UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

SKRBNIŠTVO IZDAJ PROGRAMSKE OPREME

Ljubljana, maj 2008 Alen Oblak

IZJAVA

Študent Alen Oblak izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal pod mentorstvom prof. dr. Miro Gradišar in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega/specialističnega dela na fakultetnih spletnih straneh.

V Ljubljani, dne _____ Podpis:_____

Vsebina

Kazalo slik

Kazalo tabel

Slovarček slovenskih prevodov tujih izrazov

Angleški izraz	**Slovenski prevod**
Availability Management	Upravljanje razpoložljivosti
Build Management	Upravljanje gradnje
Capacity Management	Upravljanje zmogljivosti storitev
Change Management	Upravljanje sprememb
Change Advisory Board (CAB)	Odbor za spremembe
Change Manager	Upravitelj sprememb
Configuration Item (CI)	Element konfiguracije
Configuration Management	Upravljanje konfiguracij
Configuration Management Database (CMDB)	Centralna zbirka podatkov o konfiguracijah
Configuration Management system (CMS)	Sistem za upravljanje konfiguracij
Continuity Management	Upravljanje neprekinjenosti delovanja
Definitive Software Library (DSL)	Knjižnica veljavne programske opreme
Delta Release	Delta izdaja
Emergency fix	Izredna izdaja
Financial Management for IT Services	Finančno upravljanje IT storitev
Full Release	Celotna izdaja
Hot-fix	Izredna izdaja
Incident Management	Upravljanje incidentov
Information System (IS)	Informacijski sistem
Information Technology (IT)	Informacijska tehnologija
Key performance indicator (KPI)	Ključni kazalec uspešnosti
Major Release	Večja izdaja
Minor Release	Manjša izdaja
Packaged Release	Paketna izdaja
Problem Management	Upravljanje problemov
Release	Izdaja
Release Back-out	Umik izdaje
Release Management	Skrbništvo izdaj, Upravljanje izdaj
Release Manager	Upravitelj izdaj
Release Notes	Izdajni zapisnik
Release Policy	Politika izdajanja
Release Roll-out	Uvedba izdaje
Release Unit	Enota izdaje
Request for Change (RFC)	Zahtevek za spremembo
Risk Management	Obvladovanje tveganj
Service	Storitev
Service Delivery	Dostava storitev
Service Desk	Storitveni center
Service Level Agreements (SLA)	Dogovor o ravni storitev
Service Level Management	Upravljanje ravni storitev
Service Support	Vzdrževanje storitev
Test Manager	Upravitelj testiranja
Upgrade	Nadgradnja
Workaround	Nadomestna rešitev

1. Uvod

1.1. Skrbništvo izdaj

Tema magistrske naloge je skrbništvo izdaj[1] programske opreme. Izdaja ni vsakdanja beseda – besedi *razvoj* in *uporaba* programske opreme, ki opisujeta stanji v življenjskem ciklu programske opreme, sta veliko bolj pogosti. Vendar igra izdaja programske opreme ključno vlogo pri povezavi razvoja opreme in njene kasnejše uporabe (Hall, 1999).

Potreba po razvoju novih storitev IT (*informacijske tehnologije*) in spreminjanju obstoječih je jasna. IT storitve predstavljajo ključ do uspeha mnogih podjetij, saj zagotavljajo konkurenčno prednost na vedno bolj zahtevnem trgu. Po drugi strani se od informacijske tehnologije pričakuje neprekinjeno delovanje, saj lahko že kratkotrajna motnja v delovanju predstavlja ogromno poslovno izgubo ali celo resno grožnjo obstoju marsikaterega podjetja. Torej je uspešen prenos programske opreme iz razvoja v produkcijo ključni element informacijskih storitev, za katere skrbijo vzdrževalci programske opreme. Celovito upravljanje prenosov skrbi za racionalnost, uspešnost in učinkovitost vseh aktivnosti v povezavi z izdajo in nameščanjem programske opreme (COBIT, 2005), kar podjetjem omogoča hitro prilagajanje konstantnim spremembam.

Optimalno izvajanje aktivnosti se od podjetja do podjetja razlikuje, saj je potrebno upoštevati splošne nevarnosti, vire in druge značilnosti nekega podjetja ter dejanske lastnosti izdane programske opreme. Pri prenosu programske opreme od razvoja do produkcije je potrebno upoštevati hitrost, ceno in varnost prenosa. Prioritete, skrbi, omejitve in pogoji, ki diktirajo odločitve, se med podjetji razlikujejo. V nekaterih panogah je najpomembnejša varnost (jedrska tehnologija, medicina), v nekaterih je bolj pomembno hitro doseganje ciljev (mobilna telefonija, online prodaja), vsem pa je skupen temeljni koncept: razvito programsko opremo je potrebno prenesti v produkcijsko rabo, kjer bo zadovoljila poslovne potrebe (ITIL, 2007).

Primarna dejavnost podjetja, kjer sem trenutno zaposlen, je razvoj in vzdrževanje zavarovalniškega informacijskega sistema. Ker se obseg in z njim kompleksnost informacijskega sistema povečuje, kakor se povečuje tudi število ljudi, ki IS (*informacijski sistem*) razvijajo, ter število strank, ki IS uporabljajo, nam obstoječe metode izdajanja in nameščanja programske opreme ne zadovoljujejo več vseh potreb, na nekaterih področjih pa predstavljajo omejitveni dejavnik pri širjenju poslovanja.

Upravljanje z izdajami je celota procesov, s katerimi razvito programsko opremo prenesemo k uporabniku v uporabo (Hoek et al., 1996; Smith, 1999). S pojavom modularne programske opreme se je upravljanje z izdajami nekoliko zakompliciralo. Moduli, iz katerih je

[1] Ker se angleški izraz »release management« prevaja tako v »upravljanje izdaj« kot v »skrbništvo izdaj«, sta v nalogi enakovredno uporabljana oba izraza.

programska oprema sestavljena, se namreč lahko razvijajo in izdajajo neodvisno drug od drugega, hkrati pa so lahko drug od drugega odvisni (Jansen, Brinkkemper, 2005). Med razvojem je potrebno poznati kompleksne in spreminjajoče se odvisnosti med moduli, ki sestavljajo določen IS, ter takšne odvisnosti dokumentirati (CMMI, 2006). Pri izdaji modularne programske opreme pa se mora glede na medsebojne odvisnosti pripraviti takšna izdaja, ki zagotovi ustrezno verzijo vseh modulov IS (Brinkkemper, Klint, 2003).

Stalni spremljevalki velikih sprememb sta kompleksnost in tveganje. Potrebno je obvladati veliko število medsebojnih odvisnosti in usklajevati nasprotujoče si prioritete, medtem ko se novi in spremenjeni moduli izdajajo, nameščajo, testirajo in uporabljajo. Na vsakem koraku je potrebno tako kot možnost uspeha predvideti tudi posledice napak ter se odločati v skladu z njihovim razmerjem. K razumevanju le tega nam pripomorejo vsakodnevne napovedi in ocene bodočih zmogljivosti ter tveganj glede na trenutno stanje (ITIL, 2007).

Upravljanje z izdajami je lahko uspešno le, če mu sorodni procesi zagotavljajo trdne temelje: upravljanje sprememb, zagotavljanje kvalitete, obvladovanje tveganj, vodenje projektov (ISO/ IEC 20000-2, 2005). Šele takrat je možno na vsakem koraku planirati, voditi in nadzirati potek procesov ter jih voditi glede na zastavljene cilje in temu primerno ukrepati.

1.2. Cilj magistrskega dela

Cilj magistrskega dela je izboljšanje upravljanja z izdajami v podjetju In2. Glede na predlagane izboljšave sistema za upravljanje z izdajami in avtomatizacijo nekaterih procesov pričakujem:

- enostavnejše izdajanje programske opreme, kar bo povzročilo hitrejše razvojne cikle in naklonjenost tako ponudnika kot uporabnikov k večjemu številu izdaj,
- manjše število napak pri nameščanju programske opreme, kar bo povečalo zaupanje v proces nameščanja programske opreme,
- višjo kvaliteto izdajanja programske opreme zaradi bolje definiranih procesov,
- zmanjšanje količine zamudnega ročnega dela na strani ponudnika storitev, torej bolj produktivno uporabo virov in manjšo možnost napak,
- višji nivo varnosti zaradi definirane politike nameščanja: kdo lahko kaj namesti kam in kdaj.

1.3. Metode dela

V magistrski nalogi sem vseskozi uporabljal svoje izkušnje pri delu v informatiki. Po študiju literature in člankov sem kritično ovrednotil trenutno stanje upravljanja z izdajami in upravljanja namestitev v podjetju. Sledi primerjalna analiza, kjer sem aktualne procese podjetja primerjal in poizkušal nadgraditi do svetovno uveljavljenih najboljših praks. Predloge izboljšav sem ocenil glede na strošek njihove realizacije in pričakovano donosnost rezultatov ter predlagal vrstni red vpeljave.

1.4. Temeljna vprašanja

Ker je že mnogo strokovnjakov preučevalo procese, povezane s skrbništvom izdaj programske opreme, in je posledično bilo na to temo napisane veliko literature, je smiselno, da vsaj nekaj te literature preučim in se seznanim z nekaterimi praksami. Zato postavim prvo temeljno vprašanje magistrskega dela:

Katere so svetovno uveljavljene najboljše prakse izdajanja programske opreme?

Če želim izboljšati procese izdajanja programske opreme v podjetju, kjer sem trenutno zaposlen, moram dodobra spoznati trenutno stanje v podjetju. Pri tem lahko izkoristim svoje izkušnje in poznavanje procesov podjetja, navsezadnje jih tudi sam izvajam v sklopu svojih vsakodnevnih zadolžitev. Drugo temeljno vprašanje magistrskega dela bo torej:

Kakšni procesi, povezani z izdajanjem programske opreme, so trenutno v uporabi v podjetju In2?

Pri optimizaciji poslovnih procesov nekega podjetja je potrebno upoštevati značilnosti podjetja in okolja, v katerem deluje. Ogrodja, kot je recimo ITIL, so zelo splošna in avtorji vseskozi poudarjajo, da je pri vpeljavi nekega procesa potrebno kritično ovrednotiti smiselnost takega početja. Nekateri pristopi so npr. primerni za manjša, nekateri pa za večja podjetja in kar odlično deluje v enem, je lahko pogubno za drugo podjetje. Zato se tretje in zadnje temeljno vprašanje magistrskega dela glasi:

Po katerih praksah bi se lahko zgledovali, jih poizkušali vpeljati ali prilagoditi našim potrebam in tako izboljšati naše procese?

1.5. Zasnova magistrskega dela

V prvem delu je predstavljena raznovrstna literatura s področja upravljanja izdaj. Splošnim konceptom in uporabljani terminologiji sledi poglobljeno obravnavanje tega področja.

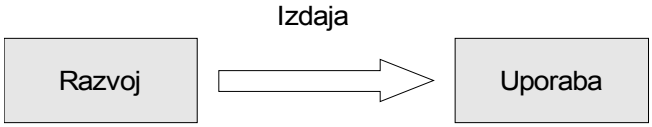
V drugem delu je upravljanje izdaj v dejanskem podjetju primerjano s koncepti in najboljšimi praksami, pri čemer so nakazane prednosti in slabosti trenutnih aktivnosti.

V zadnjem delu so obravnavani incidenti, povezani z upravljanjem izdaj, ki skupaj s primerjavo služijo kot osnova za ugotavljanje možnih izboljšav. Dalje so predstavljeni in kritično ovrednoteni predlogi izboljšav, glede na zahtevnost njihove realizacije je podan predlog vrstnega reda vpeljave.

2. Skrbništvo izdaj

Upravljanje z izdajami je celota procesov, s katerimi razvito programsko opremo prenesemo k uporabniku v uporabo (Hoek et al., 1996; Smith, 1999), in spada med primarne procese življenjskega cikla programske opreme (ISO/IEC 12207.0-1996, 1998; ISO/IEC TR 15504-2, 1998). Izdaja je sestavljena iz novih ali spremenjenih delov programske in strojne opreme, ki so potrebni za implementacijo določene aplikacije. S pojavom modularne programske opreme se je upravljanje z izdajami nekoliko zakompliciralo. Moduli, iz katerih je programska oprema sestavljena, se namreč lahko razvijajo in izdajajo neodvisno drug od drugega, hkrati pa so lahko drug od drugega odvisni (Jansen, Brinkkemper, 2005). Med razvojem je potrebno poznati kompleksne in spreminjajoče se odvisnosti med moduli, ki sestavljajo določen IS, ter takšne odvisnosti dokumentirati (CMMI, 2006). Pri izdaji modularne programske opreme pa se mora glede na medsebojne odvisnosti pripraviti takšna izdaja, ki zagotovi ustrezno verzijo vseh modulov IS (Brinkkemper, Klint, 2003).

Slika 1: izdaja – vezni člen med razvojem in uporabo programske opreme.



Vir: lasten.

Stalni spremljevalki velikih sprememb sta kompleksnost in tveganje. Potrebno je obvladati veliko število medsebojnih odvisnosti in usklajevati nasprotujoče si prioritete, medtem ko se novi in spremenjeni moduli izdajajo, nameščajo, testirajo in uporabljajo. Na vsakem koraku je potrebno tako kot možnost uspeha predvideti tudi posledice napak ter se odločati v skladu z njihovim razmerjem. K razumevanju le tega nam pripomorejo vsakodnevne napovedi in ocene bodočih zmogljivosti ter tveganj glede na trenutno stanje (ITIL, 2007). Večji in kompleksnejši kot je projekt, večja je potreba po učinkovitem upravljanju izdaj.

2.1. Terminologija

V literaturi, ki obravnava upravljanje izdaj programske opreme, se uporabljajo termini, specifični za to področje. Sledi navedba najpogosteje uporabljanih terminov skupaj z njihovo definicijo.

Izdaja programske opreme: zbirka potrjenih sprememb določene IT storitve. Izdajo definirajo zahtevki za spremembe, ki jih ta vsebuje. Izdaja je ponavadi sestavljena iz določenega števila popravkov, ki odpravljajo probleme in/ali določenega števila nadgradenj storitve. Izdaja se nanaša tako na programsko kot tudi na strojno opremo. (ITIL, 2000)

Izdati programsko opremo: pripraviti programsko opremo na način, da postane dosegljiva stranki (Hoek et al., 1996). Po definiciji torej izdaja ni le korak, ki sledi razvoju programske opreme, ampak vsebuje tudi pripravo na ta korak.

Upravitelj izdaj: oseba, ki nadzira integrirane procese razvoja, testiranja in nameščanja programske opreme. Spozna se na življenjski cikel razvoja programske opreme, različne platforme, na katerih le ta teče, ter na poslovne potrebe, katerim programska oprema služi. Upravitelj med drugim nastopa v vlogi koordinatorja med uporabniki in razvijalci ter skrbi za učinkovito izvedbo potrebnih aktivnosti (Wikipedia: Release management, 2007).

Namestitev programske opreme: proces namestitve je sestavljen iz dostave, zagona in vzdrževanja programske opreme (Hall, 1997; Dolstra, Visser, Jonge, 2004). Programsko opremo je torej potrebno prenesti k stranki, jo tam usposobiti za uporabo in jo kasneje vzdrževati.

Glede na obseg sprememb delimo izdaje na:

- **Večja izdaja**: ponavadi vsebuje veliko število novih funkcionalnosti, hkrati pa odpravi kakšen obstoječ problem in tako nadomesti nameščanje popravkov.
- **Manjša izdaja**: ponavadi vsebuje manjše dograditve in popravke.
- **Izredna izdaja**: ponavadi vsebuje popravke za specifične probleme. (ITIL, 2007)

Glede na način nameščanja delimo izdaje na:

- **Delta izdaja**: nameščeni bodo samo spremenjeni deli programske opreme, npr. izredni popravek.
- **Celotna izdaja**: nameščeni bodo vsi moduli, npr. prva namestitev določene aplikacije.
- **Paketna izdaja**: nameščeno bo večje število manjših sprememb, npr. najprej manjša izdaja, zatem pa še več izrednih popravkov. (ITIL, 2007)

Glede na obseg opravljenega testiranja delimo izdaje na:

- **Alfa izdaja**: izdaja, na kateri so bili opravljeni le osnovni testi, navadno se takšne izdaje uporabljajo z namenom testiranja znotraj organizacije oz. v zaprtem krogu izbranih posameznikov.
- **Beta izdaja**: prva izdaja, namenjena zunanjim uporabnikom z namenom testiranja na bolj realnih primerih. Beta izdaja vsebuje že vso predvideno funkcionalnost, lahko pa so prisotne manjše napake.
- **Kandidat za izdajo**: potencialna končna izdaja, brez večjih napak, z vso predvideno funkcionalnostjo. Dovoljene so le še manjše spremembe izvorne kode, popravki dokumentacije, testnih procedur ali pripomočkov.
- **Končna izdaja**: zadnja verzija določenega produkta, identična potencialni izdaji z manjšimi popravki. Za takšno izdajo se pričakuje stabilnost delovanja in zadovoljivo kvaliteto, primerno za distribucijo končnim uporabnikom. (Bays, 2003; Wikipedia: Software release life cycle, 2007)

2.2. ITIL

The Information Technology Infrastructure Library oz. ITIL je ogrodje najboljših praks, konceptov in tehnik, s katerimi se lahko doseže visoko kvaliteto storitev v informacijski tehnologiji. Ogrodje ITIL podaja sistematski pristop k upravljanju IT storitev, od analize, razvoja, testiranja, implementacije in uporabe do neprestanega izboljševanja. Procesi, ki jih ITIL definira, so zasnovani tako splošno, da so neodvisni od platforme in dejanskih storitev, a se vseeno lahko uporabljajo na vseh področjih informacijske tehnologije. Od sredine devetdesetih let prejšnjega stoletja naprej je ITIL v svetovnem merilu de facto standard pri upravljanju IT storitev.

V sklopu knjig, izdanih s strani OGC (*Office of Government Commerce*), lastnika blagovne znamke ITIL, niso vključene prakse s področja vodenja projektov. Priznavajo pa, da je uspešno vodenje projektov ključno za uspešno uvajanje ITIL-a.

Zbirka knjig v prvi verziji ITIL-a, vsaka je pokrivala specifično področje upravljanja IT storitev, je kmalu narasla na več kot 30 knjig. Da bi zbirko naredili bolj dostopno so v verziji 2 združili sorodne tematike pod isto streho in jih seveda dopolnili. Dve najbolj uporabljani knjigi druge verzije ITIL-a sta **Service Support** in **Service Delivery**. Skupaj z ostalimi, čeprav manj znanimi knjigami, pa je ITIL že v drugi verziji prinesel celovit nabor dobrih praks in povezal tehnično implementacijo ter operativno rabo s strateškim in finančnim managementom modernega poslovanja.

Vzdrževanje storitev (Service Support) se osredotoča na uporabnika IT storitev. Primarna skrb vzdrževanja je zagotavljanje uporabnosti storitve za izvajanje poslovnih funkcij. Uporabniki so z vzdrževanjem storitve povezani, saj:

- zahtevajo spremembe,
- prijavljajo težave s storitvijo,
- imajo vprašanja, povezana s storitvijo.

Storitveni center (Service Desk) je kontaktna točka med uporabniki in ponudnikom storitve. Center beleži in rešuje težave uporabnikov, poizkuša rešiti kar je v njegovi domeni oz. zabeleži incident, v kolikor se težave ne da odpraviti. Druga funkcija storitvenega centra je obveščanje uporabnikov o stanju nerešenih incidentov. Incidenti sprožijo verigo procesov: upravljanje incidentov, problemov, sprememb, izdaj in konfiguracij.

Upravljanje incidentov (Incident Management) skrbi za čimprejšnjo vzpostavitev normalnega stanja uporabnosti storitve, da bi tako preprečili poslovno škodo, ki bi pri tem lahko nastala. Skrbi torej za ohranjanje najboljšega možnega nivoja kvalitete in razpoložljivosti storitve.

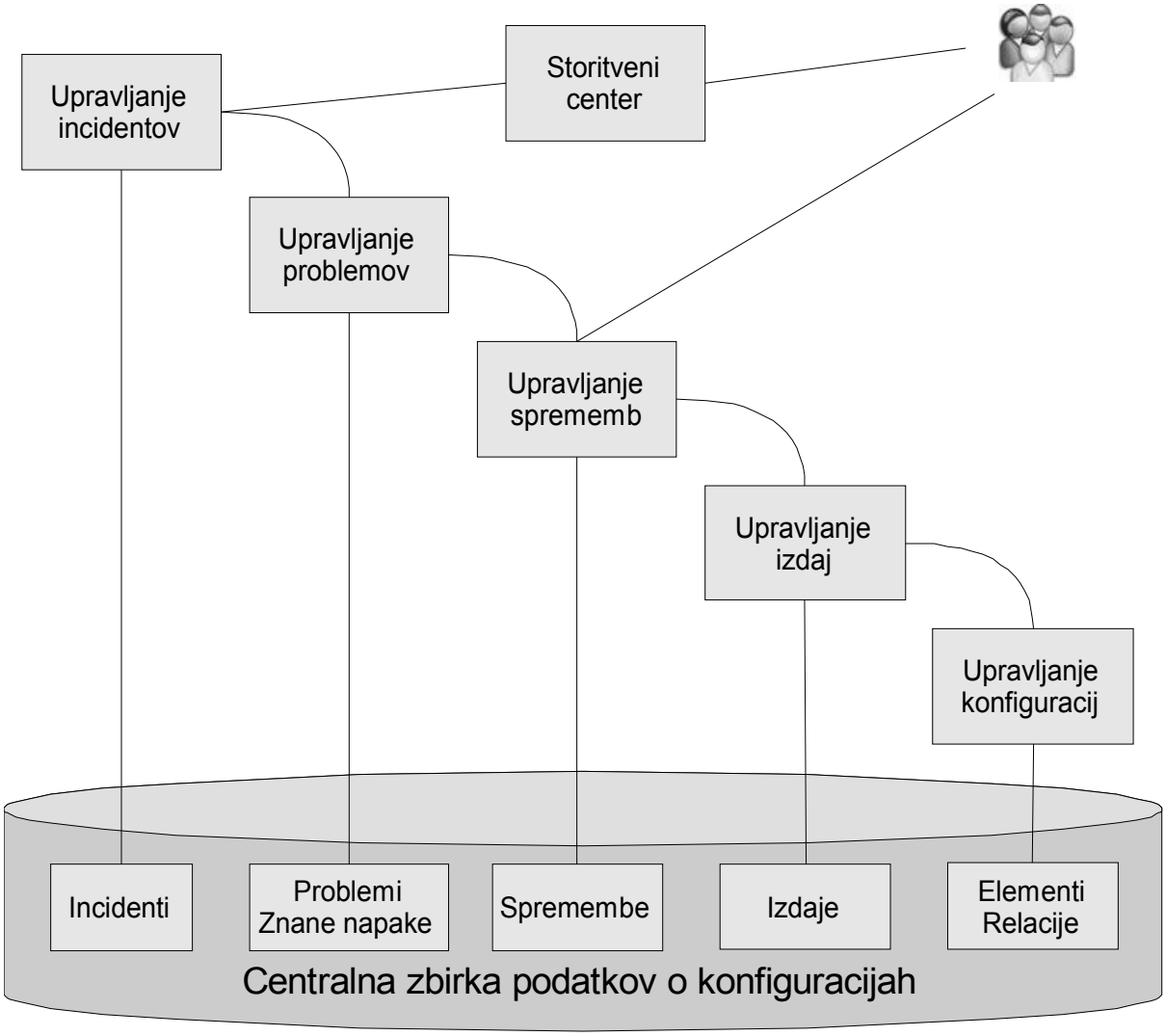
Upravljanje problemov (Problem Management) je zadolženo za iskanje in odpravo vzrokov zaradi katerih se incidenti pojavljajo, da bi tako preprečili nadaljnje incidente. Z upravljanjem problemov povezani pojmi so: problem, napaka in rešitev. Problem je neznan vzrok enega ali več incidentov. Znana napaka je problem, ki je uspešno diagnosticiran in zanj poznamo kratkotrajno, nadomestno ali dolgoročno rešitev.

