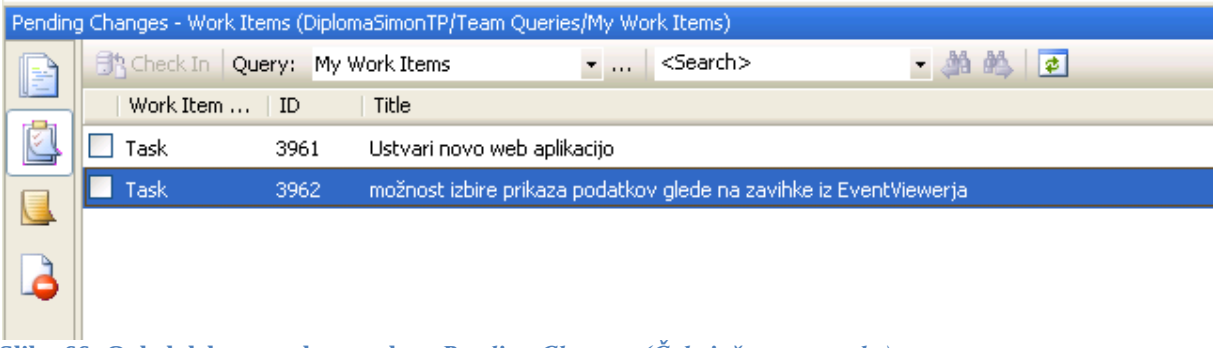
9.4.2 Ogled delovnih nalog

Delovne naloge si lahko ogledamo na tri načine. Prvi način je, da odpremo Team Explorer in kliknemo vozlišče *Work Items (Delovne naloge)*, v katerem sta mapi Team *Queries (Skupinske poizvedbe)* in *My Queries (Moje poizvedbe)*, kjer lahko dostopamo do katere koli delovne naloge.



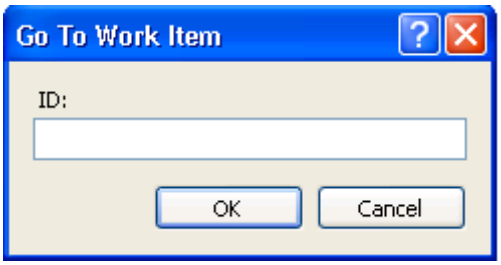
Slika 65: Ogled delovne naloge v oknu Team Explorer

Drugi način je ogled delovnih nalog v oknu *Pending changes (Čakajoče spremembe)*, kjer v polju *Query (Poizvedba)* izberemo želene delovne naloge.



Slika 66: Ogled delovne naloge v oknu *Pending Changes (Čakajoče spremembe)*

Tretji način je primeren, če poznamo ID delovne naloge. Kliknemo *Team (Skupina)-> Go To Work Item (Pojdi na delovno nalogo)*. Odpre se pogovorno okno, v katerega vnesemo ID delovne naloge.



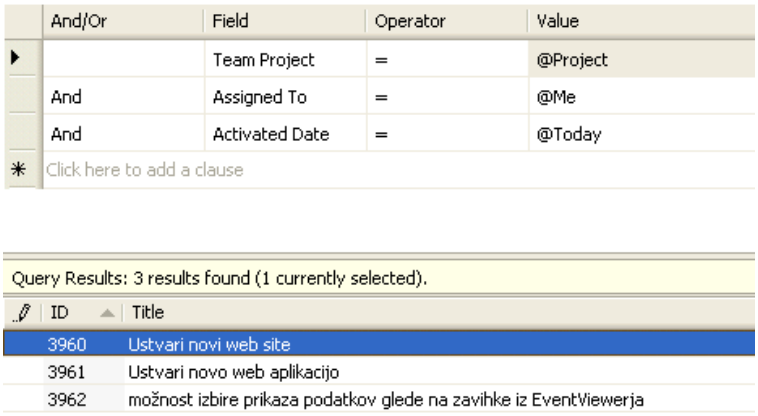
Slika 67:Ogled delovne naloge s pomočjo okna *Go To Work Item (Pojdi na delovno nalogo)*

9.4.3 Iskanje delovnih nalog

Iskanje delovnih nalog je odvisno od informacije, ki jo potrebujemo. Iščemo jih lahko na dva načina. Prvi je, da odpremo Team Explorer ter kliknemo vozlišče *Work Items (Delovne naloge).* V njem sta mapi *Team Queries (Skupinske poizvedbe)* in *My Queries (Moje poizvedbe),* kjer so že shranjene poizvedbe delovnih nalog. Ko kliknemo eno od map, se prikaže ustrezen seznam delovnih nalog.

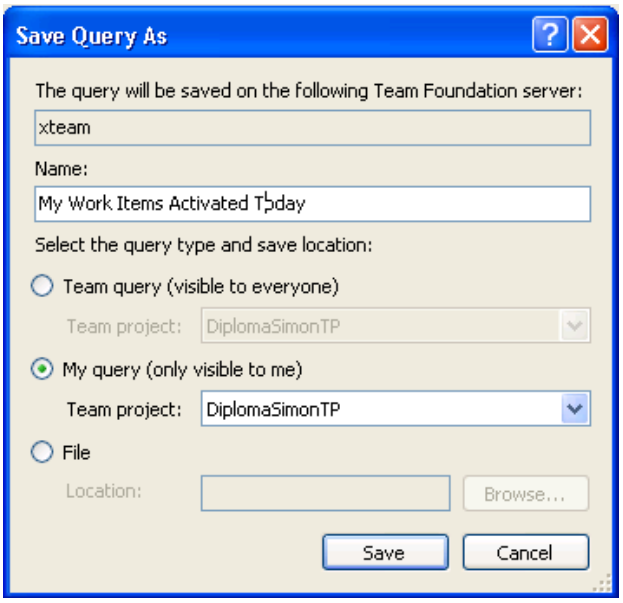
Drugi način je, da napišemo lastno poizvedbo. Glavne sestavine teh poizvedb so klavzule poizvedb, s katerimi lahko določimo kriterije in tako pridemo do želenih rezultatov.

Poizvedbo ustvarimo tako, da z desno tipko miške kliknemo mapo *My Queries (Moje poizvedbe)* v Team Explorer-ju in nato kliknemo *Add Query (Dodaj poizvedbo).* Nato dodamo klavzule, ki jih potrebujemo, in v orodni vrstici za poizvedbe kliknemo možnost *Run (Zaženi).*



Slika 68: Ustvarjanje nove poizvedbe za prikaz delovnih nalog

Prikaže se seznam delovnih nalog, ki ustrezajo kriterijem poizvedbe. Poizvedbe pa lahko tudi shranimo za nadaljnjo uporabo. To storimo tako, da kliknemo *File (Datoteka)-> Save New Query (Shrani novo poizvedbo)*. Odpre se spodaj prikazano pogovorno okno. Vpišemo naziv poizvedbe ter izberemo, komu naj bo vidna ta poizvedba (samo nam ali pa vsem uporabnikom v skupinskem projektu). Poizvedba se shrani v mapo *My Queries (Moje poizvedbe)* v Team Explorer-ju.

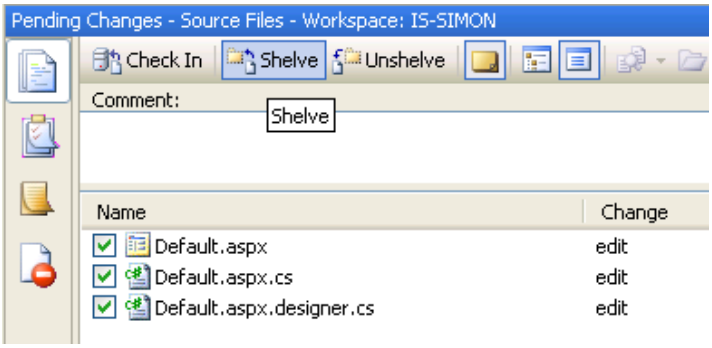


Slika 69: Shranjevanje nove poizvedbe za prikaz delovnih nalog

9.5 Odlaganje/pridobivanje kode

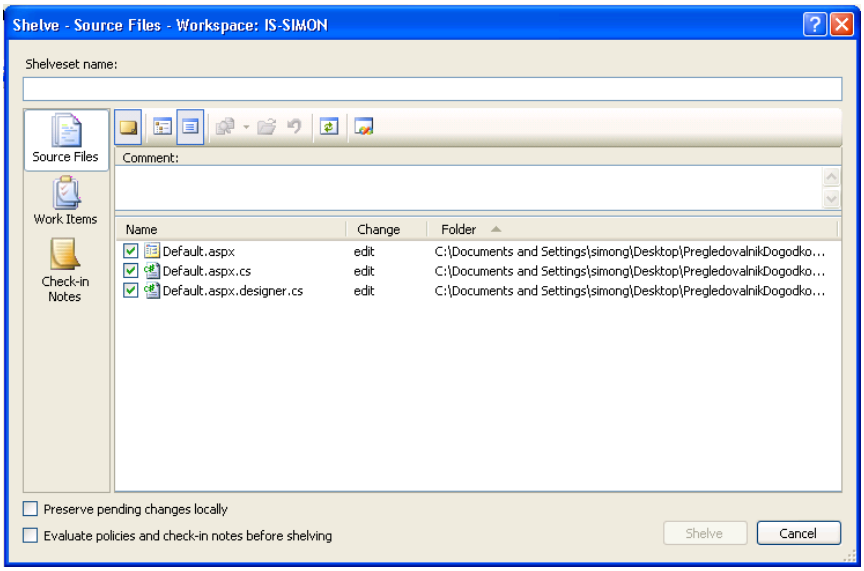
Odlaganje kode na strežnik nam omogoča, da odložimo seznam nerešenih sprememb na stran. To je zelo koristno v primerih, ko želimo shraniti naše čakajoče spremembe in še nismo rešili trenutne naloge do konca, vendar se moramo nujno lotiti naslednje, če želimo ostalim v skupini omogočiti dostop do te kode. Posledica odlaganja kode na strežnik se imenuje *Shelveset (Niz odlaganj).*

Odlaganje kode se izvede v oknu *Pending Changes (Čakajoče spremembe)*, kjer je seznam nerešenih datotek. Enostavno označimo datoteke, ki jih želimo odložiti na strežnik, in kliknemo *Shelve (Odloži).*



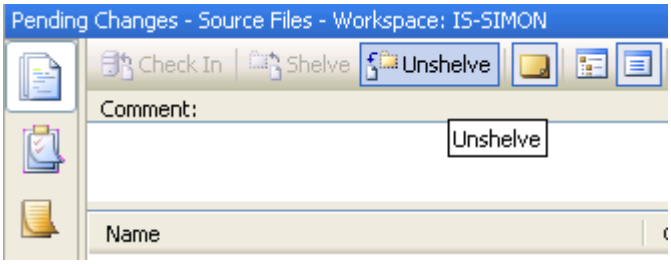
Slika 70: Odlaganje kode (prvi korak)

Odpre se pogovorno okno, kjer pod *Shelveset name (Ime niza odlaganj)* vnesemo naziv tega niza. Preverimo vse dodane datoteke, odznačimo *Preserve pending changes locally (Ohrani nedokončane spremembe lokalno)* in kliknemo *Shelve (Odloži).*