Upravljanje sprememb (Change Management) definira in standardizira metode in postopke, ki se nato uporabljajo za učinkovito obvladovanje vseh sprememb. Sprememba je dogodek, ki povzroči novo verzijo enega ali več elementov konfiguracije.

Upravljanje izdaj (Release Management) se ukvarja z distribucijo programske in strojne opreme preko celotne IT infrastrukture. Ustrezni postopki kontrole zagotavljajo dostopnost licencirane, preverjene in odobrene programske in strojne opreme, ki bo po vključitvi v obstoječo infrastrukturo delovala po predvidenih načrtih. Ker to področje zajema obravnavano tematiko, ga bom bolj podrobno razdelal v naslednjem podpoglavju.

Upravljanje konfiguracij (Configuration Management) je ogrodje, ki omogoča sledenje vsem elementom konfiguracije nekega sistema. Vsi prej našteti procesi uporabljajo centralno zbirko podatkov o konfiguracijah, kjer se vodijo incidenti, problemi, znane napake, spremembe in izdaje elementov konfiguracije.

Slika 2: ključna področja vzdrževanja storitev.



Vir: ITIL, 2000, str. 295.

Slika 2 prikazuje povezanost ključnih področij vzdrževanja storitev in njihovo odvisnost od centralne zbirke podatkov o konfiguracijah.

Dostava storitev (Service Delivery) je primarno zadolžena za proaktivne in vnaprej gledajoče storitve, ki jih poslovni sistem pričakuje od svojega IT ponudnika, da bi tako uporabnikom IT storitev lahko zagotovili ustrezne storitve. Poslovni sistem kot koristnik IT storitev tukaj predstavlja stranko in ne uporabnika, kot pri vzdrževanju storitev. Dostava storitev se deli na:

- upravljanje ravni storitev (Service Level Management)
- upravljanje zmogljivosti (Capacity Management)
- upravljanje neprekinjenosti delovanja (IT Service Continuity Management)
- upravljanje razpoložljivosti (Availability Management)
- finančno upravljanje informacijskih storitev (Financial Management for IT Services)

2.2.1. Cilji upravljanja izdaj

Izdaja je sestavljena iz novih ali spremenjenih delov programske in/ali strojne opreme, ki so potrebni za implementacijo dogovorjenih sprememb. Upravljanje izdaj se ukvarja z distribucijo programske in strojne opreme preko celotne IT infrastrukture. Ustrezni postopki kontrole zagotavljajo dostopnost licencirane, preverjene in odobrene programske in strojne opreme, ki bo po vključitvi v obstoječo infrastrukturo delovala po predvidenih načrtih. Kontrola kvalitete med razvojem in implementacijo nove strojne in programske opreme je prav tako v domeni upravljanja izdaj, to namreč zagotavlja ustreznost z vidika poslovnih potreb.

Cilji upravljanja izdaj so:

- planiranje in nadzor uvedb programske opreme,
- definiranje postopkov za distribucijo in namestitev sprememb IT sistemov,
- zagotavljanje sledljivosti in varnosti sprememb, nameščanje le ustreznih, potrjenih in preverjenih verzij,
- učinkovita komunikacija in upravljanje pričakovanj uporabnikov med planiranjem in uvedbo novih izdaj,
- kontroliranje distribucije in namestitve sprememb IT sistemov,
- nameščanje novih izdaj v produkcijsko okolje z uporabo kontrolnih mehanizmov upravljanja s konfiguracijami in upravljanja s spremembami,
- beleženje opravljenih namestitev v centralni zbirki podatkov o konfiguracijah.

Upravljanje izdaj se mora osredotočati na varovanje produkcijskega okolja in njegovih storitev, pri čemer se uporabljajo formalno definirani procesi in kontrole. Upravljanje izdaj mora delovati usklajeno z upravljanjem sprememb in upravljanjem konfiguracij, saj lahko le tako zagotovi konsistentnost informacij o spremembah in njihovih izdajah v centralni zbirki podatkov o konfiguracijah.

Financiranje upravljanja izdaj pogosto črpa sredstva iz večjih projektov in ni vključeno v ceno vzdrževanja storitev. Čeprav vzpostavitev upravljanja izdaj zahteva določene stroške, so ti neprimerno nižji od potencialnih stroškov napak zaradi neprimernega planiranja, upravljanja in kontroliranja novih izdaj.

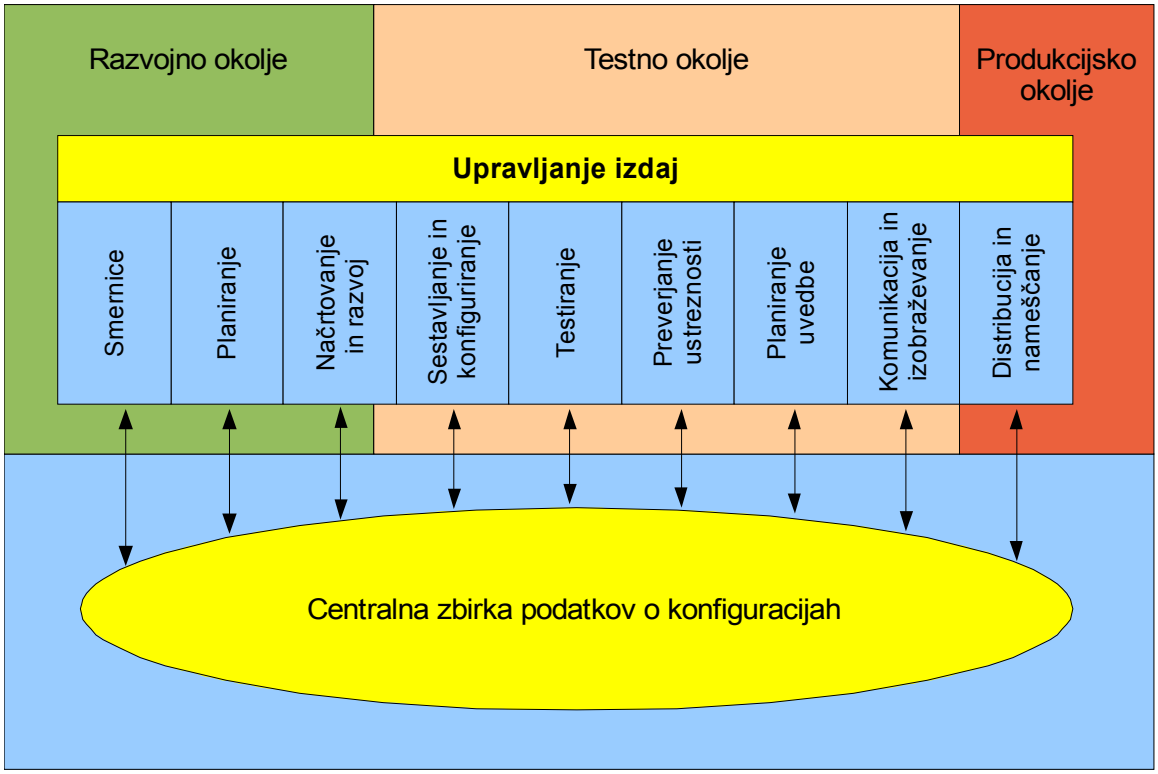
2.2.2. Obseg upravljanja izdaj

Upravljanje izdaj zajema planiranje, načrtovanje, sestavljanje, konfiguriranje in testiranje strojne ter programske opreme, ter v končnem koraku pripravo elementov izdaje za produkcijsko okolje. Aktivnosti zajemajo tudi planiranje, pripravo in razvrščanje uvedb izdaj k uporabnikom. Druge aktivnosti so še:

- definiranje smernic in planiranje izdaj,
- načrtovanje, sestavljanje in konfiguriranje izdaj,
- potrjevanje sprejemljivosti vsebine izdaj,

- načrtovanje nameščanja izdaj,
- intenzivno testiranje do ustreznega nivoja kakovosti,
- potrjevanje ustreznosti izdaje za nameščanje,
- komunikacija, priprava in izobraževanje,
- beleženje strojne in programske opreme pred in po opravljenih spremembah,
- nameščanje nove ali nadgrajene strojne opreme,
- hranjenje programske opreme tako v centraliziranih kot v distribuiranih sistemih,
- izdaja, distribucija in nameščanje programske opreme.

Slika 3: aktivnosti upravljanja izdaj po ITIL v2.



Vir: ITIL, 2000, str. 204.

Slika 3 prikazuje glavne aktivnosti upravljanja izdaj in njihove vloge v življenjskem ciklu sprememb.

Elementi, ki morajo biti upravljani, so:
- aplikacije, razvite znotraj podjetja,
- eksterno razvite aplikacije,
- manjše aplikacije oz. pripomočki,
- sistemska programska oprema zunanjih dobaviteljev,
- strojna oprema in njene specifikacije,
- navodila za namestitev in dokumentacija, vključno z uporabniškimi navodili.

Vsi našteti elementi morajo biti upravljani učinkovito od razvoja ali nakupa, preko konfiguriranja, testiranja in implementacije do njihove uporabe v produkcijskem okolju. Upravljanje izdaj bi se moralo uporabljati pri:

- večjih ali kritičnih nadgradnjah strojne opreme, ali kjer je potrebno sočasno s strojno opremo namestiti tudi spremembe programske opreme,
- večjih izdajah programske opreme, naj bo to prvo nameščanje aplikacije skupaj z ostalo potrebno programsko opremo ali pa pri nadaljnjih vzdrževalnih posegih na tej aplikaciji,
- združevanju logično povezanih sprememb v obvladljive enote.

2.2.3. Koncepti

Izdaja pomeni zbirko potrjenih sprememb določene IT storitve. Izdajo definirajo zahtevki za spremembe, ki jih ta vsebuje. Izdaja je ponavadi sestavljena iz določenega števila popravkov, ki odpravljajo probleme in/ali določenega števila nadgradenj storitve. Izdaja se nanaša tako na programsko kot tudi na strojno opremo.

Moduli programske opreme ter programska in strojna oprema so medsebojno povezani in odvisni drug od drugega. Te odvisnosti lahko vplivajo na sestavljanje primerne izdaje, ker je potrebno zagotoviti koherentnost elementov izdaje. Nova verzija aplikacije npr. potrebuje novo verzijo operacijskega sistema, nov operacijski sistem pa potrebuje hitrejši procesor ali več spomina.

Politika in planiranje izdaj

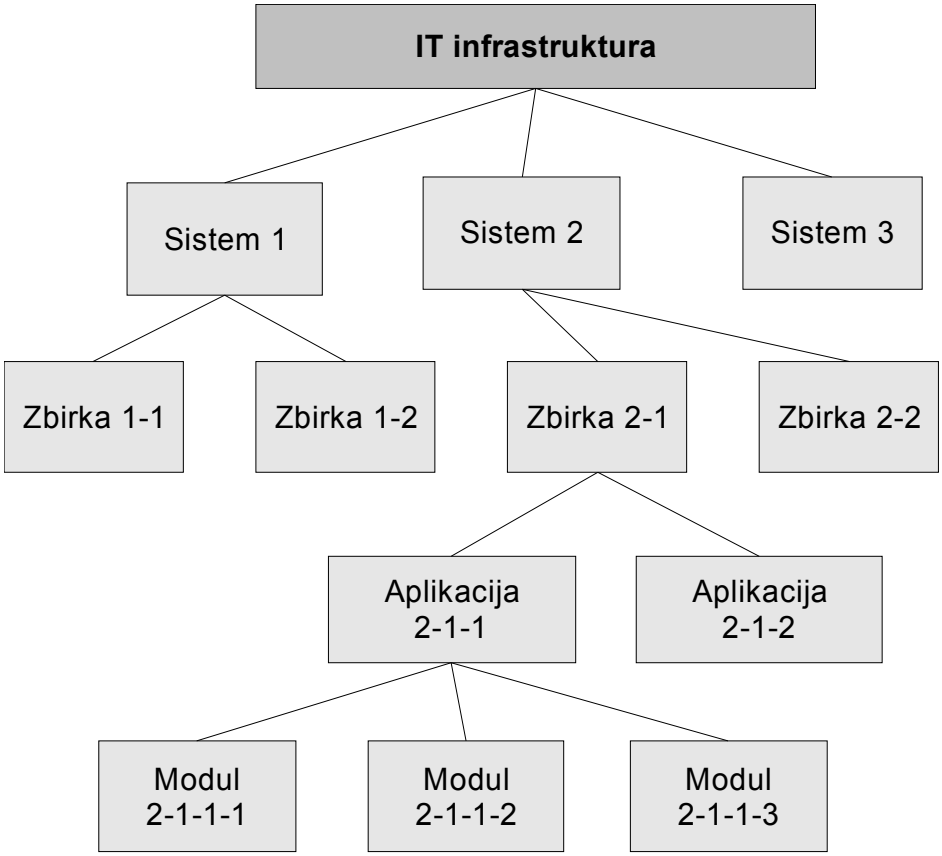
Definirane morajo biti glavne vloge in odgovornosti pri upravljanju izdaj, tako da vsak posameznik razume svojo vlogo in avtoriteto ter vloge ostalih, ki sodelujejo v povezanih procesih.

Smernice morajo vsebovati številčenje izdaj, frekvenco izdaj in nivo IT infrastrukture, ki bo vključen v izdaje. Podjetje se mora glede na velikost in naravo sistema, števila in pogostosti izdaj ter posebnih potreb uporabnikov IT storitev odločiti za najprimernejši pristop k upravljanju izdaj. Vsaka izdaja mora imeti enoličen identifikator, da ga lahko upravljanje konfiguracij uporablja pri identifikaciji določene izdaje.

Enota izdaje

Enota izdaje definira delež IT infrastrukture, ki se navadno namešča hkrati. Enota izdaje se lahko spreminja v odvisnosti od tipa ali samih elementov programske in strojne opreme. Slika 4 kaže poenostavljen primer IT infrastrukture, sestavljene iz sistemov, ki se dalje delijo na zbirke aplikacij, te na aplikacije, te pa na module.

Slika 4: poenostavljen primer IT infrastrukture.



Vir: ITIL, 2000, str. 206.

Potrebno je določiti najprimernejšo enoto izdaje za vsak element ali tip programske opreme. Organizacija se lahko npr. odloči, da bo del svojih storitev izdajala na nivoju sistema, kar pomeni, da bodo spremembe elementov konfiguracije tega sistema povzročile izdajo celotnega sistema. Ista organizacija se lahko odloči, da bo drug del storitev izdajala na nivoju zbirke aplikacij, tretji del storitev pa na nivoju modulov.

Pri določitvi ustrezne enote izdaje naj se upošteva:

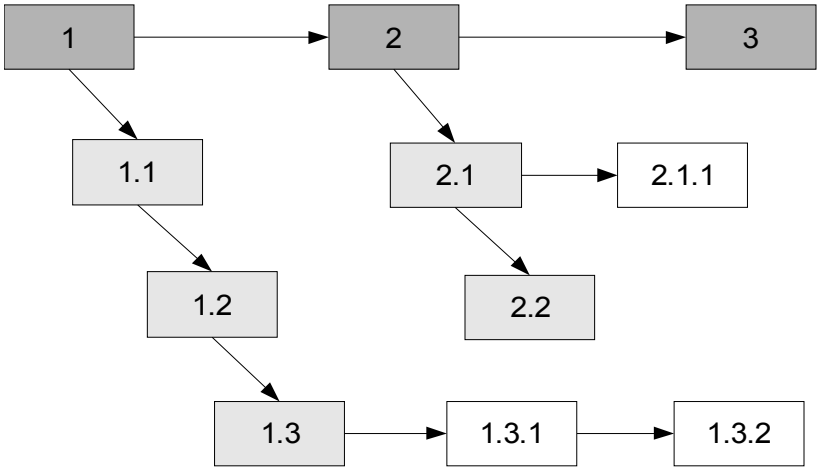
- količina potrebnih sprememb na vsakem nivoju,
- količina sredstev, potrebnih za sestavo, testiranje, distribucijo in implementacijo sprememb na posameznem nivoju,
- enostavnost implementacije,
- kompleksnost vmesnikov med predlagano enoto in ostalo IT infrastrukturo,
- potrebne strojne opreme v razvojnem, testnem in produkcijskem okolju.

Številčenje izdaj

Vsaka izdaja mora imeti enoličen identifikator v skladu s shemo številčenja, definirano v smernicah. Identifikator naj bo sestavljen iz imena elementa konfiguracije, ki ga predstavlja, in številčnega dela, ki označuje verzijo tega elementa. Številčni del je velikokrat sestavljen iz dveh ali treh delov, npr.:

Večja izdaja:	Program_A v.1, Program_A v.2, Program_A v.3
Manjša izdaja:	Program_A v.1.1, Program_A v.1.2, Program_A v.1.3
Izredna izdaja:	Program_A v.1.1.1, Program_A v.1.1.2, Program_A v.1.1.3

Slika 5: primer številčenja izdaj.



Vir: lasten.

Slika 5 prikazuje različne verzije nekega elementa. Z zeleno barvo so označene večje izdaje, z modro barvo manjše izdaje, s sivo pa izredne izdaje.

Tip izdaje

Glede na način nameščanja sprememb delimo izdaje na:

- celotne izdaje,
- delta izdaje,
- paketne izdaje.

Glavna prednost celotnih izdaj je sočasno sestavljanje, testiranje, distribucija in implementacija vseh elementov konfiguracije. Zaradi tega ni bojazni, da bi v izdajo vključili starejšo verzijo kakšnega elementa, za katerega bi napačno mislili, da ni bil spremenjen. Ker je potrebno pretestirati celotno izdajo je manj možnosti, da preskočimo testiranje kakšnega elementa aplikacije, za katerega napačno domnevamo, da ni bil spremenjen oz. odkrijemo napake, ki se pojavljajo v sicer nespremenjenih elementih, ki pa vsebujejo vmesnike do spremenjenih elementov.

Več je torej možnosti, da bomo probleme odkrili pred nameščanjem aplikacije v produkcijsko okolje, čeprav moramo za to plačati ceno v obliki dolgotrajnejšega in zahtevnejšega sestavljanja, testiranja, distribucije in implementacije izdaje. Regresijsko testiranje, ki je navadno del implementacije večje izdaje povzroči ponovno testiranje velikega števila elementov in tako preveri funkcionalnosti oz. njihovo kvaliteto.

Delta izdaja vsebuje le tiste elemente konfiguracije, ki so se spremenili ali na novo nastali od zadnje celotne ali delta izdaje. Če je npr. enota izdaje aplikacija, potem delta izdaja vsebuje le spremenjene module te aplikacije. V nekaterih situacijah je izdaja celotne aplikacije neprimerna, takrat lahko uporabimo delta izdajo. Definirati je potrebno, ali so delta izdaje sploh dovoljene in pod kakšnimi pogoji. Priporočljivo namreč je, da se delta izdaje dovolijo, a se hkrati definirajo pogoji, ki jih mora takšna izdaja zadovoljevati.

V vsakem primeru bi moral odbor za spremembe glede na zbrana dejstva izdati mnenje o naslednji izdaji – ali naj bo to celotna izdaja ali pa delta izdaja. Pri tej odločitvi mora odbor upoštevati:

- velikost delta izdaje v primerjavi s celotno izdajo in s tem povezana sredstva,
- sredstva, ki so na razpolago za sestavljanje, testiranje, distribucijo in implementacijo,
- nujnost potreb po novih funkcionalnostih, ki jih bo izdaja prinesla,
- število spremenjenih elementov od zadnje celotne izdaje – veliko število sprememb je bolje izdati kot celotno izdajo,
- tveganje, povezano z možnimi napakami zaradi omejenega testiranja delta izdaj.

Da bi se z zmanjšanjem števila izdaj produkcijskemu okolju zagotovilo daljša obdobja stabilnega delovanja je priporočljivo združiti posamezne izdaje, kjer je to možno, in kjer je moč obvladati večje število sprememb brez dodatnih incidentov. Takšnemu skupku izdaj pravimo paketna izdaja. Spremembe določenega sistema so lahko odvisne od sprememb v drugih sistemih. Če morajo spremembe biti nameščene sočasno, potem je smiselno takšne izdaje združiti. Paketna izdaja tako lahko vsebuje eno novo storitev in več dograjenih storitev, ki novo storitev uporabljajo.

Paketne izdaje lahko zmanjšajo možnost nekompatibilne programske opreme ter organizaciji zagotovijo, da so spremembe, ki bi morale biti v različnih sistemih nameščene sočasno, dejansko nameščene sočasno, hkrati pa omogočajo testiranje povezav med sistemi in aplikacijami. Paziti pa je potrebno, da se v paketno izdajo zajame le toliko sprememb, kolikor jih je še možno obvladati. Ko se odloča o vsebini izdaje je potrebno poznati in razumeti vpliv vsakega elementa na vse ostale elemente.

Knjižnica veljavne programske opreme

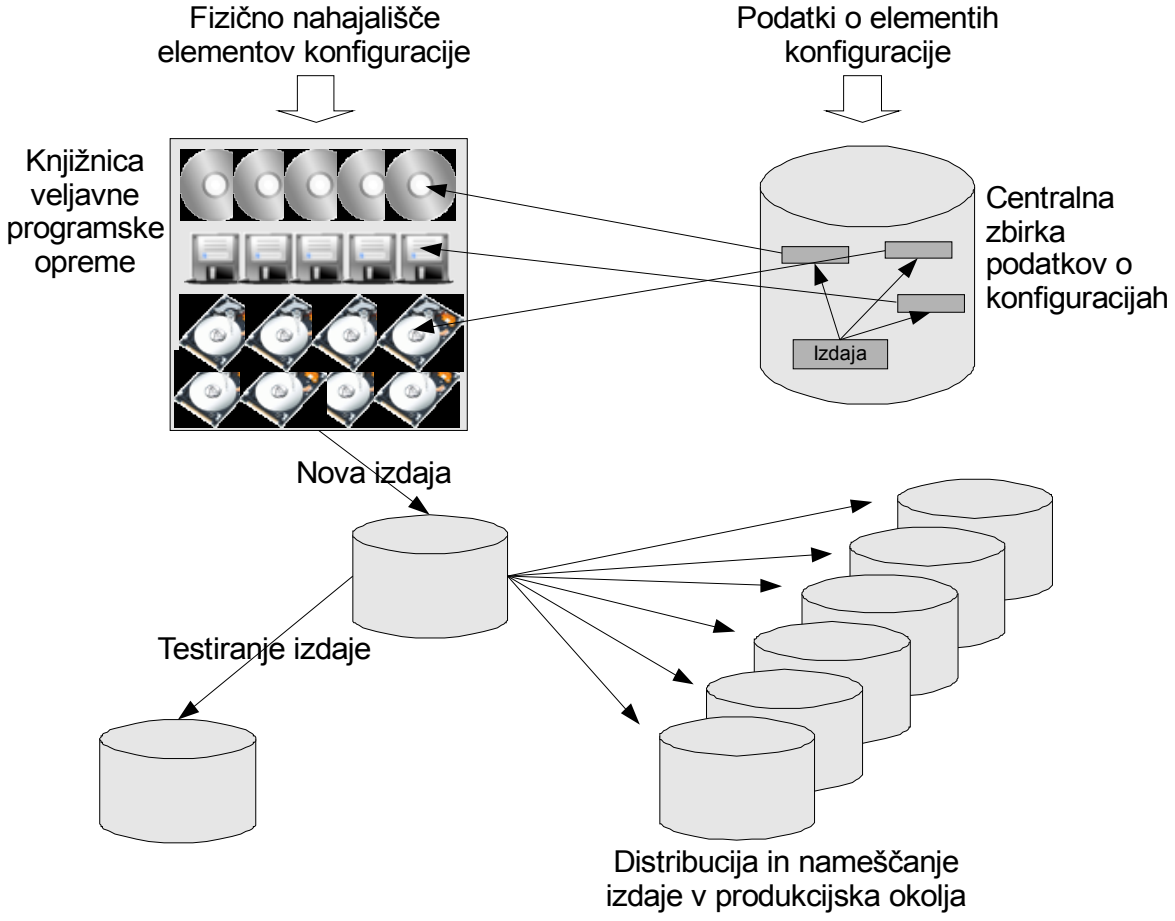
Knjižnica veljavne programske opreme se uporablja kot odlagališče, v katerem so shranjene in zaščitene vse odobrene verzije elementov konfiguracije programske opreme. To odlagališče je lahko dejansko sestavljeno iz ene ali več pomnilniških naprav, ki pa morajo biti

ločene od razvojnega, testnega ali produkcijskega okolja. Shranjena mora biti vsa razvita programska oprema ter kopije kupljene programske opreme skupaj z licencami in dokumentacijo.

Točna vsebina in struktura knjižnice mora biti definirana pred pričetkom razvoja. Definicija knjižnice je del smernic upravljanja izdaj organizacije in mora vsebovati:

- seznam medijev, fizičnih lokacij, strojne in programske opreme, ki se bo uporabljala v ta namen,
- načrt poimenovanja shramb podatkov in fizičnih medijev,
- podprta okolja, npr. testno in produkcijsko,
- varnostno politiko dostopa in spreminjanja elementov, plan izdelave varnostnih kopij in obnavljanja,
- obseg knjižnice, npr. izvorne datoteke, prevedene datoteke, dokumentacija,
- čas hrambe starih izdaj in programske opreme,
- načrt kapacitete knjižnice in postopke za spremljanje naraščanja velikosti,
- postopke revizije,
- postopke, ki zagotavljajo varnost pred napačnimi ali nepooblaščenimi spremembami.

Slika 6: povezava centralne zbirke podatkov o konfiguracijah in knjižnice veljavne programske opreme.



Vir: ITIL, 2000, str. 209.

Slika 6 kaže tesno povezavo med centralno zbirko podatkov o konfiguracijah in knjižnico veljavne programske opreme. V centralni zbirki so informacije o vsakem elementu konfiguracije, ki je del neke izdaje, dejanska vsebina elementa (izvorna koda, prevedena koda, dokumentacija, kupljena programska oprema, licence) pa se nahaja v knjižnici.

Centralna zbirka podatkov o konfiguracijah

Zbirka se osvežuje in uporablja v procesih upravljanja izdaj sočasno s knjižnico veljavne programske opreme. Za podporo procesom upravljanja izdaj mora vsebovati naslednje elemente:

- definiran plan bodočih izdaj, ki ga sestavljajo elementi konfiguracije skupaj z zahtevki za spremembe,
- vodenje informacij o elementih preteklih in bodočih izdaj,
- vodenje ciljnih destinacij izdaj, npr. fizičnih lokacij za strojno opremo in strežnikov za programsko opremo.

Grajenje izdaj

Elementi konfiguracije, ki tvorijo novo izdajo IT storitve morajo biti sestavljeni v celoto na kontroliran in ponovljiv način. Pri grajenju programske opreme je običaj, da se izvorno kodo, dobljeno od razvijalcev, prevede v izvršilne datoteke, torej v obliko, primerno za uporabo in/ali testiranje. Proces upravljanja gradnje programske opreme spada pod upravljanje izdaj.

Pogosta praksa je avtomatizacija teh postopkov zaradi zmanjšanja človeških posegov in posledično večje zanesljivosti. Avtomatizacija, pogosto izvedena s pomočjo dodatnih aplikacij, mora biti kontrolirana kot dodatni element konfiguracije.

Tudi strojna oprema navadno potrebuje sestavljanje in konfiguracijo, kar mora biti opravljeno na kontroliran način, celoten postopek pa skrbno dokumentiran. Pogosta praksa je pisanje skript za avtomatizacijo nameščanja sistemov in aplikacij na strežnike in delovne postaje. Odvisno od plana uvedbe izdaje se lahko nameščanje programske opreme odvije vnaprej, npr. ob nakupu novih delovnih postaj in torej dolgo pred dejansko uporabo teh postaj.

Testiranje

Izdaja mora pred uvedbo v produkcijska okolja prestati številne teste (funkcionalna testiranja, operativna testiranja, testiranja učinkovitosti, integracijska testiranja, testiranja nameščanja) in v končni fazi pridobiti odobrenje uporabnikov, za kar je zadolženo upravljanje sprememb. Nezadostno testiranje je najpogostejši vzrok neuspeha vseh sprememb in izdaj.

Planiranje umikov

Obstajati mora plan umika, ki natančno dokumentira potrebne akcije, ki jih je potrebno izvesti v primeru neuspešne uvedbe izdaje. Sestava plana umika za vsako spremembo je naloga

upravljanja sprememb, vloga upravljanja izdaj pri tem postopku pa je združevanje planov posameznih sprememb določene izdaje v plan umika celotne izdaje.

Obstajata dva pristopa, ki se lahko uporabita:

- Neuspešna uvedba se lahko popolnoma prekliče in tako omogoči povrnitev IT storitve v prejšnje stanje. To je bistvenega pomena pri celotni izdaji in priporočljivo pri delta izdaji.
- Pogojno se lahko dovoli obnovo takšnega obsega storitev, kot je še možno, če se celotne obnove ne da doseči. Ta pristop se lahko uporabi pri delta izdaji, v slučaju da umik izdaje ni praktičen.

Primeri:

- Izdaja se je le delno namestila – ker je zmanjkalo prostora na trdem disku strežnika, se je namestilo le del vseh sprememb. V tem primeru nam plan umika izdaje definira postopke za umik izdaje in povrnitev aplikacije v stanje pred nameščanjem izdaje. To je npr. možno narediti z varnostno kopijo aplikacije ali celotnega strežnika pred nameščanjem izdaje in obnovo aplikacije ali strežnika iz varnostne kopije ob napaki.
- Uvedba izdaje lahko vključuje zamenjavo kritične strojne ali programske opreme (strežnik ali pa operacijski sistem) pri čemer časa za obnovitev prvotnega stanja, v kolikor bo šlo kaj narobe, ne bo na razpolago. Plan v tem primeru lahko definira, da se bo za prizadete storitve začasno uporabljalo drug strežnik.
- Plan umika se lahko uporabi tudi v primeru, da nameščanje sprememb traja dlje od predvidenega časa in lahko vpliva na delovanje storitev. V tem primeru bi plan umika moral vsebovati rok, do katerega bi se morala namestitev izvesti, sicer bi se sprožil umik. Zato je potrebno oceniti čas, potreben za izvedbo umika, tako da se lahko umik izdaje izvrši pravočasno, da uporabniki ne zaznajo motenj v storitvi.

Plan umika je del upravljanja s tveganji in bi moral biti preverjen skupaj s planom uvedbe. Planiranje umikov je odvisno tudi od potreb uporabnikov. Za storitve, ki niso kritične, je dovolj že dogovor, da se v primeru neuspešnih izdaj ročno odpravi težave. Plani umikov bi morali biti preverjeni v procesu testiranja uvedbe izdaj.

2.2.4. Koristi in možni problemi

Pri uvajanju upravljanja izdaj je udeležence potrebno seznaniti s koristmi, ki jih organizacija lahko pričakuje od uspešne uvedbe, hkrati pa predvideti možne probleme in omiliti njihove posledice, oz. z ustreznimi ukrepi poskrbeti, da se ne bodo pojavili.

Koristi

Z uporabo informacijske tehnologije postajajo organizacije vedno bolj odvisne od informacijskih storitev, hkrati pa se veča pomen nadzora in varnosti informacijskih sistemov. Organizacije morajo biti sposobne obvladati številne izdaje in pri tem ohranjati visoko

kvaliteto storitev. Procesi in prakse upravljanja izdaj pomagajo k doseganju ciljev na učinkovit in ekonomičen način. Bistvene koristi upravljanja izdaj, skupaj z učinkovitim upravljanjem konfiguracij, upravljanjem sprememb in operativnim testiranjem so:

- višji odstotek uspešno izdane strojne in programske opreme in torej večja kakovost storitev, ki jo občuti poslovni sistem,
- konsistentnost procesov izdajanja strojne in programske opreme,
- zmanjšano število motenj v storitvah zaradi sinhronizacije izdaj strojne in programske opreme,
- zagotovilo da je uporabljana strojna in programska oprema visoke in poznane kvalitete, saj so izdaje bile sestavljene kontrolirano, kvaliteta izdaj in sprememb je bila preverjena, upravljanje sprememb pa je poskrbelo za ustrezne teste,
- stabilna testna in produkcijska okolja, ker se posamezne spremembe združijo v izdajo in je zato manj individualnih namestitev,
- boljši izkoristek človeških virov zaradi skupnih naporov pri testiranju večjih izdaj – lažje je upravičiti strošek regresijskega testiranja celotnega sistema pri večji izdaji kot pri posameznih manjših spremembah,
- realna pričakovanja zaradi vnaprej znane časovne razporeditve novih izdaj,
- manj napak pri nameščanju izdaj (npr. namestitev neustrezne izdaje) zaradi kontroliranega izdajanja,
- na voljo je seznam vseh sprememb produkcijskega okolja, tako programske kot strojne opreme,
- ustrezen nadzor nad lastnino – strojno in programsko opremo, od katere je organizacija življenjsko odvisna,
- s hkratno uvedbo več sprememb v eni izdaji se poveča sposobnost učinkovitega obvladovanja velikega števila sprememb produkcijskega okolja, brez slabih učinkov na kvaliteto IT storitev,
- sposobnost sestave in centraliziranega nadzora programske opreme v uporabi na oddaljenih lokacijah,
- zmanjšanje stroškov vzdrževanja zaradi sposobnosti ohranjevanja konsistenega stanja programske opreme na številnih lokacijah,
- zmanjšana možnost uporabe nelegalnih kopij programske opreme,
- lažje odkrivanje napačnih verzij in nepooblaščenih kopij programske opreme,
- zmanjšana možnost neopaznega vnosa virusov ali druge škodljive programske opreme,
- hitrejši reakcijski čas med zahtevkom za spremembo, spremembo in izdajo ter manj zakasnitev v procesih,
- zmanjšanje števila izdaj zaradi združevanja sprememb,
- tekoče prehajanje izdaj iz razvojnega v produkcijsko okolje.

S povečanjem učinkovitosti in uspešnosti upravljanja izdaj se poviša tudi produktivnost strokovnjakov, zadolženih za informacijske storitve. Večjo produktivnost občutijo tudi

uporabniki storitev, saj je potrebnih izdaj manj in so boljše načrtovane, višja pa je tudi kvaliteta izobraževanja in dokumentacije.

Možni problemi

Možni problemi pri uvajanju upravljanja izdaj so:

- Odpor in nepripravljenost na spremembe vpletenih oseb. Ljudje so navajeni delati po obstoječih postopkih in jim novi procesi predstavljajo nepotrebno breme. Zato je potrebno razložiti koristi novih postopkov, pri uvajanju sprememb pa sodelovati z vpletenim osebjem.
- Izkušnje so pokazale, da imajo ekipe, ki najbolj potrebujejo učinkovito upravljanje izdaj, najmanj časa za uvedbo le-tega. Zato je potrebno s karseda malo spremembami doseči začetne koristi in jim nuditi podporo pri uvedbi še ostalih sprememb.
- Posamezniki lahko poizkusijo zaobiti procedure upravljanja izdaj, proti čemur se je potrebno odločno boriti, še posebej ko se to izkorišča za nameščanje nepooblaščenih verzij programske opreme, kar pogosto pomeni nameščanje škodljive ali netestirane programske opreme, kar oteži in podraži vzdrževanje informacijskega sistema.
- Posameznike lahko premaga skušnjava in zaobidejo standardne procedure, da bi lahko namestili izredno izdajo, kar mora biti prepovedano in onemogočeno s pomočjo varnostnih pravil.
- Lahko se pojavi nasprotovanje kontroliranemu sestavljanju izdaje v testnem okolju in želja po preprostem kopiranju programske opreme iz razvojnega okolja.
- V porazdeljenih sistemih se lahko pojavijo težave zaradi potrebe po sočasni namestitvi novih verzij programske opreme. Uporaba orodij za avtomatizirano distribucijo lahko premosti te težave.
- Nekateri posamezniki (vključno z vodji informacijskih oddelkov) vidijo v procesih upravljanja izdaj nepotrebno breme in dodatne stroške, čeprav so takšni postopki predpogoj za uspešno in učinkovito spopadanje s spremembami programske opreme.
- Pojavijo se lahko nejasnosti vlog ter odgovornosti med osebjem, ki skrbi za razvoj, in osebjem, ki skrbi za izdaje. Vsem mora biti jasno npr. kdo je zadolžen za elemente konfiguracij v različnih časovnih točkah življenjskega cikla izdaje.
- Nezadostna sredstva, ki so na razpolago za zadovoljivo testiranje, lahko zmanjšajo učinkovitost postopkov. Veliko število različnih variant v produkcijskih okoljih lahko povzroči nepopolno testiranje.
- Če ni na razpolago dovolj strojne in mrežne opreme, potem je nemogoče sestaviti in testirati nove izdaje v dovolj velikem obsegu ali dovolj kratkem časovnem intervalu. Časa mora biti dovolj, tudi če prva iteracija testiranja ne obrodi sadov in je potrebno testiranje ponoviti. Tako uvedba kot umik izdaje morata biti preverjena v testnem okolju.
- Nepoznavanje vsebine izdaje, sestavnih elementov ali načina nameščanja lahko vodi k napakam.
- Testiranje v enem okolju je lahko uspešno, v drugem pa ne, npr. zaradi drugačne infrastrukture ali inicializacijskih parametrov.

- Osebje lahko čuti odpor do preklica izdaje. Zaradi lovljenja rokov lahko vodstveno osebje pritiska k uvedbi nezadostno testirane programske opreme v produkcijsko okolje.
- Obstajajo lahko slaba, nepopolna ali nereprezentativna testna okolja in postopki.

V vsakem primeru pa bi moral načrt upravljanja izdaj skupaj s smernicami in politiko izdaj biti predmet dogovora, poleg tega pa bi se moral periodično preverjati in spreminjati glede na potrebe, da bi tako dosegli pričakovanja in zastavljene cilje.

2.2.5. Planiranje in implementacija

Učinkovitost upravljanja izdaj je odvisna od upravljanja konfiguracij, upravljanja sprememb in operativnega testiranja. Če ta področja še niso prisotna jih je smiselno uvesti hkrati z uvedbo upravljanja izdaj. Planiranje uvedbe upravljanja izdaj mora vsebovati:

- politiko in postopke izdajanja,
- vloge in odgovornosti operativnega osebja ter vodstva upravljanja izdaj,
- orodja, ki bodo uporabljana za učinkovito in avtomatizirano uvedbo izdaj,
- osebje, ki se bo ukvarjalo z upravljanjem izdaj,
- izobraževanje tega osebja,
- smernice pri časovnem razvrščanju izdajnih aktivnosti znotraj organizacije,
- predloge dokumentov kot pomoč pri planiranju izdaj,
- uporabo ustreznih okolij za gradnjo in testiranje izdaj.

Definirati je potrebno procese za:

- načrtovanje, grajenje in uvedbo izdaj,
- distribucijo in nameščanje izdaj, vključujoč nadzor in vzdrževanje knjižnice veljavne programske opreme,
- nakup, nameščanje, premeščanje in nadzor programske in strojne opreme,
- upravljanje in uporabo podpornih aplikacij ali storitev,
- nadzor upravljanja izdaj, revizije, obvladovanje tveganj in eskalacijo težav.

Predstavniki IT osebja bi morali preveriti definirane procese. Popravki dokumentacije zaradi sprememb v procesih ali politiki bi morali biti predmet upravljanja sprememb. Zastarele verzije dokumentov je potrebno umakniti iz uporabe.

Politika izdajanja

Organizacijska politika izdajanja z definirano vlogo in odgovornostjo upravljanja izdaj bi morala biti zapisana. Lahko je v obliki enega dokumenta za celotno organizacijo ali pa obstaja več dokumentov, ločeno po sistemih. Politika izdajanja je del organizacijskega plana upravljanja sprememb. Politika se lahko spremeni ali dopolni, ko organizacija sprejme novo tehnično infrastrukturo. Novi postopki upravljanja izdaj se morajo pilotsko preizkusiti pred

dejansko implementacijo, npr. nameščanje verzij programske opreme na novo arhitekturo strojne opreme ali podpora novemu operacijskemu sistemu.

Politika izdajanja naj vključuje:

- enoto izdaje posamezne storitve (npr. celoten sistem ali posamezen modul),
- pravila poimenovanja in številčenja izdaj,
- definicijo večjih in manjših izdaj, hkrati s politiko izdajanja izrednih izdaj,
- frekvenco izdajanja, tako večjih, manjših kot izrednih izdaj (npr. večja izdaja vsake tri mesece za prihodnje leto),
- definicijo časovnih terminov, ko se izdaj ne namešča (npr. ob finančnem zaključevanju poslovnega leta),
- zahtevan spremni material glede na tip izdaje (npr. navodila za namestitev in izdajni zapisnik),
- smernice za dokumentacijo izdaj (npr. katera orodja naj se zato uporabijo),
- smernice za izdelavo in testiranje postopkov umika izdaje,
- dogovorjeno vlogo upravljanja izdaj pri tehnični reviziji arhitekture in implementacije aplikacije,
- definirane nadzorne procese, npr. revizijske sestanke, spremljanje napredka, kontrolne točke, eskalacijo težav, analize vpliva sprememb,
- dokumentacijo o konfiguraciji knjižnice veljavne programske opreme in meril za sprejem nove ali spremenjene programske opreme.

Primer pravil za sprejem programske opreme:

- vsa kupljena programska oprema je licencirana,
- programski paketi ne vsebujejo škodljive programske opreme,
- razvita programska oprema je uspešno prestala dokumentirano revizijo kvalitete,
- spremembe elementov konfiguracije so bile avtorizirane s strani upravljanja sprememb in ustrezno označene v centralni zbirki podatkov o konfiguracijah.