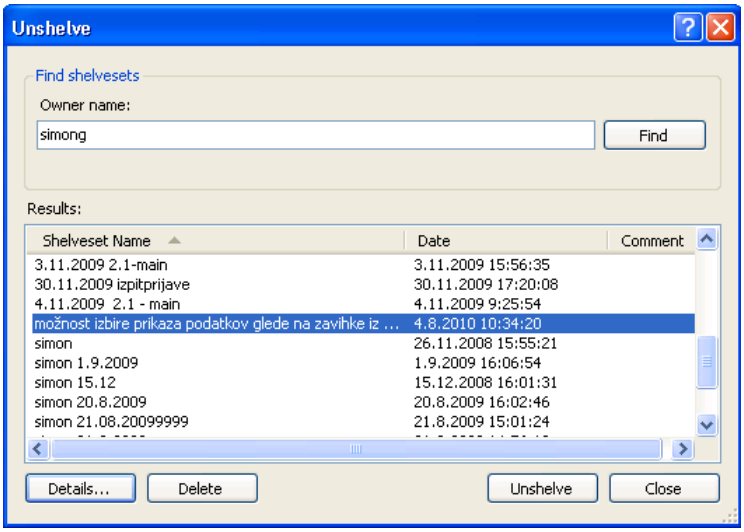


Slika 71: Odlaganje kode (drugi korak)

Če želimo pozneje pridobiti te datoteke nazaj v lokalno delovno okolje, kliknemo *Unshelve (Pridobi)*, izberemo ime primernega niza odlaganj ter še enkrat kliknemo *Unshelve (Pridobi).*



Slika 72: Pridobivanje kode (prvi korak)

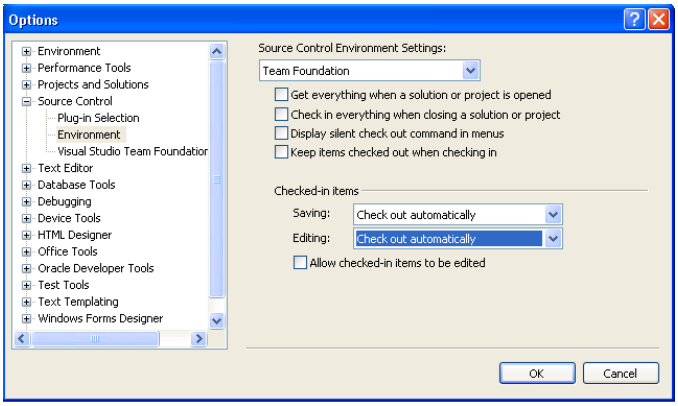


Slika 73: Pridobivanje kode (drugi korak)

9.6 Rezervacija/sprostitev kode

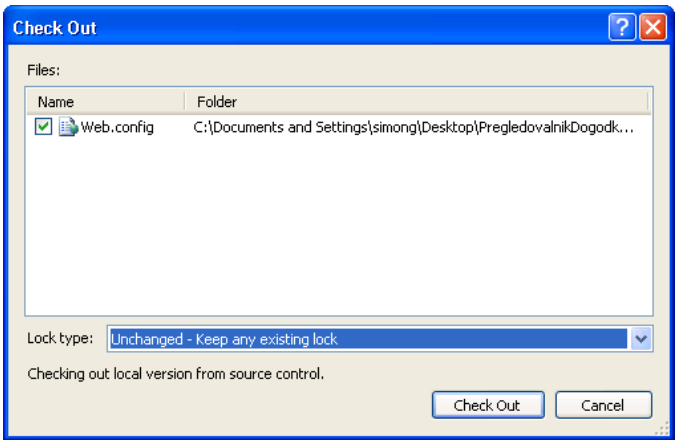
Check-out (Rezervacija) kopira datoteko iz strežnika v lokalni računalnik in odstrani lastnost »samo za branje«.

Uporabimo lahko več načinov za rezervacijo datoteke. Če želimo, da se datoteka samodejno rezervira,, ustrezno nastavimo polji *Saving (Shranjevanje)* in *Editing (Urejanje),* ki ga najdemo, če kliknemo *Tools (Orodja) -> Options (Lastnosti)*, vozlišče *Source Control (Nadzor nad viri)* in zavihek *Evironment (Okolje)* ter izberemo *Check-out Avtomatically (Samodejno rezerviraj).*



Slika 74: Nastavitev lastnosti za samodejno rezervacijo datoteke

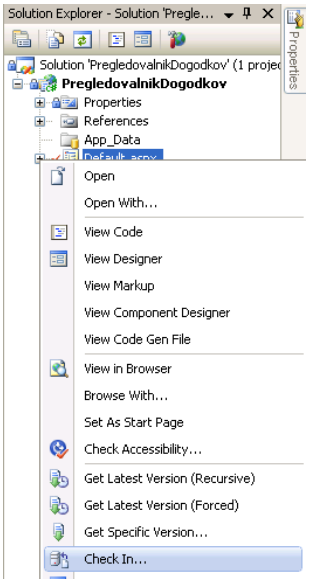
Če želimo, da smo vedno pozvani k rezervaciji, kliknemo *Prompt for Check out (Vprašaj za rezervacijo).*



Slika 75: Okno za rezervacijo datoteke

Ko končamo s spremembami, je čas, da sprostimo te spremembe nazaj na strežnik. To lahko storimo na dva načina:

- Z desno tipko miške kliknemo datoteko v Solution Explorerju ali v Source Control Explorerju ter izberemo *Check-in (Sprostitev)*.
  Odpre se pogovorno okno, ki vsebuje seznam vseh datotek, ki so na trenutno na voljo za sprostitev. Označimo datoteke, izberemo delovno nalogo (priporočljivo), komentar (priporočljivo) ter kliknemo *Check-in (Sprostitev)*.