Planiranje uvedbe izdaj

Postopki, predloge in smernice bi morale biti vzpostavljene, da bi lahko osebje upravljanja izdaj uspešno in učinkovito izdalo programsko in strojno opremo. Dokumentacija naj pripomore k:

- razumevanju standardov organizacijske infrastrukture in zahtev upravljanja izdaj,
- razumevanju upravljanja storitev in operativnih zahtev, vključno z navodili ravnanja ob napakah, odvisnostih in drugih operativnih zahtevah,
- usmerjanju načrtovanja, grajenja in konfiguriranja izdaje za vsako platformo,
- razumevanju postopkov odobravanja izdaj, vključno z merili, ki jim mora izdaja ustrezati,
- učenju uporabe predlog za planiranje uvedb izdaj, tako da se vedno izvedejo vse potrebne aktivnosti,

- razumevanju zahtev po komunikaciji, pripravi in izobraževanju,
- razumevanju postopkov distribucije in nameščanja.

Izdaja in distribucija programske opreme

Standardni postopki za distribucijo programske opreme iz testnega v produkcijsko okolje bi morali biti definirani. Kjer testno in produkcijsko okolje nista na istem računalniku, je potrebno uporabiti medij za prenos podatkov, npr. mrežno povezavo ali nek prenosni medij (CD-ROM, magnetni trak, trdi disk) pri tem pa poskrbeti za zmanjšanje možnosti sprememb programske opreme med prenosom. Poleg samega prenosa je v ciljnem okolju navadno potrebno izvršiti dodatne procese, da se dostavljena programska oprema lahko prične uporabljati. To lahko vključuje npr. arhiviranje starih verzij, nalaganje novih podatkov v bazo podatkov, skoraj vedno pa je potrebno opraviti neko vrsto preklopne akcije, kot npr. prepis knjižnice ali ponoven zagon aplikacije.

Če je izdajo potrebno distribuirati na veliko število lokacij je navadno zahtevano, da se namestitev izvede sočasno za vse uporabnike na vseh lokacijah. To navadno povzroči nekaj dodatnih zapletov, zaradi omejene hitrosti mrežnih povezav moramo npr. distribucijo opraviti nekaj dni vnaprej, nato pa na vseh lokacijah ob določenem časovnem terminu pričnemo z nameščanjem nove verzije. Takšen način nameščanja mora prestati intenzivna planiranja in testiranja.

Vse potrebne skripte in parametre, ki so potrebni pri nameščanju, moramo nadzorovati tako kot same elemente konfiguracije. Za cilj si moramo postaviti nameščanje, ki bo enostavno, varno in odporno na napake. Idealno je, če lahko osebje upravljanja izdaj po uspešnem zaključku distribucije z enim samim ukazom sproži proces nameščanja. Osebje mora dobiti povratno informacijo o uspešnem zaključku nameščanja. Če procesa distribucije in nameščanja ni možno avtomatizirati in nadzorovati z orodji, morajo ročni postopki zagotoviti, da je izdaja distribuirana ob pričakovanem času, da je pregledana na praktičen način in da je nameščena v predvidenih časovnih terminih. V centralni zbirki podatkov o konfiguracijah mora biti zapisano, katera okolja bodo deležna katerih izdaj. Po uspešni namestitvi v vsako posamezno okolje je to potrebno zabeležiti v centralni zbirki.

Upravljanje sprememb priporoča plan umika za vsako posamezno spremembo. V primeru izdaje to navadno pomeni obnovitev stanja, kot je bilo ob predhodno nameščeni izdaji. Postopki umika in obnove stanja pred nameščanjem morajo biti preverjeni in dokazano delujoči.

Vloge in odgovornosti

Izdaje, ki so sestavljene iz različnih tipov programske in strojne opreme, navadno vključujejo veliko ljudi v procesu izdaje in nadzora. Tipične odgovornosti za sprejem elementov konfiguracije v izdajo bi morale biti centralno definirane in po potrebi prilagojene za

določeno izdajo. Tipične vloge so upravitelj sprememb, upravitelj izdaj in upravitelj testiranja. Kjer obstajajo postopki grajenja in nameščanja, ki so standardizirani in vnaprej odobreni, se lahko aktivnosti odvijejo brez posega upravitelja sprememb, npr. ob nameščanju standardne pisarniške programske opreme na novo delovno postajo.

V nekaterih organizacijah je potrebno vloge definirati za vsak tip elementa konfiguracije posebej, kot je to prikazano v tabeli 1.

Tabela 1: primer vlog glede na tip elementa konfiguracije.

Element konfiguracije	**Izdan od**	**Sprejet s strani**	**Pooblastilo za izdajo v produkcijo**	**Vzdrževalec**
Strežnik	Sestavljalec strežnikov	Upravitelj strežnikov	Upravitelj sprememb	Upravitelj strežnikov
Modul aplikacije	Upravitelj razvoja	Upravitelj testiranja	Upravitelj sprememb	Storitveni center
Izdaja aplikacije	Upravitelj razvoja	Upravitelj testiranja	Upravitelj sprememb, Upravitelj izdaj, Upravitelj testiranja, Storitveni center, Uporabniki	Storitveni center

Vir: ITIL, 2000, str. 219.

Osebje

Število osebja je odvisno od zahtevnosti definiranih postopkov. Ker je upravljanje izdaj na kritični poti med razvojem programske opreme in njeno uporabo ter sodeluje pri vsaki pomembnejši spremembi produkcijskega okolja, mora organizacija poskrbeti, da je na voljo dovolj strokovnega osebja, upoštevajoč letne dopuste, izobraževanja in druge odsotnosti.

V manjših organizacijah lahko ena oseba zaseda več vlog, npr. upravitelj izdaj je lahko hkrati upravitelj sprememb in konfiguracij. V večjih organizacijah je zaradi obsega dela smiselno vloge razdeliti med več ljudi, ponekod pa so potrebne celo večje ekipe za obvladovanje posameznega področja.

Upravitelji izdaj potrebujejo tehnična znanja, poznati morajo IT infrastrukturo in storitve, ki jih organizacija podpira. Seznanjeni morajo biti s pripomočki in orodji, zaradi sodelovanja s povezanimi področji je bistveno poznavanje osnov in praks upravljanja sprememb in upravljanja konfiguracij. Zaradi usklajevanja prioritet morajo poznati organizacijsko poslovno strategijo, zaželene so tudi sposobnosti vodenja osebja in projektov.

Izobraževanje

Upravitelji izdaj bi morali biti izobraženi o principih in praksah upravljanja sprememb ter upravljanja konfiguracij, o vzdrževanju in razvoju programske opreme ter o uporabi ustreznih

orodij in pripomočkov. Seznanjeni bi morali biti z IT infrastrukturo, s storitvami, ki jih vzdržujejo, ter s poslovnimi potrebami, ki jih storitve zadovoljujejo.

Upravitelji razvoja in vodje projektov bi morali poznati politiko, postopke in orodja izdajanja. Postopki in smernice bi morale biti objavljene in vedno na voljo osebju, ki je kakorkoli povezano z izdajanjem programske ali strojne opreme.

Izobraziti je potrebno tudi osebje, ki je zadolženo za nameščanje izdaj v produkcijsko okolje in končne uporabnike, ki storitve uporabljajo.

Uvajanje upravljanja izdaj

Ko so ustrezno osebje, postopki, strojna oprema in pripomočki na voljo in so zaključena vsa potrebna izobraževanja, je potrebno ustvariti knjižnico veljavne programske opreme ter postaviti okolje za gradnjo izdaj, hkrati pa poskrbeti, da lahko do teh okolij dostopa le pooblaščeno osebje upravljanja izdaj. Knjižnica veljavne programske opreme, okolje za gradnjo in okolje za testiranje morajo biti preverjeni v skladu z definiranimi kriteriji.

Razvojni inženirji morajo biti seznanjeni s časovnim terminom, ko lahko pričnejo z vključevanjem elementov konfiguracije v knjižnico veljavne programske opreme. Poskrbeti je potrebno za zavedbo vse kupljene programske opreme v knjižnici in centralni zbirki podatkov o konfiguracijah.

Grajenje in distribucijo izdaj je pred uporabo potrebno preveriti. Ob prvem grajenju in nameščanju izdaj se lahko odkrije še kakšna težava, zato je priporočljivo prvo nameščanje opraviti v testno okolje, tako da se lahko napake odpravijo pred poseganjem v produkcijsko okolje.

Obstajati mora tudi zasilni plan, za primere ko novi postopki odpovejo. Če je izdaje nujno potrebno namestiti, se lahko začasno preide na uporabo prejšnjih postopkov nameščanja, dokler se napake novih postopkov ne odpravi. Priporočljivo je ločiti postopke distribuiranja programske opreme od postopkov nameščanja programske opreme, saj je tako navadno lažje izolirati in odpraviti morebitne probleme.

Čeprav so bili postopki podvrženi intenzivnim testiranjem pred uporabo v produkcijskem okolju, se vseeno lahko pojavijo nepredvidene težave ali problemi, ki so se pri testiranju izmuznili, zato je smiselno planirati nekaj časa za odpravo teh napak.

Da bi se zagotovilo uspešno osvajanje novih postopkov, bi upravitelj izdaj moral:

- zagovarjati nove postopke in vzpodbujati vodje projektov ter razvojne ekipe k uporabi le teh,
- spremljati uvedbo in oglaševati koristi novih postopkov,
- spremljati količino potrebne pomoči pri uvedbi,

- poročati o kršenju organizacijske politike izdaj,
- poročati ali sodelovati s projektno pisarno,
- skrbeti za avtomatizacijo postopkov, kjer je le možno,
- skrbeti, da so uporabniki, storitveni center in osebje upravljanja izdaj ustrezno izobraženi, da imajo potrebno dokumentacijo in dodatna pojasnila.

Stroški

Stroški pri uvedbi upravljanja izdaj so posledica:
- razvoja in definicije praks ter postopkov upravljanja izdaj,
- izobraževanja o politiki izdaj, postopkih in orodjih,
- osebja, potrebnega za razvoj in uporabo postopkov in orodij,
- strojne opreme, potrebne za knjižnico veljavne programske opreme ter za okolja za gradnjo, testiranje, distribucijo in arhiviranje programske opreme,
- programske opreme, potrebne za knjižnico in omenjena okolja,
- avtomatizacije postopkov,
- operativnih stroškov, ki so lahko na začetku nekoliko višji, dokler se osebje še privaja.

V skoraj vseh primerih so stroški, povezani z uvedbo upravljanja izdaj, zanemarljivi v primerjavi s koristmi, ki jih upravljanje izdaj prinaša. Organizacija ne more uspešno delovati, dokler ni sposobna obvladati velikega števila sprememb strojne in programske opreme brez žrtvovanja kakovosti. Upravljanje izdaj pa ji to omogoča.

Brez zadostnega nadzora so organizacije izpostavljene tveganjem: računalniškim prevaram, nenamernim napakam v programski opremi, virusom in ostali škodljivi programski opremi. Odpravljanje tako povzročene škode lahko povzroči neprimerno večje stroške v primerjavi z uvedbo upravljanja izdaj, nekaterih stvari, kot je ugled podjetja, pa niti ni možno obnoviti.

2.2.6. Aktivnosti

Glavne aktivnosti upravljanja izdaj so podrobno razdelane v naslednjih podpoglavjih.

Planiranje izdaj

Planiranje izdaj vključuje naslednje aktivnosti:
- pridobivanje soglasij za vsebino izdaj,
- dogovarjanje o času in načinu uvedbe po posameznih geografskih lokacijah, poslovnih enotah in strankah,
- izdelava okvirnega časovnega plana izdaje,
- analiza posameznih lokacij in ocena uporabljane strojne in programske opreme,
- planiranje potrebnih sredstev,
- definiranje vlog in odgovornosti,
- dogovarjanje z dobavitelji glede nove strojne opreme, programske opreme in storitev,
- priprava plana umika izdaje,

- priprava plana nadzora kvalitete izdaje,
- priprava kriterijev za sprejem izdaje s strani testne skupine in uporabnikov.

Vhodi v aktivnosti vključujejo življenjski cikel projekta, avtorizirane zahtevke za spremembe, politiko izdaj, poslovne potrebe, omejitve, odvisnosti, napotke in predloge odbora za spremembe.

Rezultati planiranja so plan določene izdaje, okvirni plan testiranja in kriteriji za sprejem izdaje. Omenjeni dokumenti so navadno del plana upravljanja sprememb dotičnega projekta.

Načrtovanje, gradnja in konfiguriranje izdaje

Obstoječe aktivnosti gradnje izdaj morajo biti analizirane in dokumentirane, da se zagotovi njihova konsistentna uporaba skozi celotno organizacijo. Konfiguracija določene izdaje je lahko sestavljena iz strojne in programske opreme, iz opreme, razvite znotraj organizacije in opreme, razvite izven organizacije ali kupljene opreme, zato morajo obstajati navodila za grajenje takšnih izdaj.

Grajenje običajno sestavljajo aktivnosti, kot so prevajanje kode in povezovanje modulov, razvitih ali kupljenih, v vsakem primeru pa morajo biti moduli zavedeni v knjižnici veljavne programske opreme. Grajenje lahko vključuje tudi spreminjanje baze podatkov, polnjenje baze podatkov s šifranti ali testnimi podatki, nameščanje operacijskega sistema, ipd. Pogosta je uporaba avtomatiziranih skript, ki zagotovijo uspešno nameščanje izdaje, a tudi te morajo imeti plan umika za primer neuspešnega nameščanja.

Za uporabo orodij je potrebno predvideti nakup licenc in izobraževanje osebja. Pri uvedbi nove strojne opreme je potrebno preveriti varnostne in zdravstvene regulative. Za spremembe vzdrževalnih pogodb za strojno in programsko opremo so navadno potrebna pogajanja.

Vsa programska oprema, parametri, testni podatki in okolje, kjer se programska oprema izvaja, mora biti pod nadzorom upravljanja konfiguracij. Nad programsko opremo, ki je predmet izdaje, morajo biti opravljeni testi kvalitete. Celoten potek grajenja mora biti zaveden v centralni zbirki podatkov o konfiguracijah, kar zagotavlja ponovljivost gradnje.

Okvirni plani testiranja izdaje morajo zajeti tudi teste, s katerimi se preveri uspešnost nameščanja izdaje. Če je npr. za neko izdajo spremenjen postopek avtomatskega nameščanja, ga je potrebno preveriti.

Vhodi v aktivnost vključujejo definicijo in plane izdaje.

Rezultati aktivnosti so:
- natančna navodila grajenja izdaje, vključujoč zaporedja potrebnih korakov,
- ukazi za nakup strojne in programske opreme,

- avtomatizirane namestitvene skripte s plani testiranja.

Potrjevanje ustreznosti izdaj

Testiranje izdaje mora biti opravljeno neodvisno od razvoja strojne ali programske opreme. Preveriti je potrebno funkcionalnost programske opreme, postopke nameščanja ter postopke umika izdaje. S tem zagotovimo tako uspešnost nameščanja, kot tudi funkcionalno uspešnost izdaje. Potrditi ali zavrniti je potrebno ustreznost vsakega koraka testiranja, nato pa te potrditve zavesti v procesih upravljanja sprememb.

Ustreznost izdaj je potrebno preveriti v nadzorovanem testnem okolju, ki ga lahko postavimo v neko znano konfiguracijo programske in strojne opreme. Testno konfiguracijo je potrebno zabeležiti v definicijah izdaje in v centralni zbirki podatkov o konfiguracijah.

Če je ustreznost izdaje negativna, jo je potrebno časovno prerazporediti s postopki upravljanja sprememb. Neustrezne izdaje in vsebovane spremembe je potrebno označiti v centralni zbirki podatkov o konfiguracijah. Spremljati je potrebno število in pogostost neustreznih sprememb ter izdaj in ugotoviti njihov vpliv na sredstva in vire operativnega in podpornega osebja.

Vhodi v aktivnost vključujejo:
- definicijo testnega okolja z vnaprej znano konfiguracijo,
- definicijo in plan izdaje,
- plan testiranja izdaje in kriterije za sprejem ustreznosti izdaje,
- kopije namestitvenih medijev in navodil za namestitev,
- plan testiranja namestitve,
- dokumentirane postopke umika izdaje.

Končni rezultat aktivnosti bi morala biti potrjena ustreznost izdaje in avtorizacija nameščanja.

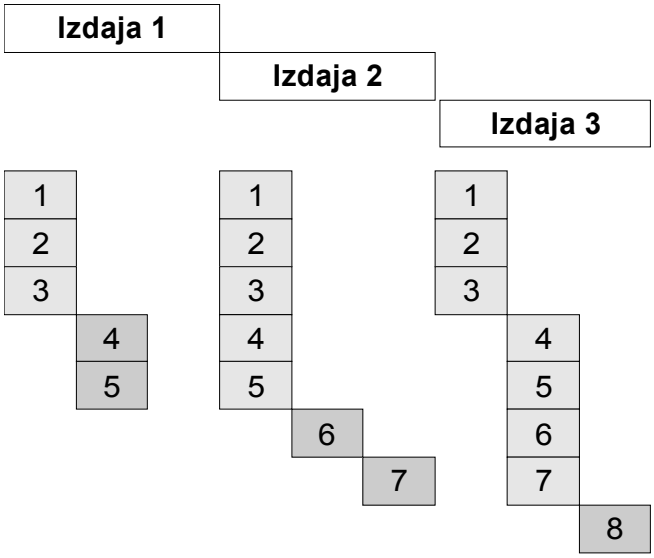
Izhodi iz aktivnosti vključujejo:
- opravljeno testiranje namestitve,
- opravljeno testiranje vsebine izdaje,
- opravljeno testiranje umika,
- rezultate testiranj,
- znane napake, ki jih izdaja vsebuje,
- dokumentacijo kot pomoč pri vzdrževanju izdaje,
- operativna in administrativna navodila,
- časovne termine izobraževanja: uporabnikov, upraviteljev storitev, osebja za podporo,
- dokumentirano ustreznost izdaje, podpisano s strani odgovornih oseb,
- avtorizacijo za nameščanje izdaje.

Planiranje uvedbe

Planiranje uvedbe razširi dosedanji plan izdaje z opisom natančnih aktivnosti uvedbe in plana namestitve. Aktivnost vključuje:

- izdelavo natančne časovne sheme dogodkov in zadolžitev osebja,
- izdelavo seznama vseh elementov konfiguracije, ki jih je potrebno namestiti ali odstraniti, z morebitnim planom uničenja strojne opreme,
- izdelavo časovnega plana uvedbe v produkcijska okolja, pri čemer je za organizacije, ki delujejo globalno, potrebno upoštevati časovna območja, saj so njihove storitve konstantno v uporabi,
- izdelavo izdajnega zapisnika za končne uporabnike,
- planiranje komunikacije,
- izdelavo planov za nakup opreme,
- nakup strojne in programske opreme,
- planiranje sestankov za upravljanje osebja in skupin, povezanih z izdajo.

Slika 7: uvedba izdaje.



Vir: ITIL, 2000, str. 226.

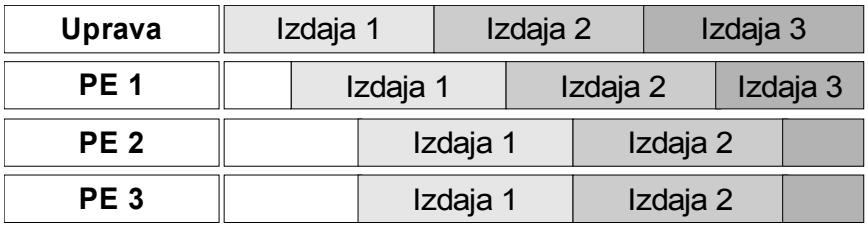
Slika 7 prikazuje tipično zaporedje dogodkov skozi čas za določeno politiko izdajanja, kjer si dogodki sledijo tako:

1. Na tri delovne postaje (1, 2, 3) namestimo prvo izdajo aplikacije.
2. Čez nekaj časa dokupimo še dve delovni postaji (4, 5) in tudi nanje naložimo prvo verzijo.
3. Drugo izdajo aplikacije namestimo na vse obstoječe postaje (1-5) hkrati.
4. V dveh korakih dokupimo še dve delovni postaji (6, 7).
5. Tretjo izdajo uvedemo najprej na tri delovne postaje (1-3), čez nekaj časa pa še na preostale štiri (4-7).
6. Dokupimo še eno delovno postajo (8).

Ta scenarij ilustrira nekaj pomembnih konceptov:

- Izdaje se lahko namestijo hkrati na vse delovne postaje ali pa po korakih. Takšnih postopnih uvedb je dejansko več vrst:
    - Deli funkcionalnosti so lahko v produkcijsko okolje nameščeni postopoma, ampak za vse uporabnike hkrati, npr. postopno spreminjanje centralizirane aplikacije.
    - Celotna izdaja je nameščena postopoma po skupinah uporabnikov, npr. ena geografska enota naenkrat.
    - Najprej namestimo strojno opremo, v naslednjem koraku pa še programsko.
    - Poljubna kombinacija zgoraj naštetih.
- Postopna uvedba, ilustrirana na sliki 7, je drugega tipa glede na zgornji seznam. Takšna uvedba je možna le, če je sistem zasnovan tako, da dopušča hkratno uporabo stare in nove verzije aplikacije. Kjer to ni možno, se uporabi drugačen pristop – hkratno nameščanje vsem uporabnikom.
- V splošnem se je dobro izogniti povečanju tveganja hkratnega nameščanja, kajti v primeru neuspešne namestitve bodo prizadeti vsi uporabniki. Vendar se v nekaterih primerih temu pač ni mogoče izogniti, npr. pri nadgradnji ključne strojne opreme ali nadgradnji operacijskega sistema na kritičnem strežniku.
- Manjše nadgradnje produkcijskega sistema so možne brez potrebe po novih izdajah, npr. dodajanje novih delovnih postaj. To je možno le, ko so spremembe dejansko le dodatni primeri že definiranih elementov določene izdaje. Vendar navadno obstajajo operativne omejitve (npr. največ 10 novih delovnih postaj).

Slika 8: postopno uvajanje glede na geografsko lokacijo.



Vir: ITIL, 2000, str. 227.

Slika 8 prikazuje primer postopnega uvajanja aplikacije na številne geografsko porazdeljene lokacije. Predpostavka je, da vsaka nova verzija sistema lahko deluje hkrati z vsaj eno predhodno verzijo. Na primeru je nova izdaja nameščena najprej v upravi organizacije, potem v pilotski poslovni enoti, nato pa še v ostale poslovne enote. Če je geografskih enot veliko, je potrebno tudi več časa za nameščanje novih izdaj. Hkrati se poveča tudi število izdaj, ki jih je potrebno vzdrževati in zanje nuditi uporabniško podporo.