Slika 76: Sprostitev datoteke

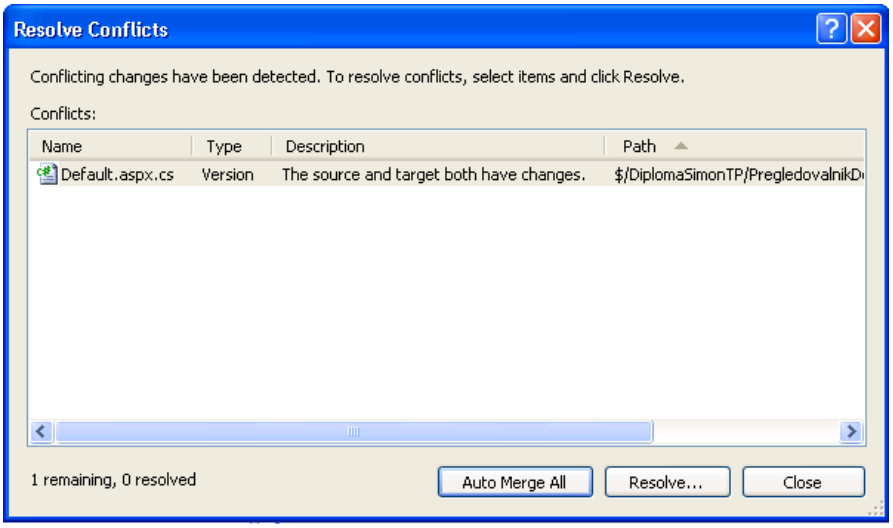
- Tretji način je uporaba okna *Pending Changes (Čakajoče spremembe)*. Označimo datoteke, ki jih želimo sprostiti, izberemo delovno nalogo (priporočljivo), komentar (priporočljivo) ter kliknemo *Check-in (Sprostitev)*.

9.7 Spajanje sprememb (Merging changes)

Spajanje sprememb pomeni reševati spore, do kateri pride, če rezerviramo datoteko:
- ki lokalno ni ista kot na strežniku (ni bila izbrana možnost *Get Lastest Version (Pridobi zadnjo različico)*) – nekdo je spreminjal datoteko, medtem ko je bila rezervirana;

- ko spajamo dve različici aplikacije itd.

Če pride do spora, se prikaže pogovorno okno *Resolve Conflict (Reši konflikt).*

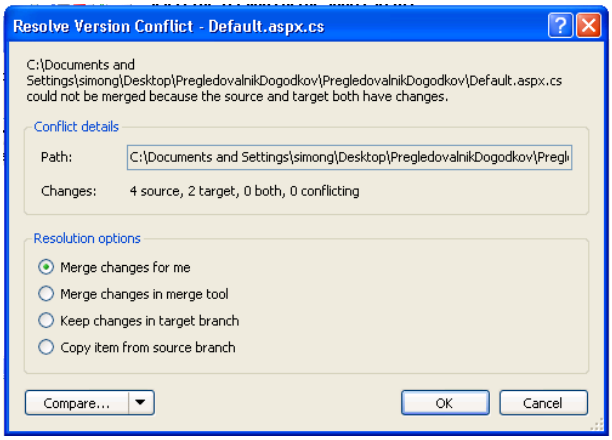


Slika 77: Okno za reševanje spora

Na voljo sta dve možnosti. Prva je izbira gumba *Auto Merge All (Samodejno spoji vse),* ki povzroči, da TFS sam opravi spajanje sprememb in reši spor.

Druga možnost je izbira gumba *Resolve (Razreši),* ki odpre okno *Resolve Version Conflict (Reševanje spora med različicami),* v katerem je na voljo več podrobnosti o sporu in tudi štiri možnosti za razrešitev spora:

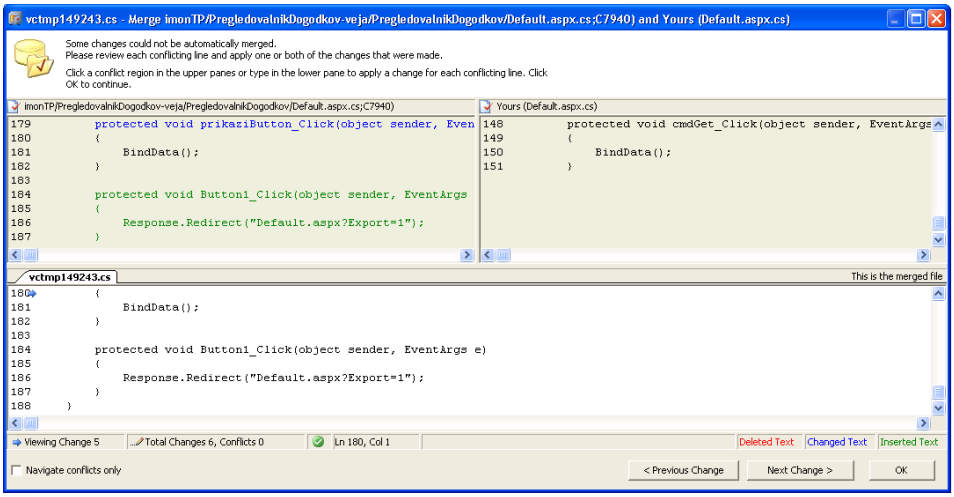
- Samodejno spajanje sprememb v orodju Microsoft Visual Studio.
- Spajanje sprememb dveh datotek z orodjem za spajanje.
- Razveljavitev sprememb, ki so bile narejene na lokalni kopiji datoteke.
- Razveljavitev sprememb, ki so bile narejene na strežniški kopiji datoteke.



Slika 78: Izbira možnosti za reševanje spora

Če se odločimo, da bomo sami reševali spor z orodjem za spajanje, izberemo drugo možnost (Spajanje sprememb dveh datotek z orodjem za spajanje). Orodje za spajanje je sestavljeno iz treh delov: obeh datotek, ki ju primerjamo, in rezultata spojene datoteke. Orodje za spajanje poudari besedilne razlike med dvema datotekama. Modra barva predstavlja spremenjen tekst,

zelena predstavlja novo vstavljeno besedilo in rdeča barva izbrisano besedilo. S premikom kazalnika miške v okno rezultata spojene datoteke lahko premaknemo trenutni položaj in vstavimo spremembe iz ene od dveh nasprotujočih si datotek.