Komunikacija in izobraževanje

Uporabnike in storitveni center moramo seznaniti s planiranimi izdajami in posledicami, ki jih bodo izdaje imele na njihovo delo. To se navadno doseže preko izobraževanja, vključevanja v

testiranje in v proces potrjevanja ustreznosti. O spremembah, ki jih bo izdaja prinesla, je potrebno pravočasno obvestiti vse udeležence, skupaj z možnimi problemi pri uporabi storitev med potekom nameščanja izdaje. Skliče se lahko enega ali več sestankov, na katerih se predstavi in ovrednoti plan uvedbe ter ga po potrebi spremeni glede na pripombe.

Pomembno je, da se predstavi mehanizem izdajanja in da se uporabnike seznani z možnimi omejitvami (npr.: v veliki organizaciji ni možno čez noč namestiti popravkov na več tisoč delovnih postaj). Spremembe strojne opreme ali pogodb za vzdrževanje programske opreme je potrebno sporočiti ustreznemu osebju. Odgovornost za sporočanje ima storitveni center, zaradi boljšega poznavanja področja pa lahko to opravi tudi upravitelj sprememb.

Vhodi v aktivnost vključujejo:
- definicijo in plan izdaje,
- plan uvedbe izdaje,
- kopije namestitvenih medijev in navodil za namestitev,
- operativna in administrativna navodila,
- trenutna navodila za izobraževanje uporabnikov,
- kriterije za sprejem ustreznosti izdaje in dokument o ustreznosti izdaje.

Izhoda iz aktivnosti sta:
- dopolnjena operativna, administrativna navodila in navodila za izobraževanje,
- dopolnjen plan izdaje.

Distribucija in namestitev

Distribucija izdaj programske opreme iz okolja za gradnjo v testno okolje in nato v produkcijsko okolje bi se morala opraviti hkrati z distribucijo potrebne strojne opreme ali drugih sprememb. Procesi nabave, hranjenja, razpošiljanja, prejema in odstranjevanja strojne opreme morajo zagotavljati varno distribucijo le te. Sprejemne dokumente je potrebno primerjati z dejansko dostavo in to označiti v centralni zbirki podatkov o konfiguracijah. Namestitev ter varnostni pregledi (npr. električni testi, okoljski testi) opreme morajo biti planirani in končani pred vklopom v omrežje.

Distribucija programske opreme mora biti zasnovana tako, da se ohrani integriteta opreme. Z avtomatsko dostavo opreme na oddaljene lokacije lahko privarčujemo sredstva in skrajšamo čas distribucije. Po dostavi na ciljno lokacijo moramo preveriti integriteto izdaje.

Pričetek uporabe nove izdaje v produkcijskem okolju je končni korak vsake spremembe. Dokaj pogosto se nova verzija aplikacije dostavi v ciljno okolje, kjer čaka na trenutek aktivacije, ko se sproži preverjen postopek namestitve. Da se zagotovi uspešno nameščanje je priporočljivo pred nameščanjem še enkrat preveriti dejansko okolje, in ugotoviti ali res zadošča vsem zahtevam. To je dobra kontrola za upravljanje konfiguracij, kajti če do napake

pride v tem koraku, je očitno prišlo do nekonsistentnih informacij v centralni zbirki podatkov o konfiguracijah in je potrebno revidirati postopke upravljanja konfiguracij.

Ob namestitvi ali odstranitvi strojne ali programske opreme je to potrebno zabeležiti v centralni zbirki podatkov o konfiguracijah. Za nekatere izdaje je primerno opraviti še končen test uspešnosti namestitve, uporabnikom lahko ponudimo tudi reševanje ankete, kjer zberemo informacije o njihovem zadovoljstvu.

Vhodi v aktivnost vključujejo:
- plan uvedbe izdaje,
- testirani postopki namestitve,
- testirana vsebina izdaje,
- testiran umik izdaje,

Izhodi iz aktivnosti so:
- nova verzija IT storitve, z novo dokumentacijo in navodili,
- osvežena centralna zbirka podatkov o konfiguracijah,
- seznam odstranjenih elementov konfiguracije (npr. odrabljena strojna oprema),
- znane napake v novi izdaji.

2.2.7. Nadzor nad procesi

Ključni kazalci uspešnosti

Za bi ugotovili učinkovitost upravljanja izdaj je potrebno spremljati številne ključne kazalce uspešnosti. Razmisliti je potrebno o merilih, po katerih je moč objektivno oceniti:
- izdaje so zgrajene in uvedene v predvidenih rokih in s porabo predvidenih sredstev,
- čim manj (idealno nič) mora biti primerov umika izdaj zaradi nesprejemljivih napak (izdaje sicer lahko vsebujejo znane napake, navadno z zanemarljivim vplivom),
- malo primerov neuspešnega grajenja izdaj,
- varno in zanesljivo upravljanje knjižnice veljavne programske opreme,
- vsa programska oprema v knjižnici je prestala kvalitete kontrole,
- kupljena programska oprema zadovoljuje zakonske omejitve,
- časovno ustrezna distribucija izdaj na oddaljene lokacije,
- časovno ustrezna namestitev izdaj na vse lokacije,
- neavtorizirani umiki izdaj se ne smejo pojavljati,
- neavtorizirana uporaba programske opreme se ne sme pojavljati,
- za programsko opremo, ki se ne uporablja, se ne sme izgubljati sredstev za nakup licenc ali vzdrževanja opreme,
- nepotrebno večkratno grajenje ene in iste izdaje se ne sme dogajati,
- natančno in časovno ustrezno beleženje vseh aktivnosti gradnje, distribucije in namestitev v centralni zbirki podatkov o konfiguracijah,
- naknadna analiza vseh izdajnih aktivnosti naj predlaga ustrezne dopolnitve postopkov,

- planirana vsebina izdaje mora ustrezati dejanski vsebini, kar dokazuje dobro planiranje,
- sredstva in vire, potrebne za upravljanje izdaj, se vnaprej planira.

Obveščanje upraviteljev

Drugi kazalniki vključujejo:
- število velikih in manjših izdaj v obdobju poročanja,
- število problemov v produkcijskem okolju, ki jih bodo rešile nove izdaje,
- število novih, spremenjenih in odstranjenih elementov vsake izdaje,
- število izdaj, zaključenih v predvidenih rokih.

2.2.8. Povezava z ostalimi področji ITIL ogrodja

Upravljanje konfiguracij

Ob vsaki novi verziji in ob odstranjevanju programske opreme v knjižnici veljavne programske opreme morajo biti informacije o novi/odstranjeni verziji hkrati zabeležene v centralni zbirki podatkov o konfiguracijah. Ta mora vedno vsebovati ažurne informacije o vsej avtorizirani programski in strojni opremi. Upravljanje izdaj lahko uporablja številne storitve upravljanja konfiguracij med nameščanjem izdaj. Skupaj z upravljanjem sprememb lahko spada pod eno organizacijsko enoto.

Upravljanje sprememb

Ob upoštevanju nasvetov upravljanja izdaj mora odbor za spremembe določiti vsebino in postaviti okvirni časovni plan posamezne izdaje. Upravljanje sprememb je odgovorno za samo implementacijo sprememb. Upravitelj izdaj navadno sodeluje v odboru za spremembe in aktivno sodeluje pri pripravi organizacijske izdajne politike. Čeprav upravljanje izdaj skrbi za uspešno uvedbo sprememb so spremembe pod nadzorom in avtoriteto upravljanja sprememb.

Programska oprema razvijalcev in dobaviteljev

Ko razvijalci ali dobavitelji dostavijo programsko opremo v knjižnico veljavne programske opreme in to zavedejo v centralni zbirki podatkov o konfiguracijah, je upravljanje izdaj zadolženo za gradnjo izdaj in nameščanje v nadzorovano testno okolje. Ko se testiranje uspešno zaključi, lahko upravljanje izdaj nadaljuje z uvedbo izdaje v produkcijska okolja.

Upravljanje problemov in storitveni center

Po uspešni distribuciji in namestitvi nove izdaje je potrebno dopolniti informacije v sistemu upravljanja problemov:
- odpravljeni problemi ali zahteve po izboljšavah morajo biti zaključeni,

- novi problemi, uvedeni z novo izdajo, morajo biti dodani na seznam problemov, da lahko storitveni center nudi ustrezno podporo.

Upravitelji problemov in osebje storitvenega centra morajo biti obveščeni in ustrezno izobraženi o novih izdajah, da lahko zanje uspešno nudijo podporo. Upravitelji problemov navadno sodelujejo v iskanju novih napak, ki vodijo do novih zahtevkov za spremembe in posledično do novih izdaj.

2.2.9. Orodja, specifična za upravljanje izdaj

Orodje za upravljanje sprememb

Upravljanje sprememb navadno uporablja informacijski sistem, kjer se beležijo in planirajo spremembe. Upravljanje izdaj lahko nadgradi ta sistem z informacijami o izdajah in povezavah med izdajami in spremembami, tako da so povezane informacije zbrane na enem mestu.

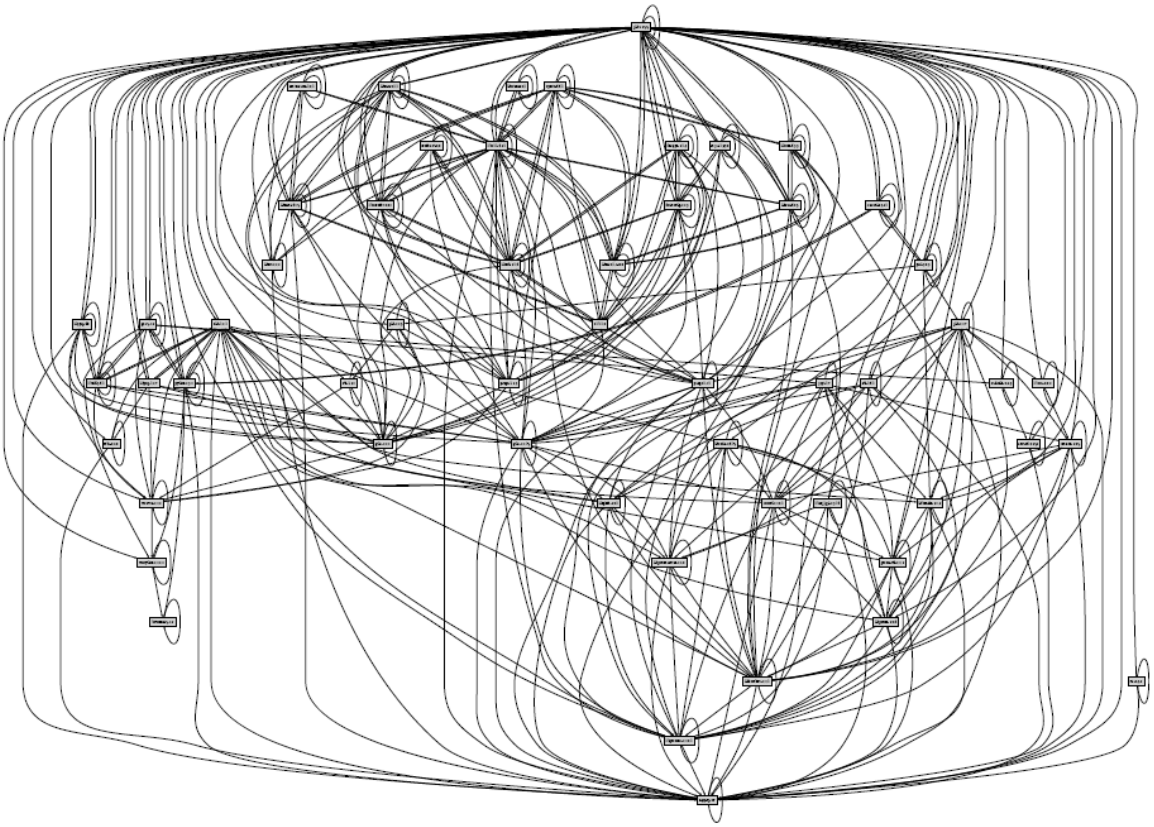
Dober sistem omogoča nadzor nad stanjem posameznih sprememb, izdaj, v katere so spremembe vključene, poleg tega pa nudi možnost avtoriziranja raznih korakov v življenjskem ciklu izdaje. Preko takega sistema se lahko sproži tudi avtomatsko nameščanje v testno in kasneje v produkcijsko okolje.

Orodja za upravljanje konfiguracij

Elementi strojne in programske opreme bi morali biti zavedeni v centralni zbirki podatkov o konfiguracijah. Z napredovanjem izdaje od gradnje, testiranja, distribucije do nameščanja, bi se moralo stanje elementov ažurno označevati v zbirki podatkov.

Obstaja mnogo orodij, s katerimi si lahko pomagamo pri upravljanju različnih verzij izvorne kode med razvojem. Veliko prednost prinašajo orodja, ki obvladajo relacije med elementi konfiguracije. To namreč omogoča analizo vpliva spremembe enega elementa na ostale elemente, s čemer lahko preprečimo določene probleme in planiramo ustrezna testiranja. Relacije med elementi so lahko zelo kompleksne, kar kaže slika 9. Kvadratki predstavljajo posamezen element konfiguracije, krivulje, ki povezujejo kvadratke, pa relacije med elementi.

Slika 9: odvisnost elementov konfiguracije v aplikaciji Mozilla Firefox.



Vir: Dolstra, 2006, str. 18.

Upravljanje izdaj potrebuje tudi orodja, ki obvladajo skupke sprememb, povezane z enim samim zahtevkom za spremembo. Tako orodje mora uporabljati informacije o problemih in spremembah, ki so navadno shranjene v informacijskem sistemu storitvenega centra.

Orodja za upravljanje gradnje

Izdajni proces je bolj učinkovit, če uporablja avtomatsko grajenje izdaj programske opreme. To zahteva avtomatizacijo prevajanja in povezovanja programske opreme v ustreznem vrstnem redu, z uporabo ustreznih verzij izvorne kode, shranjene v knjižnici veljavne programske opreme. Potrebujemo tudi informacije o medsebojni odvisnosti elementov konfiguracije, da bi lahko ob spremembi enega ponovno zgradili odvisne elemente.

Nekatera orodja za upravljanje konfiguracij že vključujejo orodja za grajenje programske opreme, kar omogoča avtomatizacijo grajenja. Ker ta orodja navadno hranijo odvisnosti med elementi konfiguracije, odpade potreba po ročnem vodenju odvisnosti. Če takšnega orodja nimamo na voljo, lahko sami napišemo bolj ali manj enostavne skripte grajenja izdaj. Ko nam informacije o odvisnostih niso na voljo, se moramo zadovoljiti z vsakokratnim grajenjem celotne izdaje.

Orodje bi moralo ob zaključku grajenja dobljene izvršilne datoteke shraniti v knjižnico veljavne programske opreme, v zbirko podatkov o konfiguracijah pa vnesti ustrezne informacije (prstni odtis izvršilnih datotek, podatke o poteku gradnje, uporabljene verzije elementov konfiguracije).

Elektronska distribucija programske opreme

Organizacije uporabljajo številne delovne postaje in strežnike, na katere je v določeni izdaji potrebno namestiti spremembe. To lahko zahteva koordinirano nameščanje sprememb na različne platforme. V procesu distribucije lahko gre kaj narobe, ne glede na to, ali ta poteka ročno ali avtomatsko. Zato je potrebno spremljati potek distribucije in beležiti morebitne napake (npr. nekaj delovnih postaj je bilo izklopljenih iz mreže). Ponekod je potrebno celo umakniti celotno izdajo, če je bila ta le delno uspešna.

Dandanes je vedno težje uspešno in učinkovito dostaviti programsko opremo z ročnimi postopki, zato so se na trgu pričela pojavljati orodja za avtomatsko distribuiranje programske opreme, začenši od enostavnih pripomočkov za prenos datotek do obsežnih modulov, ki so del zbirke orodij za upravljanje večjih sistemov. Uvedba takšnega orodja bi morala biti podrobno analizirana, ker so orodja lahko zahtevna za uporabo in potrebujejo močno strojno opremo.

Sposobnost avtomatiziranja nameščanja programske opreme zahteva razumevanje ročnega postopka in splošnega razumevanja nameščanja in konfiguriranja programske opreme za določeno platformo. Veliko aplikacij že vsebuje namestitvene postopke in včasih lahko že enostavna skripta poskrbi za avtomatiziranje nameščanja.

Orodje za elektronsko distribucijo programske opreme bi moralo podpirati:

- Zagotovljeno dostavo datotek. Dobri sistemi vključujejo preverjanje integritete poslanih podatkov in zmožnost ponovnega pošiljanja od pojava napake naprej.
- Več dostavnih možnosti, ki ustrezno izkoriščajo različne mrežne kapacitete. Nekje je dostava smiselna le preko fizičnega medija (CD ali DVD), nekje se lahko uporabi vmesni strežnik, iz katerega potem vsaka delovna postaja prenese potrebne datoteke.
- Možnost sprožanja namestitve ob vnaprej določenem časovnem trenutku.

Orodja za revizijo strojne in programske opreme

Da bi lahko izdajo uspešno namestili v katerokoli okolje, moramo o njem imeti dovolj informacij. Te lahko zberemo z orodji za revizijo, ki lahko natančno določijo strojno in programsko opremo določenega okolja.

S takšnimi orodji lahko vnaprej preverimo npr. ali je dovolj prostega prostora na disku in ali je nameščena potrebna programska oprema. Če kakšen pogoj ni izpolnjen, lahko reagiramo že pred samo izdajo in tako zmanjšamo število napak pri uvedbi izdaj.

Orodja za upravljanje delovnih postaj

Spreminjanje delovnih postaj s strani uporabnikov je možno omejiti in tako zagotoviti nameščanje izdaj v zanesljiva okolja. Omejevanje se lahko doseže z ustreznimi nastavitvami parametrov operacijskega sistema, ali z namestitvijo dodatne programske opreme. Dobra praksa je, da se nove delovne postaje vedno gradi na enak način, s standardiziranim postopkom nameščanja programske opreme.

Orodja za upravljanje strežnikov

Oddaljen nadzor in oddaljeno diagnosticiranje produkcijskih strežnikov pomaga pri iskanju in odpravi napak med in po uvedbi izdaje. Takšna orodja navadno podpirajo:

- oddaljen nadzor nad delovanjem strežnika, z možnostjo nameščanja sprememb,
- oddaljeno spremljanje seznama napak,
- spremljanje zasedenosti procesorja, pomnilnika in diskovnega sistema,
- upravljanje prostora na diskih, npr. reorganizacija datotek, dodeljevanje prostora aplikacijam.

2.2.10.Vpliv novih tehnologij

Lahki odjemalec

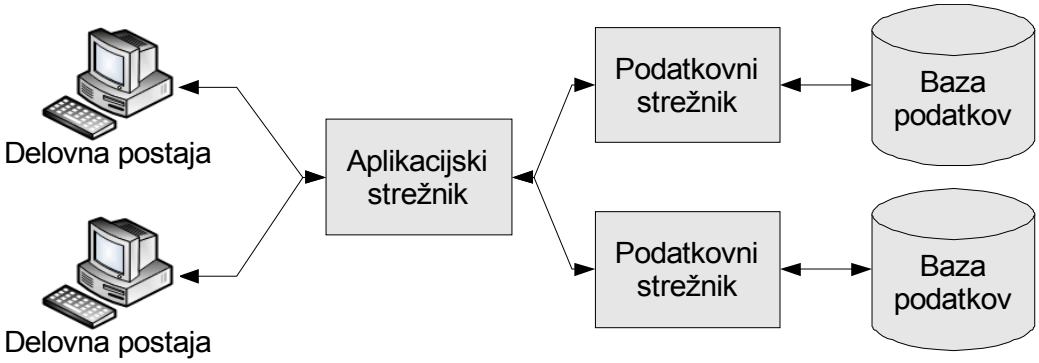
Organizacije, ki se zavestno odločijo zmanjšati količino programske opreme na delovnih postajah, lahko zmanjšajo stroške vzdrževanja. Manj je namreč distribucij programske opreme, hkrati pa lahko gre manj stvari narobe na posameznem odjemalcu. Implementacij lahkega odjemalca je več, vse pa sledijo principu dinamičnega nalaganja le potrebne programske opreme preko mreže, raje kot da je ta nameščena lokalno na vsaki postaji. Ta pristop zmanjša število potrebnih namestitev, saj je novo izdajo potrebno namestiti le na centralni strežnik.

Nekatere implementacije lahkega odjemalca ne potrebujejo nikakršne lokalno nameščene programske opreme, saj se vsa naloži preko mreže. Nekatere pa potrebujejo lokalno nameščen operacijski sistem, programska oprema pa se potem dejansko izvaja na strežniku. Pri taki implementaciji potrebujemo avtomatizirano nameščanje operacijskega sistema.

Večnivojska arhitektura

Vse več informacijskih sistemov je sestavljenih iz programske opreme, ki se izvaja na različnih platformah. Večnivojska arhitektura je navadno sestavljena iz delovnih postaj, aplikacijskih strežnikov in podatkovnih strežnikov.

Slika 10: primer večnivojske arhitekture.



Vir: ITIL, 2000, str. 237.