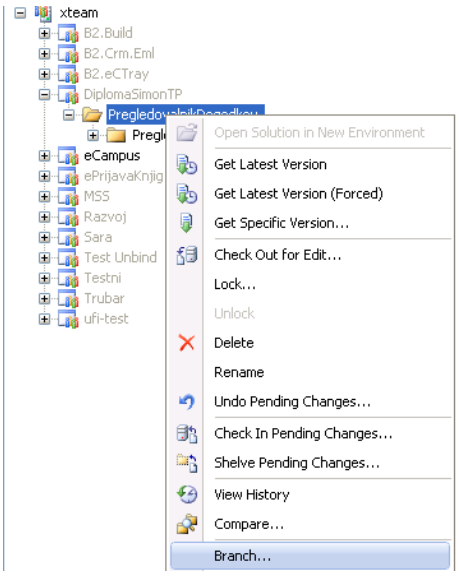
Slika 79: Orodje za spajanje

9.8 Vejanje/Spajanje vej

Vejanje je funkcija sistema za nadzor nad viri za ustvarjanje novih datotek ali map, ki temeljijo na obstoječih. Vejanje se lahko uporablja iz različnih razlogov:

- izolacija vzporednih skupin;
- skupina dela na ločenih funkcijah, vendar se te funkcije prekrivajo;
- podpora izdaji.

Če želimo narediti vejo, moramo odpreti kontrolnik virov. V kontrolniku virov izberemo mapo, z desno tipko miške kliknemo to mapo in izberemo *Branch (Vejanje).*

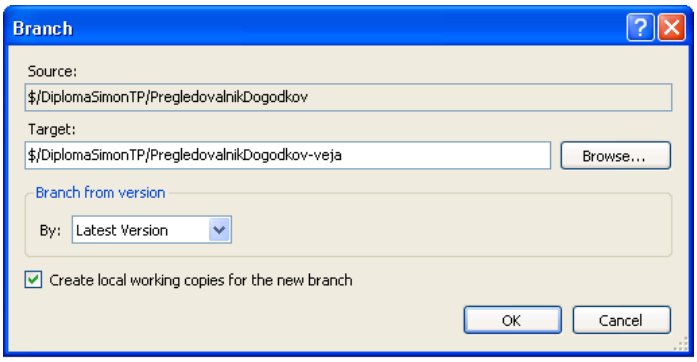


Slika 80: Ustvarjanje veje

Odpre se novo pogovorno okno *Branch (Veja).* V polju *Source (Vir)* je projekt, iz katerega bomo naredili vejo, v polje *Target (Cilj)* pa vnesemo ime novega projekta (veje). V polju *Target (Cilj)* so na voljo možnosti za ustvarjanje veje iz določene različice. Te možnosti so:

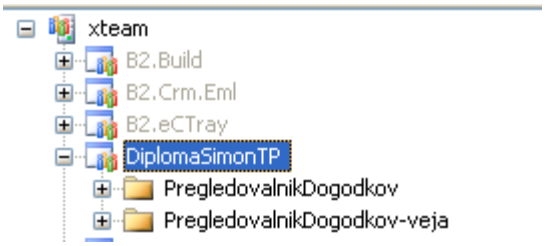
- Niz sprememb
- Datum
- Oznaka
- Najnovejša različica
- Delovni prostor

Izbrali bomo zadnjo različico, ki je tudi najprimernejša in najpogosteje uporabljena. Na dnu označimo možnost, da želimo novo vejo kopirati tudi lokalno v delovni prostor, in kliknemo *OK (V redu).*



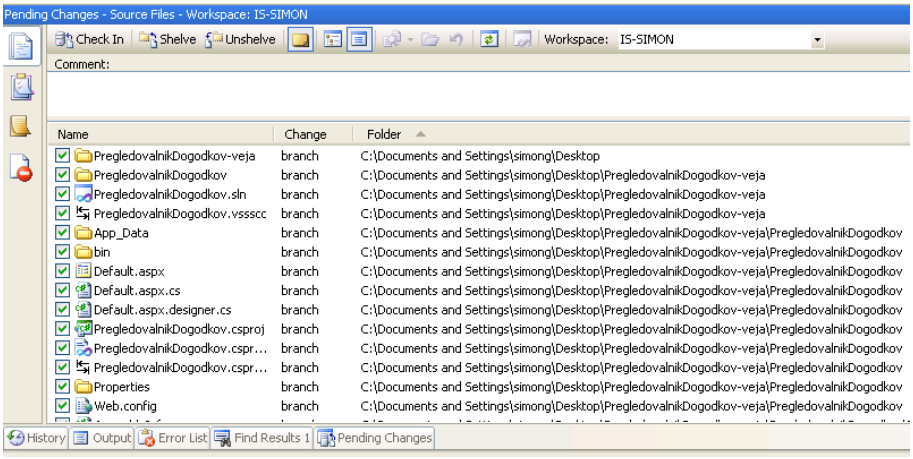
Slika 81: Okno za poimenovanje veje

Ustvari se nova veja *PregledovalnikDogodkov-veja*, ki je vidna tudi v oknu kontrolnika virov.



Slika 82: Rezultat vejanja

Vse novo ustvarjene mape in datoteke so zdaj rezervirane in jih moramo sprostiti.

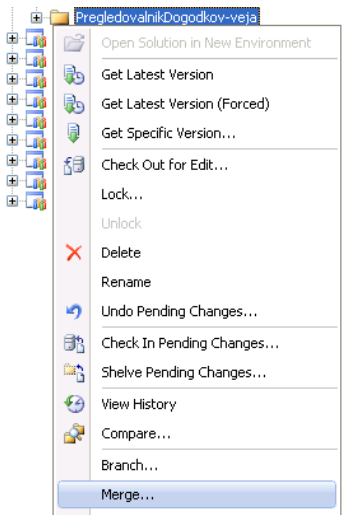


Slika 83: Rezervirane datoteke po vejanju

Spajanje vej je proces združevanja sprememb dveh različnih vej. Postopek spajanja vzame spremembe, ki so se zgodile na neki veji, in jih integrira v ciljno vejo. Spajanje vej vključuje vse vrste sprememb (spremembe imen, urejanje datotek, dodane datoteke, izbrisane datoteke in neizbrisane spremembe). Če so bile iste datoteke spremenjene na obeh vejah, moramo rešiti spor.