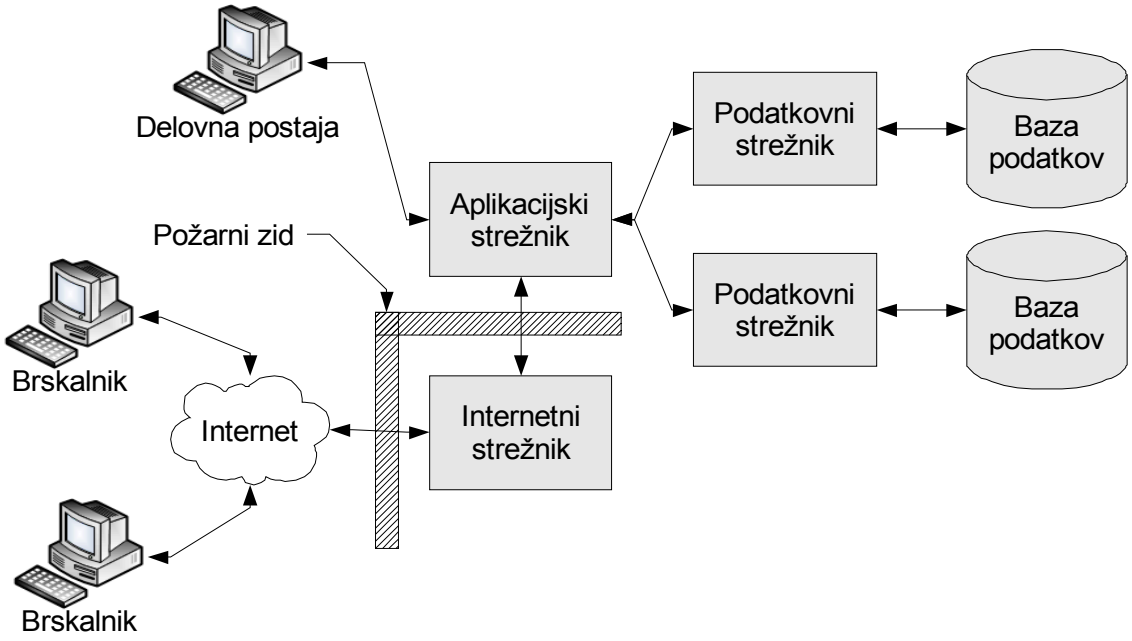
Na sliki 10 je prikazan primer, kjer se več delovnih postaj povezuje na aplikacijski strežnik, ta pa dostopa do podatkov preko dveh različnih podatkovnih strežnikov. Pri nameščanju izdaj v takšen sistem moramo biti sposobni namestiti programsko opremo na vse nivoje: delovne postaje, aplikacijske strežnike in podatkovne strežnike.

Upravljanje izdaj mora zagotoviti koordinirano nameščanje sprememb preko vseh nivojev, hkrati s potrebno zamenjavo strojne opreme.

Internetne aplikacije

Variacija modela večnivojske arhitekture je možnost dodatnega dostopa do organizacijskega sistema preko interneta. Pogosta praksa je uporaba internetnega strežnika, na katerega se uporabniki povežejo preko brskalnika in tako uporabljajo podobne aplikacije kot sicer na delovnih postajah. Internetni strežnik je preko požarnega zidu povezan v organizacijsko notranjo mrežo, kot je prikazano na sliki 11, s čemer se onemogoči neavtoriziran dostop.

Slika 11: primer večnivojske arhitekture z internetnim dostopom.



Vir: ITIL, 2000, str. 237.

S takšno arhitekturo je potrebno nadzorovati še več elementov:

- Povezava z internetom zahteva ustrezne nastavitve usmerjevalnikov in nadzor prepustnosti povezave. Nova verzija aplikacije lahko narekuje dodatno pasovno širino.
- Požarni zid, ki je navadno ločen strežnik, ravno tako zahteva nastavitve in vzdrževanje.
- Internetni strežnik navadno vsebuje spletne strani, do katerih uporabniki dostopajo preko brskalnika. Nova verzija aplikacije navadno zahteva namestitev spremenjenih spletnih strani, lahko pa tudi boljšo strojno opremo.

Razvidno je torej, da ima upravljanje izdaj s pojavom novih tehnologij večjo vlogo, saj mora skrbeti za hkratno nameščanje sprememb na različne sisteme. Dodaten izziv je koordinacija različnih delov IT oddelka pri gradnji in uvajanju izdaj takšnih aplikacij: ekipa za mrežo in požarni zid, strokovnjak za spletni strežnik, razvijalci aplikacije in razvijalci podatkovnega modela. Vloga upravljanja izdaj je pomoč pri usklajevanju teh aktivnosti.

Nadgrajevanje programske opreme preko interneta

Kot medij prenosa novih izdaj komercialne programske opreme se vse bolj uporablja internet. Veliko je tudi celotnih aplikacij, ki se lahko preprosto prenesejo in namestijo. Čeprav se zdi pristop takšnega vzdrževanja aplikacij atraktiven, obstajajo slabosti:

- zaradi velikega števila uporabnikov je onemogočeno hranjenje in nadzorovanje konfiguracij vseh okolij, kjer je programska oprema nameščena,
- možno je, da programska oprema ni bila dovolj testirana, skoraj gotovo pa ni bila regresijsko testirana z ostalimi aplikacijami in tako obstaja možnost napak,

- testna okolja ne odražajo več dejanskega produkcijskega stanja,
- odvisnost od strojne ali druge programske opreme morda ni pravilno ugotovljena,
- programska oprema, prenesena z interneta, lahko vsebuje viruse,
- neavtorizirane spremembe produkcijskega okolja lahko povzročijo napake pri nameščanju izdaj, čemur se lahko izognemo le s temeljitim preverjanjem ciljnega okolja.

Takšno nadgrajevanje je torej dopustno le na način, ki omogoča nadzor nad nameščanjem opreme najprej v testno okolje in nato v produkcijsko, kot del nove izdaje. Uporaba nenadzorovanega avtomatskega nadgrajevanja programske opreme na produkcijskih sistemih je torej odsvetovana.

2.2.11.Napotki za uspešno uvedbo upravljanja izdaj

Sledi nekaj praktičnih nasvetov za enostavnejšo uvedbo upravljanja izdaj. Nasveti so razdeljeni glede na povezane ITIL procese, kar spet kaže na tesno odvisnost procesov. Najboljši nasvet je, da principe upravljanja izdaj uporabimo že pri implementaciji upravljanja izdaj. Postopki upravljanja izdaj naj bodo predmet nadzorovanih sprememb in naj se vodijo s pomočjo upravljanja konfiguracij. Planirane spremembe naj se združijo v eno ali več izdaj skupaj s potrebnim izobraževanjem in dokumentacijo.

Upravljanje konfiguracij

Področje upravljanja konfiguracij mora biti sposobno zagotoviti kakovostne storitve, ki jih upravljanje izdaj potrebuje za izvajanje aktivnosti. Zato je smiselno, da se ob uvedbi upravljanja izdaj vpelje manjkajoče ali dogradi pomanjkljive aktivnosti upravljanja konfiguracij.

1. Avtomatizira naj se revizija strojne in programske opreme, primerja naj se dejansko stanje s stanjem v centralni zbirki podatkov o konfiguracijah.
2. Minimizira naj se število različic, tako strojne kot programske opreme, v produkcijskih okoljih. Vzdrževanje je tako bolj enostavno in zanesljivo. Vodi naj se seznam uradno podprte strojne in programske opreme.
3. Ugotavljanje trenutne konfiguracije nekega sistema mora biti enostavno tudi za uporabnike. Na nivoju sistema mora obstajati podobna funkcionalnost, kot jo imajo aplikacije navadno v menuju »Pomoč – o programu«.
4. Integriteta kritičnih datotek naj se preverja s prstnim odtisom.
5. Izogibati se je potrebno plačevanju podpore za strojno ali programsko opremo, ki ni več v uporabi. Dodatno je z upravljanjem konfiguracij možno preverjati, ali je kje nameščena nelegalna programska oprema. To se lahko preverja ob nameščanju izdaj ali pa na zahtevo.
6. Preko parametrov operacijskega sistema je potrebno uveljaviti strog nadzor nad uporabniškimi delovnimi postajami, da se zmanjša možnost spreminjanja okolij, na katere se namešča nove izdaje.

Upravljanje sprememb

Ker so spremembe glavni razlog, zaradi katerih izdaje sploh potrebujemo, sta ti dve področji tesno povezani. Ob uvedbi upravljanja izdaj je zato potrebno prilagoditi tudi upravljanje sprememb.

1. Preko parametrov programske opreme naj se omeji možnost spreminjanja konfiguracije delovnih postaj.
2. Redno se je treba sestajati in usklajevati z vsemi sodelujočimi v razvoju, implementaciji in vzdrževanju predlaganih sprememb.

Upravljanje izdaj

Uvedba upravljanja izdaj bo uspešnejša, v kolikor bomo upoštevali nekaj ključnih nasvetov.

1. Vzdrževanje programske opreme v produkcijskem okolju mora biti omogočeno s pomočjo izvornih datotek, shranjenih v knjižnici veljavne programske opreme. Dostopne morajo biti vse verzije izvorne kode, hkrati z izvršilnimi datotekami, dobljenimi v postopku grajenja programske opreme.
2. Nove spremembe naj se uvaja preko pilotskih projektov – preizkus na majhnem številu uporabnikov takoj po izobraževanju.
3. Ob zagonu aplikacije naj se avtomatsko ugotovi možne nadgradnje in ponudi njihovo namestitev. Tako zmanjšamo uporabo zastarelih verzij programske opreme.
4. Avtomatizira naj se grajenje, distribucija in nameščanje vseh ali večine programske opreme. V idealnem primeru naj bo omogočeno centralno vodenje takšnih avtomatizmov.
5. Imeti je potrebno na voljo okolja za grajenje programske opreme za specifične platforme, ki jih lahko uporabimo za več projektov ali skupin podobnih aplikacij.
6. Vzdrževati je potrebno ustrezna testna okolja, ki naj bodo čim bolj podobna produkcijski strojni in programski opremi.

Razvoj aplikacij

Zasnova in arhitektura aplikacije lahko imata vpliv na aktivnosti upravljanja izdaj. Pri izbiri možnih rešitev je potrebno upoštevati:

1. Imeti moramo možnost nadzora nad okoljem, kjer je nameščena programska oprema, ne glede ali gre za centraliziran ali porazdeljen sistem.
2. Če je zahtevano, da so spremembe nameščene do določenega roka, potem je smotrno v aplikacijo vgraditi funkcionalnost, ki prepreči njeno uporabo od nekega trenutka naprej.
3. Kjer se programska oprema hkrati izvaja na več sistemih (model odjemalec-strežnik ali n-nivojska arhitektura), moramo zagotoviti njeno konsistentnost. V aplikacijo je možno vgraditi kontrole, ki preverjajo, če so povezani moduli kompatibilni med seboj in sprožiti napako, v kolikor se naleti na zastarelo verzijo.

4. Oceniti je potrebno več možnosti za lokacije podatkov. Nekatere podatke je namreč smiselno voditi centralno na enem mestu, nekatere pa se lahko distribuira/replicira na več lokacij.
5. Pri razvoju aplikacij za mednarodni trg (različne države, različni jeziki, časovna področja) je možno aplikacijo načrtovati na način, ki omogoča, da se od države do države razlikujejo le nastavitve in je torej distribucija programske opreme neodvisna od države.

Pozicija programske opreme: kaj postaviti kam

Razmisliti velja o zmanjšanju količine programske opreme, nameščene na delovnih postajah, in programsko opremo raje namestiti na strežnik. To zmanjša potrebne napore pri distribuciji sprememb, lahko pa povzroči mrežni promet med strežnikom in delovnimi postajami. Ekstremni primer zmanjševanja količine programske opreme je model lahkega odjemalca.

Večina aplikacij potrebuje nekaj datotek, nameščenih na vsaki delovni postaji, navadno v deljenih imenikih. Tudi takšne datoteke moramo imeti pod nadzorom pri nameščanju izdaj, posebno je potrebno paziti na odvisnosti med deljenimi datotekami in drugimi aplikacijami.

Ob hitrem naraščanju kapacitete trdih diskov si dandanes lahko privoščimo nameščanje celotne aplikacije na vsako delovno postajo. Nekatere organizacije pa uporabljajo drugačen pristop – kdorkoli lahko dela na katerikoli delovni postaji. Prednost je recimo ta, da lahko pokvarjeno postajo preprosto zamenjamo z drugo (saj so vse enake). Takšen pristop uporablja kombinacijo tehnik:

- vsi podatki so shranjeni na centralnih strežnikih, nič lokalno,
- dinamično nalaganje potrebne programske opreme preko mreže,
- uporabo pametnega predpomnenja z namenom razbremenitve mrežnih povezav.

Tudi z uporabo naprednih orodij za distribucijo programske opreme je bolj enostavno skrbeti za nekaj strežnikov kot za več tisoč delovnih postaj. Univerzalna rešitev ne obstaja, ampak smiselno je hraniti hitro spreminjajočo programsko opremo na manj lokacijah. Kupljene programske opreme sicer ne moremo spreminjati, nekatere pa podpirajo tudi takšen način nameščanja. Bistveno je, da se organizacija odloči za njej najprimernejšo rešitev.

2.3. COBIT

Ogrodje COBIT (Control Objectives for Information and related Technology) je namenjeno nadzorovanju in obvladovanju informacijske tehnologije (COBIT, 2005). Z upoštevanjem praks COBITa optimiziramo investicije v informacijsko tehnologijo ter zagotovimo dostavo storitev. Ogrodje COBIT je poslovno naravnano, kar se kaže v usklajevanju poslovnih ciljev z IT cilji, v uporabi metrik in zrelostnega modela za merjenje uspešnosti in v deljenju odgovornosti med skrbniki poslovnih in skrbniki informacijskih procesov. Da bi IT oddelek lahko uspešno zadovoljil poslovne potrebe morajo upravitelji:

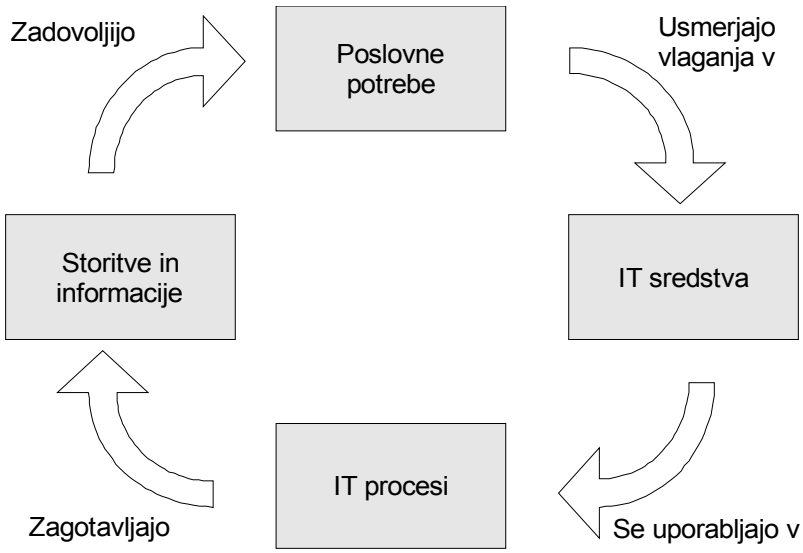
- razumeti poslovne potrebe,
- organizirati aktivnosti glede na splošno priznan procesni model,
- izkoriščati ustrezna sredstva,
- definirati metrike nadzora.

Ogrodje COBIT tem potrebam odgovarja, ker:

- je poslovno usmerjen, saj ni bil načrtovan za uporabo le znotraj IT oddelka, ampak z mislijo na upravitelje in lastnike poslovnih procesov,
- je procesno orientiran, saj vključuje referenčni procesni model in skupno terminologijo,
- temelji na nadzoru, tako da definira politiko, prakse, procedure in organizacijsko strukturo vsakega procesa,
- je metrično usmerjen, ker s pomočjo zrelostnega modela omogoča merjenje učinkovitosti procesov.

Izhodiščni princip COBIT ogrodja je: organizacija naj investira sredstva v informacijsko tehnologijo, ki v strukturiranih procesih zagotovijo ustrezne storitve in informacije, ki jih organizacija potrebuje za svoj uspeh (slika 12).

Slika 12: izhodiščni princip COBITa.



Vir: COBIT, 2005, str. 10.

Procesi so razdeljeni na štiri področja:

1. Načrtovanje in organizacija

   To področje vključuje strateško in taktično načrtovanje, iskanje najboljšega načina s katerim lahko informacijska tehnologija zadovolji poslovne potrebe. Realizacija strateške vizije mora biti načrtovana, predstavljena in upravljana. Vzpostaviti je potrebno ustrezno tehnično in organizacijsko infrastrukturo. To področje se ubada z vprašanji:

   - Sta poslovna in informacijska strategija usklajeni?

- Dosega organizacija optimalni izkoristek sredstev?
- Ali vsak v organizaciji razume informacijske cilje?
- So informacijske nevarnosti znane in nadzorovane?
- Je kvaliteta informacijskih sistemov ustrezna glede na poslovne potrebe?

2. Pridobivanje in izvedba

Da bi prišlo do realizacije informacijske strategije morajo biti informacijske rešitve znane, razvite ali kupljene, implementirane in integrirane v obstoječe procese. To področje pokriva tudi spremembe in vzdrževanje obstoječih sistemov, da zagotovi odzivnost informacijskih rešitev glede na spreminjanje poslovnih ciljev. To področje zanima:

- Bodo projekti zagotovili storitve, ki odgovarjajo poslovnim potrebam?
- Bodo projekti končani v predvidenem roku in s predvidenimi sredstvi?
- Bodo novi sistemi ob uvedbi uspešno delovali?
- Bodo sistemi nemoteno delovali ob spremembah?

3. Dostava in podpora

To področje se ukvarja z dostavo zahtevanih storitev, upravljanjem varnosti in zagotavljanjem neprekinjenosti delovanja ter nudenjem podpore uporabnikom:

- So storitve dostavljene v skladu s poslovnimi prioritetami?
- So stroški storitev optimalni?
- Je možna produktivna in varna uporaba storitev?
- So storitve varne, podatki konsistentni in na razpolago v skladu z varnostno politiko?

4. Merjenje in vrednotenje

Procesi morajo biti redno nadzorovani, saj le tako ugotavljamo odstopanja od načrtovane kvalitete in zahtev. Področje se ukvarja z upravljanjem učinkovitosti, internim nadzorovanjem, predpisi in vodenjem. Odgovarja na vprašanja:

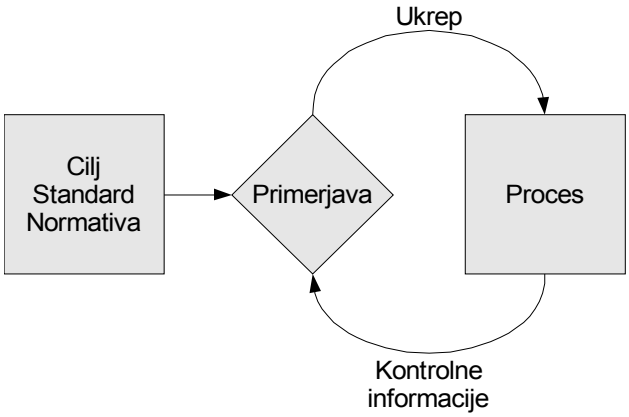
- Je učinkovitost informacijske tehnologije merjena, bodo težave ugotovljene pravočasno?
- Ali upravitelji zagotavljajo uspešnost in učinkovitost internih kontrolnih mehanizmov?
- Se uspešnost informacijskega oddelka odraža na poslovnih ciljih?
- Je varnost, konsistentnost in razpoložljivost storitev pod nadzorom?

Ta štiri področja vsebujejo 34 splošnih informacijskih procesov. Večina organizacij ima definirano načrtovanje, gradnjo, izvajanje in nadzor nad informacijsko tehnologijo, večina organizacij ima zato podobne ključne procese, le malo pa jih ima identično strukturo procesov ali uporablja vseh 34 procesov. COBIT vsebuje celoten nabor procesov, ki se lahko uporabi za preverjanje celovitosti aktivnosti in odgovornosti znotraj organizacije, pri tem pa se lahko procesi odvisno od dejanskih potreb upoštevajo in kombinirajo med seboj. Vsak od 34 procesov je povezan s poslovanjem in zadovoljuje določene IT cilje. Definirane so ključne aktivnosti, cilji, odgovornosti in informacije o merjenju uspešnosti doseganja ciljev.

Nadzor je skupek političnih usmeritev, procedur, praks in organizacijskih struktur, ki omogočajo sprejemljivo zagotavljanje doseganja poslovnih ciljev in preprečujejo pojav neželenih dogodkov ali pa jih ugotovijo ter odpravijo. Kontrolni cilji IT zagotavljajo celoten nabor splošnih zahtev upravljanja za učinkovit nadzor nad vsakim informacijskim procesom. Upravljavci se morajo glede na potrebe odločiti:

- kateri kontrolni cilji so za organizacijo smiselni,
- katere od teh bodo implementirali,
- kako jih bodo implementirali (frekvenca, obseg, avtomatizacija, ipd.),
- sprejemljivost tveganj za neimplementacijo smiselnih ciljev.

Slika 13: kontrolni model.



Vir: COBIT, 2005, str. 14.

Standardni kontrolni model je ilustriran na sliki 13 in je analogen primeru: z nastavitvijo želene sobne temperature (standard) ogrevalni sistem (proces) periodično meri (primerjava) dejansko sobno temperaturo (nadzorne informacije) in ukrepa (akcija) s povečanjem ali zmanjšanjem gretja.

Osnovna potreba vsake organizacije je razumevanje stanja svojih informacijskih sistemov. Pri definiranju nivoja upravljanja in kontrole se je potrebno vprašati kakšen obseg želimo definirati in ali koristi upravičujejo stroške. Dobiti objektiven prikaz nivoja lastne učinkovitosti ni lahko. Kaj je potrebno meriti in kako? Organizacija mora izmeriti svojo pozicijo, ugotoviti, kje bodo potrebne izboljšave, in jih nato nadzorovano implementirati. COBIT pri tem pomaga, saj vsebuje:

- zrelostne modele, ki omogočajo primerjavo in identifikacijo potrebnih izboljšav,
- učinkovitostne cilje in metrike informacijskih procesov, s katerimi ugotovimo kako učinkoviti so procesi in v kolikšni meri procesi zadovoljujejo cilje,
- aktivnostne cilje za učinkovit in uspešen nadzor.

2.3.1. Upravljanje izdaj

Upravljanje izdaj je obdelano v poglavju AI7 – Nameščanje in avtorizacija rešitev in sprememb (angl. Install and Accredit Solutions and Changes): po končanem razvoju je

potrebno sistem spraviti v operativno rabo, kar zahteva testiranje v ustreznem okolju z relevantnimi podatki, definicijo uvedbe, navodila za nameščanje, planiranje uvedbe, dejansko uvedbo v produkcijo in analizo po uvedbi. To zagotavlja skladnost rezultatov s pričakovanji.