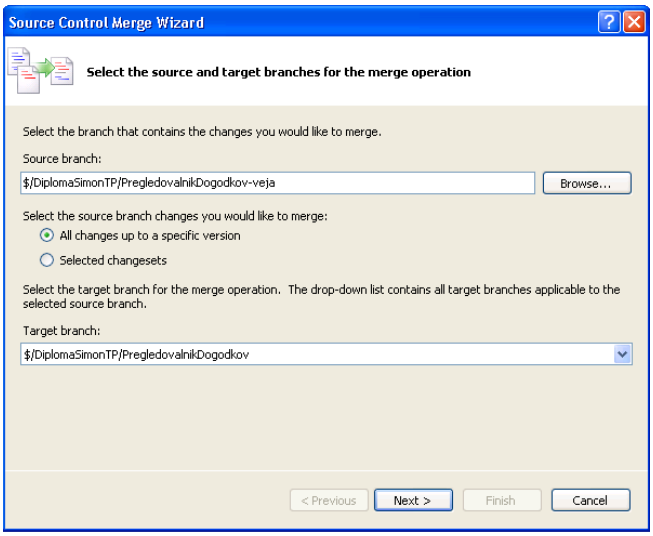
Ko ustvarimo novo vejo *PregledovalnikDogodkov-veja* in ji dodamo novo funkcionalnost (kodo), jo želimo integrirati nazaj na vejo *PregledovalnikDogodkov*.

To storimo tako, da odpremo kontrolnik virov, poiščemo naš vir (PregledovalnikDogodkov-veja), z desno tipko miške kliknemo vir in izberemo *Merge (Spoji).*



Slika 84: Spajanje vej

Odpre se pogovorno okno, kjer izberemo veji *Source (Vir)* in *Target (Cilj)* ter kliknemo *Next (Naprej).*



Slika 85: Določitev vej za spajanje

Tako kot pri vejanju moramo tudi tukaj izbrati , iz katere različice želimo narediti vejo. Te možnosti so:

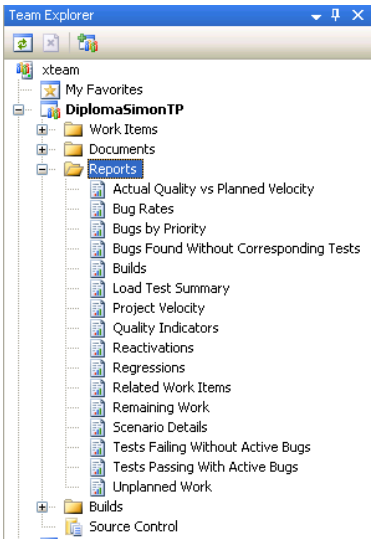
- Niz sprememb
- Datum
- Oznaka
- Najnovejša različica
- Delovni prostor

Znova izberemo možnost *Najnovejša različica* in kliknemo *Finish (Končaj).* Vse datoteke, ki se na viru kakor koli razlikujejo (spremenjene, dodane ali izbrisane) od ciljnih, so prikazane kot rezervirane v oknu *Pending Changes (Čakajoče spremembe)* in jih moramo nato sprostiti. Edina težava, ki lahko nastane, je spor med dvema datotekama. Tega rešimo po postopku, ki je opisan v točki 1.9.

9.9 Poročanje

Poročila, ustvarjena s strežnikom Team Foundation Server, nam pomagajo hitro oceniti stanje skupinskega projekta, kakovost programske opreme v razvoju, napredek v smeri proti končanju projekta itd. Ta poročila povzamejo meritve iz delovnih nalog, sistema za nadzor nad viri, rezultatov testov in graditev. Poročila nam na primer lahko povejo, kako hitro naša ekipa dela od tedna do tedna, odvisno od njihove dejanske aktivnosti.

Ko ustvarimo nov skupinski projekt, čarovnik ustvari seznam standardnih poročil v skladu s specifikacijami predloge procesa. Poročila so shranjena po abecednem vrstnem redu v oknu Team Explorer pod vozliščem *Reports (Poročila)* za skupinski projekt. Poročila, ki jih gledamo, so vedno shranjena v načinu »samo za branje«.