Opis procesa je podan v obliki naslednjega slapa:

Nadzor nad informacijskim procesom
nameščanje in avtorizacija rešitev in sprememb

zadovoljuje poslovne potrebe
implementacija novih ali spremenjenih sistemov, ki delujejo brez večjih težav

z osredotočanjem na
testiranje strojne in programske opreme, ki ustreza namenu in nima napak, planiranje uvedbe izdaje v produkcijsko okolje

kar se doseže z
- uvedbo metodologije testiranja
- planiranjem izdaj
- ocenjevanjem in avtorizacijo testnih rezultatov
- izvedbo analize po uvedbi izdaje

in se meri z
- čas prekinjenega delovanja, število potrebnih popravkov zaradi nezadostnega testiranja
- procent sistemov, ki glede na analize dosegajo pričakovane koristi
- procent projektov z dokumentiranim in avtoriziranim planom testiranja

Nadzorni cilji procesa so:

AI7.1 Izobraževanje
Pri razvoju, implementaciji ali spremembi informacijskega sistema je glede na plan izobraževanja in plan implementacije potrebno izobraziti ustrezno osebje, tako na strani uporabnikov kot na strani ponudnika storitev.

AI7.2 Plan testiranja
Na podlagi organizacijskih standardov je potrebno definirati plan testiranja, ki naj vključuje vloge, odgovornosti, vhodne in izhodne kriterije. Plan mora biti potrjen s strani vseh udeležencev.

AI7.3 Plan uvedbe
Definirati je potrebno plan uvedbe in umika izdaje, katera morata ravno tako biti potrjena s strani vseh udeležencev.

AI7.4 Testno okolje
Uporabljati je potrebno varno testno okolje, ki je kar najbolj podobno produkcijskemu v vseh pogledih: varnost, interne kontrole, operativne prakse, kvaliteta podatkov, obremenitev.

AI7.5 Migracija podatkov in sistema
Razvojne metode organizacije morajo vsebovati tudi planiranje migracije podatkov in infrastrukture.

AI7.6 Testiranje sprememb
Spremembe je potrebno testirati glede na definiran plan testiranja pred uvedbo sistema v produkcijsko rabo. Testiranje mora vključevati varnostne in učinkovitostne teste.

AI7.7 Končni sprejemljivostni test
Skrbniki poslovnih procesov in strokovnjaki s področja informatike naj ocenijo rezultate testiranja in avtorizirajo nameščanje sprememb v produkcijsko okolje.

AI7.8 Uvedba v produkcijsko okolje
Po testiranju naj se spremembe namestijo v produkcijsko okolje v skladu s planom uvedbe. Avtorizacijo za nameščanje morajo odobriti ključni udeleženci, kot so uporabniki, vzdrževalci in lastniki informacijskega sistema. Kjer je primerno, se lahko nekaj časa z obstoječim paralelno uporablja nov sistem in primerja delovanje.

AI7.9 Analiza po uvedbi
Organizacijski standardi upravljanja sprememb naj se dopolnijo tako, da postanejo analize po uvedbi obvezne. Postopke analize naj se standardizira.

Zrelostni model procesa upravljanja nameščanja in avtorizacije sprememb definira 6 zrelostnih stopenj:

- **0 – neobstoječ**
  Formalne metode nameščanja in avtoriziranja ne obstajajo, upravitelji in informacijsko osebje ne vidijo potrebe po preverjanju ustreznosti rešitev.

- **1 – začetni/ad hoc**
  Obstaja zavedanje po testiranju in avtorizaciji ustreznosti rešitev. Testiranje se izvaja le v nekaterih projektih, testiranje je prepuščeno projektni skupini, pristopi se od skupine do skupine razlikujejo. Formalna odobritev sprememb je redka ali je sploh ni.

- **2 – ponovljiv ampak intuitiven**

Testiranje in avtorizacija poteka že bolj konsistentno, ampak tipično ne sloni na nobeni metodologiji. Posamezne razvojne skupine se same odločajo o postopku testiranja in navadno ne opravijo integracijskega testiranja. Obstaja neformalen proces avtorizacije.

- **3 – definiran**

Uporabljane so formalne metodologije nameščanja, migracije in avtorizacije. Procesi so integrirani v življenjski cikel sistema in do neke mere avtomatizirani. Izobraževanje, testiranje in uvedba se navadno razlikujejo od definiranih procesov glede na posameznikove odločitve. Kvaliteta sprememb je nekonsistentna, zaradi česar prihaja do težav po uvedbi.

- **4 – upravljan in merljiv**

Formalne procedure so dobro organizirane in praktične, definirano je testno okolje in postopki avtorizacije. Vse večje spremembe sistema se odvijajo na formaliziran način. Zahteve uporabnikov in razvite rešitve se ocenjujejo na standarden in merljiv način, metrike se lahko učinkovito spremlja in analizira. Kvaliteta sprememb produkcijskega okolja je sprejemljiva, tudi če se pojavijo kakšne težave po uvedbi. Avtomatizacija procesov je odvisna od projekta in ni vnaprej planirana. Upravitelji so zadovoljni s trenutno uspešnostjo, čeprav se ne izvaja analiz po uvedbi. Testno okolje odraža značilnosti produkcijskega. Obremenitveni testi za nove sisteme in regresijski testi obstoječih sistemov se izvajajo za večje projekte.

- **5 – optimiziran**

Procesi namestitve in avtorizacije so dograjeni do nivoja dobrih praks, so popolnoma integrirani v življenjski cikel projekta in avtomatizirani, kjer je smiselno. Dobro razvita testna okolja, beleženje problemov in postopkov odprave napak zagotavljajo uspešno in učinkovito izobraževanje, testiranje in uvedbo v produkcijsko okolje. Avtorizacija se navadno zgodi brez potrebnih sprememb, analize po uvedbi ugotovijo le manjše napake. Analize po uvedbi so standardizirane, rezultati analiz pa se uporabljajo kot vir novih izboljšav. Obremenitveni testi za nove sisteme in regresijski testi obstoječih sistemov se izvajajo za vse projekte.

2.4. ISO 20000

Standard 20000 je mednarodni standard za upravljanje storitev informacijske tehnologije. Sestavljen je iz dveh delov:

- ISO 20000-1 promovira uporabo celovitega nabora procesov za učinkovito dostavo storitev, ki zadovoljijo poslovne potrebe uporabnikov (ISO/IEC 20000-1, 2005),
- ISO 20000-2 opisuje najboljše prakse za posamezna področja upravljanja storitev (ISO/IEC 20000-2, 2005).

Organizacija mora učinkovito upravljati številne povezane procese. Izhod nekega procesa je navadno vhod v enega ali več drugih procesov. Usklajenost integriranih procesov upravljanja storitev je ključ do večje učinkovitosti, nadzora in priložnosti za nenehne izboljšave.

Izvajanje procesov zahteva organizacijo osebja v storitvenem centru, vzdrževanju storitev, razvoju storitev in operativnih ekipah ter njihovo koordinacijo. Potrebna so tudi ustrezna orodja, ki zagotovijo učinkovitost in uspešnost procesov. Uresničevanje predpisov ISO standarda 20000 zahteva ustrezno kvalificirane in sposobne strokovnjake. Pri vpeljavi predpisov v določeno organizacijo jih je potrebno smotrno prilagoditi organizaciji in okolju, v katerem ta posluje.

Standard ISO 20000 definira številne tesno povezane procese upravljanja storitev, kot to prikazuje slika 14. Odnosi med procesi so odvisni od dejanske implementacije procesov v organizaciji in na splošno prekompleksni za modeliranje, zato na sliki niso prikazani. Seznami ciljev in kontrol niso vseobsegajoči, zato se lahko organizacije odločijo za dodatne cilje in kontrole, da bi izpolnile svoje poslovne potrebe.

Slika 14: procesi upravljanja storitev po ISO 20000.

Vir: ISO/IEC 20000-1, 2005, str. 7.

2.4.1. Cilji upravljanja izdaj

Glavni namen upravljanja izdaj je distribucija, dostava in sledenje eni ali več spremembam določene izdaje v produkcijsko okolje. Tudi ISO 20000 priporoča, da se upravljanje izdaj integrira z upravljanjem konfiguracij in upravljanjem sprememb.

Cilji upravljanja izdaj so:

- Politika izdaj mora definirati frekvenco in tip izdaj.
- Ponudnik storitev in uporabnik storitev morata skupaj načrtovati izdaje storitev, sistemov, programske in strojne opreme.
- Plani uvedbe izdaj morajo biti dogovorjeni in potrjeni s strani sodelujočih.
- Obstajati mora način umika izdaje, v primeru ko uvedba ne bi bila uspešna.
- Plan izdaj mora vsebovati časovni termin, seznam sprememb, znanih napak in problemov. Upravljanje izdaj mora takšne informacije posredovati upravljanju incidentov.
- Ocenjen mora biti vpliv zahtevkov za spremembe na plan izdaj.
- Upravljanje izdaj mora poskrbeti za osveževanje informacij o konfiguracijah in spremembah.
- Tudi izredne izdaje, ki so posledica izrednih sprememb, morajo biti definirane in izpeljane glede na definirane procese.
- Vzpostavljeno mora biti nadzorovano testno okolje, kjer se mora izdaje preverjati.
- Distribucija mora biti načrtovana in implementirana tako, da se integriteta strojne in programske opreme ohranja med grajenjem, dostavo in nameščanjem.
- Merjena mora biti uspešnost in neuspešnost izdaj. Merjeno mora biti število incidentov, povezanih z izdajo po njeni uvedbi. Analiza mora vključevati vpliv na poslovanje, informacijsko infrastrukturo in podporno osebje ter predlagati načrt izboljšanja storitve.

2.4.2. Procesi upravljanja izdaj

Splošni napotki

Upravljanje izdaj naj koordinira aktivnosti ponudnika storitev, dobaviteljev in uporabnikov pri planiranju in nameščanju izdaj. Dobro planiranje in upravljanje sta bistveni pri gradnji in uspešni distribuciji izdaj. Upravljati je potrebno tudi vplive in tveganja, ki ji izdaja prinaša poslovanju. Izdaja informacijskih sistemov, infrastrukture, storitev in dokumentacije mora biti dogovorjena z uporabniki storitev.

Vse pripadajoče spremembe dokumentacije morajo biti vključene v izdajo samo, npr. navodila za uporabo in dogovor o ravni storitev.

Ocenjen mora biti vpliv novih ali spremenjenih elementov konfiguracije, ki so del avtorizirane spremembe.

Sestavni elementi izdaje morajo biti sledljivi in varni pred nenadzorovanimi spremembami. Samo primerno preverjene in avtorizirane izdaje se lahko namestijo v produkcijsko okolje.

Politika izdaj

Politika izdaj mora definirati:

- frekvenco in tip izdaj,
- vloge in odgovornosti upravljanja z izdajami,
- avtoritete za potrjevanje napredovanja izdaj v testno in produkcijsko okolje,
- enolično identifikacijo in opis izdaj,
- koncept združevanja sprememb v izdaje,
- nivo avtomatizacije grajenja, nameščanja in distribucije izdaj,
- načine testiranja in potrjevanja izdaj.

Planiranje uvedbe izdaj

Ponudnik storitev mora zagotoviti medsebojno skladnost elementov konfiguracije v izdaji in v ciljnem okolju. Planiranje mora zagotoviti, da so spremembe informacijskih sistemov, infrastrukture, storitev in dokumentacije dogovorjene, avtorizirane, razvrščene, koordinirane in nadzorovane. Izdaja in uvedba izdaje je lahko načrtovana tudi po korakih, če vse podrobnosti na začetku niso znane. Plan izdaje in njene uvedbe naj bi navadno vseboval:

- časovne termine posameznega koraka in vsebino izdaje,
- seznam sprememb, problemov in znanih napak, ki ji izdaja odpravlja in znane napake, ki jih izdaja vsebuje,
- seznam procesov, potrebnih za uvedbo izdaje preko vseh poslovnih in geografskih enot organizacije,
- načrt umika izdaje, če bo to potrebno,
- proces preverjanja in avtorizacije,
- komunikacijo, pripravo, dokumentacijo in izobraževanje uporabnikov ter osebja vzdrževanja,
- logistiko in procese nabave, shranjevanja, distribucije, povezave, sprejema in odpisa strojne opreme,
- potrebne podporne vire, ki zagotovijo nemoteno delovanje storitev,
- identifikacijo odvisnih in povezanih sprememb ter tveganj, ki lahko ogrozijo nemoteno nameščanje izdaje v testno in produkcijsko okolje,
- potrjevanje izdaje,
- način revizije produkcijskega okolja po velikih izdajah, ki preveri dejansko stanje in ga primerja s pričakovanim.

Razvoj ali nakup programske opreme

Informacijski sistemi in programska oprema, kupljena ali razvita znotraj organizacije mora biti preverjena na način, ki je definiran v planu upravljanja konfiguracij.

Načrtovanje, gradnja in prilagajanje izdaj

Izdaja in distribucija morata biti načrtovani in implementirani tako, da:

- omogočata skladnost s sistemsko arhitekturo, upravljanjem storitev in infrastrukturnimi standardi ponudnika storitev,
- zagotavljata integriteto med gradnjo, dostavo in nameščanjem,
- uporabljata repozitorije pri upravljanju in nadzoru procesov,
- se lahko identificira tveganja,
- omogočata preverjanje ustreznosti ciljnega okolja pred nameščanjem,
- omogočata preverjanje celovitosti izdaje po distribuciji.

Izhod iz procesa mora vključevati izdajni zapisnik, navodila za nameščanje, seznam strojne in programske opreme glede na konfiguracijo. Izhodi morajo biti predani skupini, ki je zadolžena za testiranje. Procesi so lahko avtomatizirani, da se zmanjša možnost človeških napak, zagotoviti je potrebno ponovljivost procesa in tako omogočiti hitro uvedbo izdaj.

Preverjanje in avtorizacija izdaj

Potrjena vsebina, celovitost in ustreznost izdaje je rezultat procesa, ki mora:

- preveriti, da se nadzorovano testno okolje ujema s pričakovanim produkcijskim,
- zagotoviti, da je izdaja zgrajena iz elementov, ki so pod nadzorom upravljanja konfiguracij in da je v testno okolje nameščena na definiran način,
- preveriti ustreznost različnih testiranj: funkcionalnih testov, integracijskih testov, testov distribucije, nameščanja, ipd.,
- zagotoviti, da se z nivojem izvedenega testiranja strinjajo uporabniki in osebje vzdrževanja,
- poskrbeti, da je testiranje izdaje avtorizirala ustrezna oseba,
- preveriti, ali ciljno okolje izpolnjuje strojne zahteve,
- preveriti, da je izdaja po distribuciji celovita.

Dokumentacija

Izdaja mora vsebovati tudi ustrezno dokumentacijo, ki je pod nadzorom upravljanja konfiguracij, in vključuje:

- dokumentacijo, namenjeno upravljanju storitev, npr. dogovor o ravni storitev,
- dokumentacijo, namenjeno vzdrževanju storitev, npr. sistemski načrt, procedure nameščanja in vzdrževanja, procedura za ugotavljanje napak, operativna in administrativna navodila,
- opis procesov gradnje, izdaje, distribucije in nameščanja izdaje,
- procese umika izdaje,
- načrt izobraževanja uporabnikov in osebja vzdrževanja storitev,
- seznam elementov konfiguracije, ki so del izdaje, opis testnega okolja in testne dokumentacije,

- seznam sprememb, problemov in znanih napak,
- potrdilo o avtoriziranosti izdaje, poročila o preverjanju in sprejemljivosti.

Sistem ali storitev, ki se s pričakovanji in zahtevami ne ujema popolnoma, se mora identificirati in označiti v procesu upravljanja konfiguracij pred uvedbo v produkcijsko okolje. Informacije o znanih napakah morajo biti sporočene upravljanju incidentov. Če je izdaja zavrnjena, zakasnjena ali preklicana je potrebno obvestiti upravitelje sprememb.

Uvedba, distribucija in namestitev

Plan uvedbe je potrebno revidirati in opisati podrobnosti, ki zagotovijo izvajanje vseh potrebnih aktivnosti. Pomembno je, da je izdaja varno dostavljena v ciljno okolje. Procesi morajo zagotoviti, da:

- so vse hrambe strojne in programske opreme varne,
- obstajajo ustrezni postopki shranjevanja, pošiljanja, prejemanja in odpisa strojne opreme,
- so planirani in opravljeni varnostni pregledi (npr. električni testi, okoljski testi),
- uporabniki in osebje vzdrževanja vedó za nove izdaje,
- so nepotrebni produkti, storitve in licence odstranjeni.

Po distribuciji programske opreme preko mreže se je potrebno prepričati, da je izdaja celovita in nespremenjena. Po uspešni namestitvi je potrebno ustrezno označiti zapise upravljanja konfiguracij. Po namestitvi se lahko opravi anketiranje, s katerim ugotovimo uspešnost izdaje, služi pa tudi kot podlaga za upravljanje poslovnih odnosov.

Procesi po uvedbi izdaje

Takoj po uvedbi izdaje je potrebno meriti število incidentov kot posledica nove izdaje. Analizirati je potrebno njihov vpliv na poslovanje in na obremenjenost virov vzdrževanja storitev. Upravljanje sprememb mora po uvedbi izdaje analizirati implementacijo, izsledki pa služijo pri planiranju izboljšanja storitev.

3. Analiza trenutnega stanja

Da bi v podjetju, kjer sem trenutno zaposlen, lahko uvedli nekatere koncepte in najboljše prakse upravljanja izdaj, se je bilo najprej potrebno vprašati o trenutnem stanju. V nadaljevanju je predstavljeno podjetje In2, čemur sledi poglobljena primerjava trenutnih aktivnosti upravljanja izdaj s svetovno priznanimi najboljšimi praksami, na koncu poglavja pa so nakazane prednosti in slabosti trenutnih aktivnosti.

3.1. Predstavitev podjetja

Primarna dejavnost podjetja In2 je izdelava poslovnih informacijskih sistemov, med katerimi prednjači razvoj in vzdrževanje celovitega zavarovalniškega informacijskega sistema Insurance2. Podjetje je bilo ustanovljeno leta 1995 s sedežem v Kopru. Informacijski sistemi, razviti v In2, so v uporabi v zavarovalniških družbah v Sloveniji, podjetje pa se s svojimi produkti prebija tudi na Hrvaško in Srbijo.

Konec leta 2007 je bilo zaposlenih 30 ljudi, od tega je približno 70% aktivno udeleženih pri razvoju programske opreme. Ostalih 30% vključuje vodstvo in tehnično podporo uporabnikom. Podjetje In2 ustvarja prihodke s prodajo in vzdrževanjem informacijskih sistemov.

3.1.1. Produkt Insurance2

Insurance2 je celovit zavarovalniški informacijski sistem, ki podpira temeljne zavarovalniške procese:

- sklepanje zavarovanj,
- reševanje škodnih dogodkov,
- finančno – računovodsko poslovanje,
- sistem za podporo poslovnemu odločanju.

Vsi moduli dostopajo do enotne baze podatkov, zato je možno npr. najti določeno osebo, nadaljevati s pregledom njenih sklepalnih dokumentov, preveriti njeno plačilno disciplino, zgodovino prijavljenih škodnih dogodkov in se na podlagi teh informacij odločiti o npr. komercialnem popustu na pravkar sklepani polici.

Aplikacija Insurance2 je razvita na osnovi Oraclove tehnologije. Jedro sistema je podatkovna baza Oracle 10g, kjer je implementirana poslovna logika. Uporabniški vmesnik se izvaja na aplikacijskem strežniku OAS, sestavljen je iz vnosnih mask (Oracle Forms) in poročil (Oracle Reports). Uporabniki dostopajo do uporabniškega vmesnika preko brskalnika. Aplikacija obsega okoli 1500 baznih tabel, več kot 800 vnosnih mask in 600 poročil. Število vrstic PL/SQL kode ocenjujemo na 2 milijona.

Aplikacija je bila razvita primarno za slovenski jezik, a z uporabo nekaterih mehanizmov, ki omogočajo podporo mednarodnim trgom. Privzete valute in mere se lahko definira v šifrantih. Uporabniški vmesnik je možno s posebnimi orodji prevesti v katerikoli jezik, prav tako so vsa sporočila, ki jih aplikacija prikazuje uporabniku, parametrizirana in dejansko si lahko vsak uporabnik izbere poljuben jezik, ki ga aplikacija podpira – pogoj je le, da je bil narejen prevod uporabniških vmesnikov in sporočil v izbran jezik.

Aplikacija se lahko povezuje z ostalimi aplikacijami preko spletnih storitev. Glede na pretekle izkušnje smo razvili standardni nabor storitev, ki zadosti večino potreb po povezovanju, za specifične potrebe pa se razvijajo dodatne spletne storitve.

Vsaka stranka dobi popolno aplikacijo, ne glede na to katere module uporablja in katerih ne. Izjema so le nekatera poročila, ki se od stranke do stranke razlikujejo, predvsem gre tukaj za razne dopise, ki jih zavarovalnice pošiljajo svojim strankam.

Piratstvo programske opreme nam ne predstavlja grožnje. Produkt Insurance2 kot velik informacijski sistem namreč zahteva konstantno vzdrževanje, s čemer se med podjetjem In2 in zavarovalnico ustvari močan partnerski odnos. Tudi, če bi popolna piratska kopija naše aplikacije prišla nekomu v roke, si brez znanja o njenem delovanju, konfiguraciji in vzdrževanja težko predstavljamo njeno uspešno uporabo.

Upravljanje izdaj takšnega informacijskega sistema je zahtevna naloga. Spremembe programske opreme je potrebno dostavljati mnogim strankam, moduli informacijskega sistema so odvisni drug od drugega in se glede na spreminjajoče poslovne potrebe uporabnikov konstantno razvijajo. Posledično se v procesih izdaje programske opreme porabi veliko časa, zaradi kompleksnih procesov pa se poviša tveganje in možnost napak (Ballintijn, 2005).