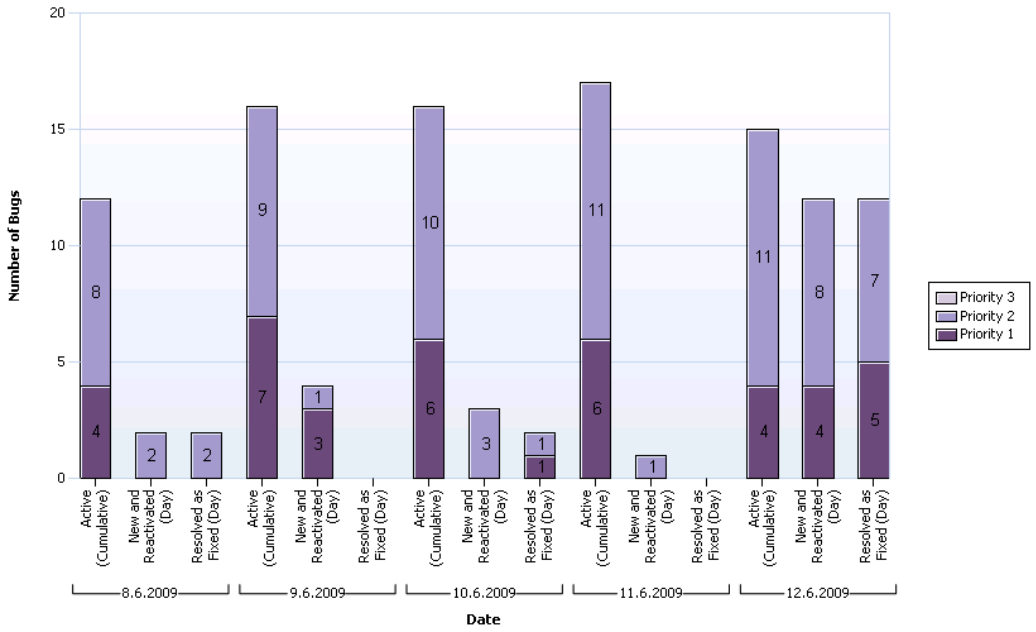


Slika 86: Seznam sporočil po tem, ko smo ustvarili nov skupinski projekt

Standardna poročila odpremo tako, da kliknemo enega izmed poročil. V našem primeru bomo izbrali poročilo *Bugs by Priority (Hrošči po prioriteti)*. Odpre se obrazec za določanje podatkov o prikazu poročila.

Slika 87: Določanje podatkov o prikazu poročila

Izberemo ponovitev, področje, začetni in končni datum ter kliknemo *View Report (Ogled poročila)*.

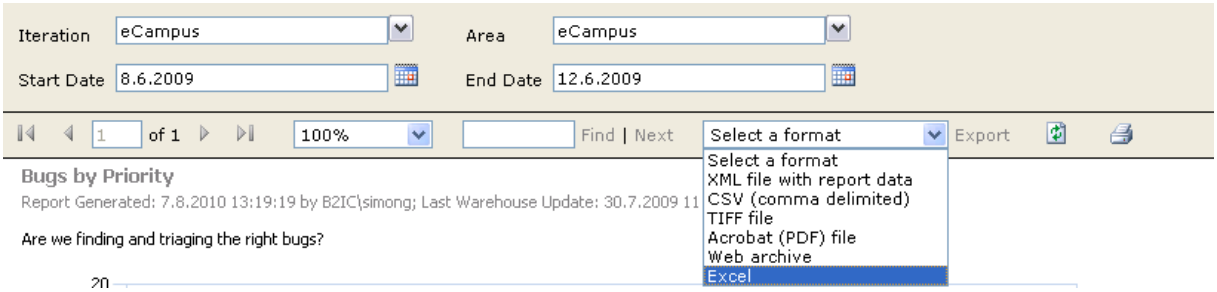


Slika 88: Graf poročila

Graf predstavlja število hroščev glede na prioriteto (prioriteta 1, prioriteta 2 in prioriteta 3) ter glede na status (aktivni, novo dodani in razrešeni hrošči) za posamezen dan. Dne 10. 6. 2009 je bila statistika takšna:
- aktivni hrošči: 16 (od tega šest s prioriteto 1 in deset s prioriteto 2)
- aktivni hrošči: 3 (od tega trije s prioriteto 1)
- aktivni hrošči: 2 (od tega eden s prioriteto 1 in eden s prioriteto 2)

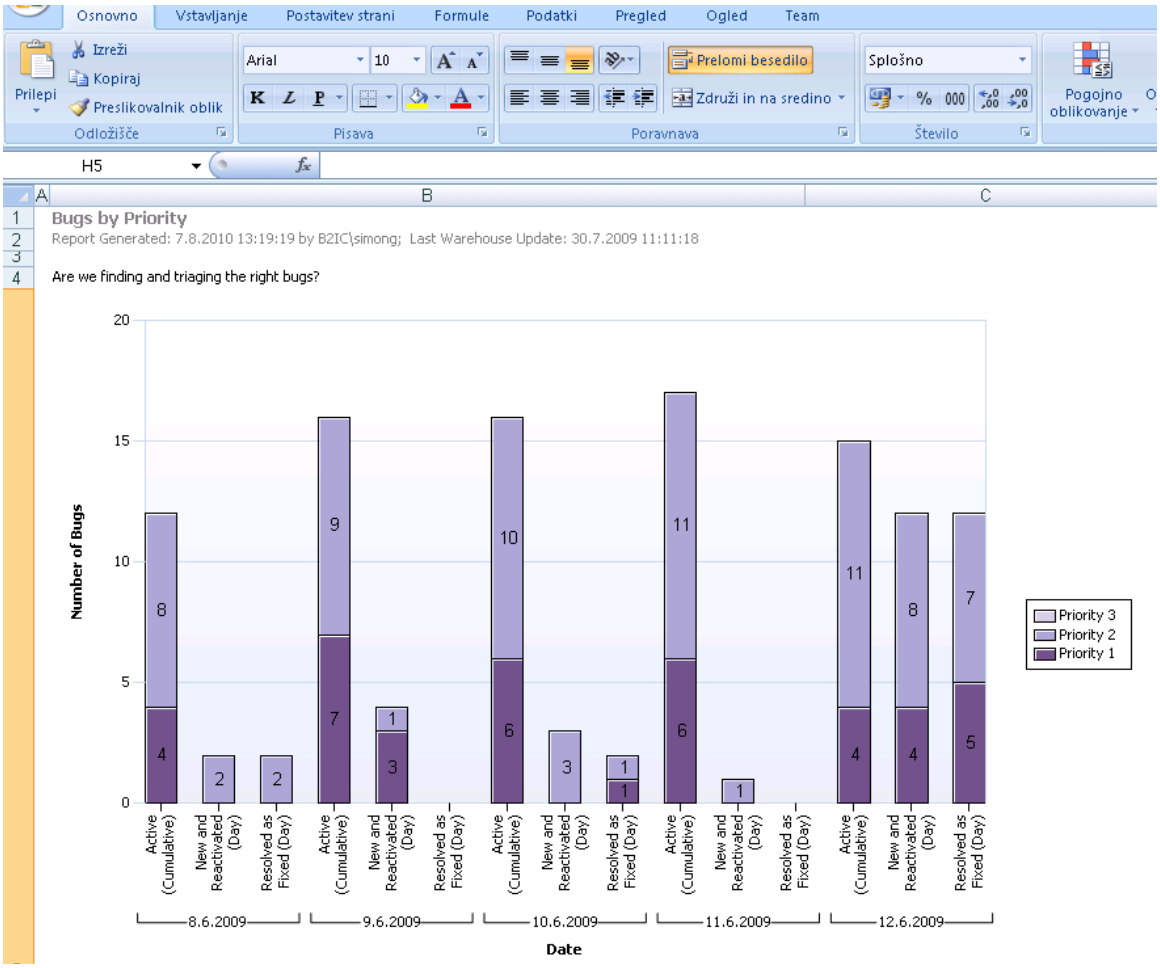
Poročilo lahko izvozimo tudi v različne formate. Po prikazu grafa se prikaže novo polje, kjer lahko izbiramo, kam želimo graf izvoziti:

- v datoteko XML
- v datoteko CSV (Comma Separated Values)
- v datoteko TIFF (Tagged Image File Format)
- v datoteko PDF (Portable Document Format)
- v spletni arhiv
- v Microsoft Excel



Slika 89: Določanje podatkov za izvoz v Microsoft Excel

Odločimo se za Microsoft Excel in kliknemo *Export (Izvoz).* Prikaže se isti graf kot prej, le da je zdaj prikazan v programu Microsoft Excel.



Slika 90: Graf poročila v programu Microsoft Excel

10. Zaključek

Cilj diplomske naloge je bil preučiti strežnik Team Foundation Server, pridobiti lastno znanje o strežniku in prikazati njegovo uporabo na način, da bo vsem razumljiv. Izdelava diplomske naloge je potekala po korakih. Prvi korak je bil pridobiti vso literaturo. Sledil je opis strežnika, kjer sem opisal samo najpomembnejše in največkrat uporabljene lastnosti, ker bi sicer naloga postala preobširna. Naslednji in najpomembnejši korak je bila ideja o aplikaciji, realizacija le-te in prikaz uporabe strežnika Team Foundation Server pri celotnem razvoju aplikacije. Zadnji korak je bila izbira primernega in uveljavljenega odprtokodnega sistema. Odločil sem se za sistem Subversion, ki smo ga uporabljali že na faksu.

Pri diplomski nalogi je prišlo do zapletov pri uporabi strežnika Team Foundation Server. Strežnik je seveda plačljiv in zato implementacija ter uporaba doma ni bila mogoča. Dogovoriti sem se moral s podjetjem, kjer sem zaposlen, da z uporabo oddaljenega dostopa dostopam do strežnika v podjetju. Dodeliti so mi morali tudi pravice glavnega skrbnika, sicer prikaz ne bi bil mogoč.

Pri pisanju diplomske naloge sem pridobil veliko znanja in izkušenj z uporabo strežnika Team Foundation Server, ki mi bodo koristila predvsem v podjetju, kjer strežnik tudi uporabljamo. Predvsem sem dobil vpogled v tisti del strežnika, ki je namenjen uporabi vodjem projekta in ga razvijalci med razvojem projekta ne spoznajo. Hkrati sem utrdil tudi znanje programiranja v objektnem jeziku C# in načrtovanja razvoja aplikacije.

Strežnik Team Foundation Server je le ena izmed komponent orodja Visual Studio Team System, zato le-to ponuja še veliko možnosti za raziskovanje in spoznavanje njegovih komponent.

Seznam slik

Literatura

[1] (2009) Definicija i forma Team Foundation Server-a. Dostopno na: http://www.link-elearning.com/linkdl/elearning/jedinica.php?IDJedinice=5225

[2] J.D. Meier, Jason Taylor, Alex Mackman, Prashant Bansod, Kevin Jones, *Team Development with Visual Studio Team Foundation Server*, 2007

[3] Lars Powers, Mike Snell, *Microsoft Visual Studio Unleashed*, 2008

[4] (2009) Branching and merging with Team Foundation Server 2010. Dostopno na: http://intovsts.files.wordpress.com/2010/06/techdays2010_branchingandmergingwithtfs2010.pdf

[5] (2009) Working with Source Control Changesets. Dostopno na: http://msdn.microsoft.com/en-us/library/ms181408%28VS.80%29.aspx

[6] (2009) How to: Set Permissions for a Report . Dostopno na: http://msdn.microsoft.com/en-us/library/ms181645%28v=VS.80%29.aspx

[7] (2009) Team Foundation Server Contributor Permissions. Dostopno na: http://msdn.microsoft.com/en-us/library/ms252467%28v=VS.80%29.aspx

[8] (2009) Team Foundation Server Project Lead Permissions. Dostopno na: http://msdn.microsoft.com/en-us/library/ms253174%28v=VS.80%29.aspx

[9] (2009) Team Foundation Server Administrator Permissions. Dostopno na: http://msdn.microsoft.com/en-us/library/ms253096%28v=VS.80%29.aspx

[10] (2009) How to: Create a Report in Microsoft Excel for Team System. Dostopno na: http://msdn.microsoft.com/en-us/library/ms244699%28v=VS.80%29.aspx

[11] (2009) How to: Edit a Report in Microsoft Excel for Team System. Dostopno na: http://msdn.microsoft.com/en-us/library/ms244692%28v=VS.80%29.aspx

[12] (2009) Creating Reports in Microsoft Excel and Visual Studio 2010 (Team Foundation Server 2010).
Dostopno na: http://www.dotnetcurry.com/ShowArticle.aspx?ID=528

[13] (2009) Creating a Report with Report Designer. Dostopno na: http://msdn.microsoft.com/en-us/library/ms156280%28SQL.90%29.aspx

[14] (2008) Visual Studio Team System tutorials: What is VSTS? Dostopno na: http://www.dotnetspider.com/Vsts-Tutorial-198.aspx

[15] (2010) Visual Studio Team System. Dostopno na: http://msdn.microsoft.com/en-us/vstudio/dd430910.aspx