3.1.2. Orodja

Za analizo (izdelavo ER-diagramov) in načrtovanje (izdelava logičnega podatkovnega modela) kakor tudi za gradnjo vnosnih mask uporabljamo orodje Oracle Designer. Poročila izdelujemo s pomočjo graditelja poročil Reports Builder. Bazni del aplikacije (PL/SQL) razvijamo z orodjema SQL Developer in TOAD.

Za spremljanje napak in razvojnih nalog uporabljamo informacijski sistem, razvit znotraj podjetja In2. Preko tega sistema poteka večina komunikacije s strankami, planirajo se kratkoročne in dolgoročne aktivnosti, beleži se porabljen čas, s pomočjo sistema se upravljajo konfiguracije in osnovno verzioniranje produkta.

3.1.3. Upravljanje sprememb

Razvoj novih modulov ali spreminjanje obstoječih modulov poteka po skupinah, glede na področje zavarovalništva (sklepalni, škodni in finančni del). Vsaka skupina ima 3 do 4 razvijalce/vzdrževalce, vodjo, ki ima na izbranem področju največ izkušenj, eno osebo, ki skrbi za podporo uporabnikom in testiranje ter še eno osebo, ki je zadolžena izključno za testiranje. Razvijalec, ki razvije nek modul, je kasneje tudi odgovoren za njegovo vzdrževanje. Izjeme so moduli, ki so zelo kompleksni in jih mora vzdrževati več razvijalcev, ali pa tako splošni, da jih lahko vzdržuje več razvijalcev. Zaradi preobremenjenosti nekaterih razvijalcev se vzdrževanje modulov lahko prenese na manj obremenjene razvijalce.

Pobuda za razvoj nekega modula je prijavljena napaka ali zahteva po dograditvi. To navadno zahteva spremembo ali razvoj določenih modulov, lahko pa gre samo za vnos podatkov v šifrant ali opravljanje drugih vzdrževalnih del. Razvijalec dobi zahtevo, jo sam ali s pomočjo vodje analizira in v dogovoru s stranko razvije predlog rešitve. Pri tem se oceni predviden čas, potreben za razvoj in testiranje, ter na podlagi prioritet zahtevo umesti v svoj plan. Pri razvrščanju nalog se držimo pravila, da imajo napake večjo prioriteto pred razvojnimi nalogami.

Hitro spreminjajoče se prioritete in zahteve uporabnikov so znana težava na področju vzdrževanja programske opreme (April et al., 2004), ki tudi v podjetju In2 povzroča veliko nejevolje. Težavo rešujemo s klasifikacijo zahtev po raznih kriterijih (tip zahteve, področje aplikacije, prioriteta) in časovnim razvrščanjem potrebnih aktivnosti (analiza, implementacija, testiranje).

Ko pride zadeva na vrsto za implementacijo, se razvijalec loti dela, pri čemer uporablja skupno razvojno okolje znotraj podjetja In2. V kolikor gre za večji obseg sprememb, se navadno oblikujejo delovne skupine, v katerih so zastopani strokovnjaki s strani uporabnikov, razvijalec, oseba ki skrbi za podporo uporabnikom in po potrebi še vodja razvijalne skupine. Med razvojem modulov se razvijalec za dodatna vprašanja ali pojasnitve obrne na strokovnjake s strani uporabnikov.

Razvijalec mora že med samim razvojem preverjati delovanje modula, ki ga razvija na novo ali spreminja. Po končanem razvoju mora razvijalec pripraviti popravek, ki vsebuje vse opravljene spremembe, te spremembe dokumentirati, in predlagati testne scenarije. Nato preda zadevo v testiranje osebi znotraj skupine, ki je zadolžena za testiranje in pisanje dokumentacije. Na podlagi testov se ta oseba odloči, ali je naloga uspešno ali neuspešno realizirana. V kolikor ni uspešno realizirana, vrne zadevo razvijalcu, če pa je uspešno realizirana, se spremembe v obliki popravka lahko namesti v testno okolje izbrane stranke, kjer lahko s testiranjem pričnejo uporabniki. Tudi uporabniki lahko spremembe zavrnejo zaradi neustreznosti, če pa jih potrdijo, se spremembe lahko namestijo v produkcijsko okolje.

Pri razvoju baznega dela aplikacije se uporablja enotna razvojna baza podatkov in enoten aplikacijski strežnik. Ker vsak razvijalec skrbi za in spreminja samo svoje module, načeloma ne prihaja do sočasnega spreminjanja enega modula s strani več oseb. Lahko pa se zgodi, da med razvojem nekega baznega paketa ta postane neveljaven in posledično postanejo neveljavni ostali paketi, ki ga uporabljajo. Zato se večkrat zgodi, da med razvojem nekega ključnega modula del aplikacije na razvojni bazi ne deluje, kar npr. onemogoča testiranje.

3.1.4. Upravljanje konfiguracij

Osnovno upravljanje konfiguracij se vrši preko internega informacijskega sistema. Za vsak element konfiguracije se lahko definira njegove osnovne podatke (tip, šifra, opis), in razvijalce, ki zanj skrbijo. Elementi konfiguracije se lahko povezujejo z nalogami in s spremembami, v katerih so vsebovani. Ker pa ni nadzora, nekateri razvijalci ne vnašajo teh povezav. Elementi konfiguracije se ne verzionirajo, njihova dejanska vsebina pa se hrani odvisno od tipa elementa konfiguracije:

- v orodju Oracle Designer se hranijo vnosne maske, ER diagrami ter bazni del aplikacije (tabele, procedure, funkcije, paketi, pogledi, ipd.),
- na mrežnem disku se hranijo poročila.

Za bazni del se večkrat zgodi, da razvijalec sprememb ne shrani v orodje Oracle Designer in je posledično zadnja verzija nekega elementa na razvojni bazi. Čeprav med elementi konfiguracije obstajajo kompleksne odvisnosti (npr. obrazec kliče drugi obrazec, funkcija uporablja tabelo, poročilo uporablja pogled, ipd.), se jih nikjer eksplicitno ne vodi.

V informacijskem sistemu za upravljanje konfiguracij ni informacij o ciljnih okoljih in tam nameščeni programski opremi. Če potrebujemo informacijo o tem, katera verzija nekega elementa je nameščena v določenem okolju, je potrebno to ročno preveriti. Ker so nekatera okolja zaščitena pred nepooblaščenim dostopom, pa tega sploh ni možno preveriti. Za nekatere elemente se da o tem sklepati na podlagi informacij o poslanih in nameščenih spremembah, kar pa vsekakor ni učinkovito.

3.1.5. Upravljanje izdaj

Izdaje

V podjetju praktično ne poznamo pojma izdaj v smislu združevanja sprememb. Vsaka sprememba, naj bo to odprava napake, ali pa nova funkcionalnost, se dejansko namesti ločeno od ostalih sprememb. Na ta način lahko zagotovimo kratek odzivni čas, saj spremembi ni potrebno čakati naslednje izdaje, ampak se lahko namesti takoj, ko testerji potrdijo njeno ustreznost. Vsaka sprememba ima svojo številko glede na zahtevek, zaradi katerega je nastala in vsebuje enega ali več elementov konfiguracije. Nameščanje spremembe je sicer odvisno od nameščanja predhodnih sprememb, se pa teh odvisnosti ne vodi v sistemu za upravljanje konfiguracij.

Enota izdaje produkta Insurance2 je posamezen modul aplikacije, torej najmanjši element, ki se sploh lahko izda. Z izdajanjem modulov dosežemo hitro prilagajanje poslovnim spremembam, saj se lahko takoj po končanem razvoju nekega modula prične njegovo testiranje, po testiranju pa se modul lahko namesti v produkcijsko okolje. Proces razvoja enega modula je torej neodvisen od razvoja drugih modulov, če le ne gre za soodvisne spremembe.

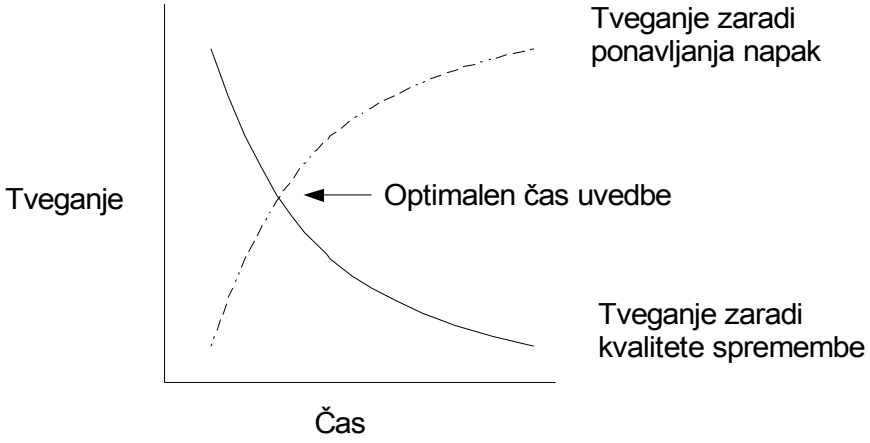
Politika in planiranje izdaj

Čeprav politika izdajanja obstaja, ni nikjer eksplicitno zapisana. Številčenje izdaj in sprememb, kot tudi frekvenca izdajanja, sta prilagojeni vsaki stranki. Ob uvedbi aplikacije k neki novi stranki najprej namestimo celotno aplikacijo, torej gre za celotno izdajo, nato pa stranki redno dostavljamo samo spremenjene elemente konfiguracije, gre torej za dostavo manjših, delta izdaj. Nekaterim strankam nameščamo spremembe pogosto, npr. vsak drugi teden, nekaterim strankam pa, odvisno od potreb in dogovorov, bolj redko, npr. vsak tretji mesec. Takšne izdaje lahko smatramo kot paketne izdaje. Pri vsaki stranki poznamo tudi t.i. izredne popravke, ki odpravljajo kakšno kritično napako, in se jih lahko namesti takoj, ko je takšen popravek pripravljen.

Ker izdajamo vsako spremembo ločeno od ostalih, se planiranje izvaja na nivoju sprememb oz. nalog, ki so podlaga vsaki spremembi. S tem ko ocenimo čas, potreben za realizacijo posamezne naloge, in nalogo postavimo v terminski plan določenega razvijalca, dejansko planiramo tudi izdajo te spremembe. Sama uvedba te izdaje pa je potem odvisna od potreb in plana nameščanja posamezne stranke.

Pri vsaki spremembi moramo poiskati kompromis med zgodnjo uvedbo, s katero se lahko hitro izkoristi novosti, ter pozno uvedbo, ko je določen del aplikacije bolj dovršen in kvaliteten. Uporabniki in razvijalci dejansko primerjajo obe alternativi (pogosto sicer nezavedno) in iščejo časovno točko, v kateri je ekonomska vrednost spremembe najvišja (Sassenburg, 2005). Včasih spremembe odpravljajo napake v aplikaciji, takrat pri izdelavi in nameščanju izrednih izdaj kolebamo med hitro uvedbo spremembe, ki je lahko manj kvalitetna in povzroči dodatne zaplete, ali kasnejšo uvedbo spremembe, ko je učinek napake na poslovanje večji (Beattie et al., 2002). Slika 15 prikazuje hipotetični graf tveganja v odvisnosti od časa.

Slika 15: hipotetični graf tveganja v odvisnosti od časa.



Vir: Beattie et al., 2002, 1 str.

Vloge in odgovornosti osebja, vpletenega v upravljanje izdaj, so poznane, a nikjer eksplicitno zapisane. Razvijalci so odgovorni za razvoj aplikacije in pripravo sprememb. Testerji so odgovorni za prvi cikel testiranja aplikacije, uporabniki pa za drugi cikel testiranja. Za nameščanje izdaj je odgovorna oseba odvisna od primera do primera. Nekatere stranke imajo svoje administratorje, ki nameščajo spremembe, druge stranke pa nameščanje sprememb prepuščajo podjetju In2, kjer to delo opravlja administrator baze podatkov ali posamezen razvijalec. Ker se nadzor in spremljanje procesov ne izvaja, odpade vloga upravitelja.

Grajenje izdaj

Za sestavo posamezne spremembe je odgovoren razvijalec. Ko se spremeni bazni del aplikacije, mora razvijalec pripraviti ustrezne skripte, ki vsebujejo ukaze za spreminjanje baznega dela aplikacije (npr. nova tabela, nov stolpec, nova procedura) v ustreznem vrstnem redu. Pri spremembi vnosne maske ali poročila, pa mora razvijalec pripraviti izvorno datoteko. Skripte in izvorne datoteke mora nato razvijalec združiti v ZIP datoteko, ki jo ustrezno poimenovano vnese v interni informacijski sistem. Na podlagi te datoteke je možno vsako spremembo namestiti v katerokoli okolje na ponovljiv način.

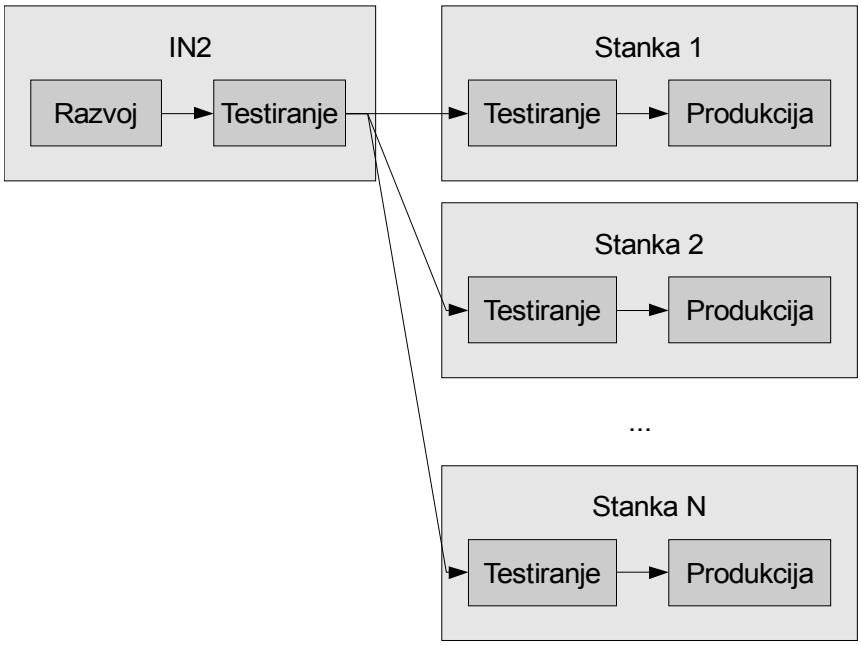
Pri strankah, ki se odločajo za uvedbo večjega števila sprememb naenkrat, se posamezne spremembe navadno sproti nalagajo v testno okolje, kjer se jih testira in preverja toliko časa, dokler uporabniki niso popolnoma zadovoljni s funkcionalnostjo. Nato pa se vse spremembe namesti v produkcijsko okolje kot ena izdaja. Grajenje izdaj v tem primeru pomeni sestavljanje velikega števila sprememb v eno izdajo. Izvršilne datoteke vnosnih mask in poročil se združi v eno mapo, ki se jo ob uvedbi izdaje skopira na ustrezen aplikacijski strežnik, skripte za spremembo baznega dela pa se sproži eno za drugo s pomočjo dodatne skripte.

Postopek grajenja sprememb in izdaj ni avtomatiziran, saj sestavo sprememb opravljajo razvijalci, sestavo izdaj pa osebe, ki izdaje nameščajo v razna okolja. Napake, povezane z ročnim sestavljanjem sprememb, se ugotovijo in odpravijo v procesu testiranja sprememb.

Testiranje

Za zagotavljanje kvalitete uporabljamo na In2 večkratno nameščanje in testiranje sprememb v različnih okoljih. Prvi del testiranj se odvije v razvojnem in testnem okolju znotraj podjetja In2, drugi del pa se odvija v okoljih pri vsaki stranki, ki ima nameščeno našo aplikacijo (slika 16). Ko so spremembe tudi tam potrjene, se namestijo v produkcijsko okolje. Z večkratnim testiranjem se preveri funkcionalnost programske opreme in postopke nameščanja posamezne spremembe.

Slika 16: napredovanje sprememb iz razvojnega v produkcijsko okolje.



Vir: lasten.

Prvo okolje je razvojno, kjer se spremembe konstantno in nenadzorovano odvijajo s strani razvijalcev. Ko je razvoj neke naloge končan, se spremembe v obliki popravka namestijo v testno okolje znotraj podjetja, kjer lahko interni testerji preverijo spremembe. Pri razvoju in testiranju znotraj podjetja In2 uporabljamo testne podatke, za katere skrbimo razvijalci oz. testerji. Pri tem testiranju gre predvsem za funkcionalno in integracijsko testiranje, zaradi omejenega obsega testnih podatkov pa se ne izvaja npr. testiranje obremenitev ali hkratnega dela več 100 uporabnikov.

V drugem delu testiranj, ki se odvija v testnih okoljih naših strank, se lahko testiranje opravi na veliko večji količini podatkov, ker so to navadno kopije produkcijskega okolja. Testiranje tam opravijo ključni uporabniki, ki so odgovorni za potrditev sprememb, razvijalec ali tester pa tudi lahko v testnem okolju dodatno preveri delovanje aplikacije. Ko so spremembe

potrjene se lahko namestijo v produkcijsko okolje. Termin nameščanja je od stranke do stranke različen. Nekatere stranke npr. same nameščajo spremembe vsak drugi teden, nekaterim strankam nameščamo spremembe samo po potrebi oz. v dogovoru s stranko, lahko tudi vsak dan ali pa na daljša časovna obdobja, npr. vsak tretji mesec.

Nekatere stranke imajo poleg testnega in produkcijskega okolja še dodatna okolja: razvojno in izobraževalno. Včasih namreč že med razvojem določene funkcionalnosti potrebujemo večje količine podatkov, ali pa razvoj poteka s kakšnim zunanjim partnerjem, zato uporabimo razvojno okolje, ki ga postavimo s kopiranjem produkcijskega. Da bi izobraževanje novih uporabnikov potekalo nemoteno od testiranja in razvoja aplikacije, si nekatere stranke postavijo dodatno okolje, ki je navadno prav tako kopija produkcijskega. Ob večjih razvojnih nalogah, kot je bil npr. prehod na novo valuto (EUR), smo ravno tako uporabili dodatno okolje, kjer smo testirali le spremembe v zvezi s to nalogo.

Komunikacija in izobraževanje

Podjetje In2 pri vsaki novi stranki opravi izobraževanje ključnih uporabnikov, ki nato izobrazijo vse ostale uporabnike informacijskega sistema. Tudi izobraževanje na novo zaposlenih ljudi opravljajo stranke same.

Ker uporabniki kasneje sodelujejo pri podajanju novih zahtev in testiranju sprememb, se z novostmi seznanijo že med podajanjem zahtev ali kasneje med testiranjem, kjer se ključne uporabnike tudi ustrezno izobrazi. Ti ključni uporabniki so nato odgovorni za prenos znanja in izobraževanje ostalih uporabnikov. Vsako spremembo mora razvijalec dobro opisati, tako da lahko interni tester dopolni navodila, ki so potem na voljo strankam.

Pred uvedbo sprememb se uporabnikov eksplicitno ne obvešča. Vse spremembe z avtorizacijo za nameščanje se namestijo ob naslednjem nameščanju. Obveščanje uporabnikov in razvijalcev po nameščanju je odvisno od stranke do stranke. Kjer nameščanje opravi nekdo na In2, potem obvesti razvijalce sprememb, razvijalci pa potem preko informacijskega sistema obvestijo stranke. Ena od strank, ki je razvila svoj sistem za vodenje sprememb, pa avtomatsko obvesti tako razvijalca kot uporabnika z elektronskim sporočilom.

Distribucija sprememb

Spremembe se vnašajo v informacijski sistem, razvit znotraj podjetja In2. Ob vsaki novi spremembi se po e-pošti pošlje obvestilo ustreznim osebam, ki so zadolžene za nameščanje sprememb. Preko informacijskega sistema je omogočen pregled vseh sprememb, skupaj z elementi konfiguracije, ki jih sprememba vsebuje. Uporabniki z ustreznimi pravicami si lahko iz sistema prenesejo datoteko s spremembami. Ker se spremembe nameščajo le na strežnike, potreba po kompleksni distribuciji sprememb na delovne postaje odpade. Glede na velikost posameznih sprememb (od 10KB do 5MB) trenutne pasovne širine mrežnih povezav ne

predstavljajo omejitvenega dejavnika, tudi ko je potrebno opraviti distribucijo večjega števila sprememb hkrati.

Nameščanje sprememb

Spremembe se po distribuciji preko e-pošte nameščajo ročno. Večinoma gre za poganjanje baznih skript in kopiranje izvršilnih datotek obrazcev in poročil na aplikacijski strežnik. Oseba, ki je zadolžena za nameščanje sprememb, navadno o uspešni namestitvi obvesti razvijalca in uporabnike, ob neuspešni namestitvi pa kontaktira razvijalca ali administratorja baze podatkov na In2, ki potem pomaga pri nameščanju spremembe. Nameščanje sprememb za vsako novo stranko opravljamo na In2 le toliko časa, dokler ne izobrazimo ustrezne osebe na strani kupca.

Nameščanje sprememb v produkcijsko okolje se navadno izvaja izven delovnega časa, tako da uporabniki ne čutijo motenj v delovanju informacijskega sistema. Manjše spremembe, ki ne zmotijo dela uporabnikov, ali pa nujni popravki, se nameščajo tudi med delovnim časom. Nameščanje sprememb v manj obremenjena razvojna in testna okolja se dogaja med delovnim časom, ob večji obremenitvi, npr. ob intenzivnem testiranju pa se tudi v ta okolja spremembe namešča le izven delovnega časa oz. po dogovoru.

Ena od strank je za vodenje sprememb razvila ločen informacijski sistem, s katerim lahko učinkovito obvladujejo nameščanje sprememb, saj v vsakem trenutku vedó, katere spremembe so že namestili in katerih še ne, katere spremembe so potrjene kot primerne za nameščanje, katere morajo uporabniki še preveriti, ipd. Preko tega sistema se vodijo tudi odvisnosti med spremembami. Njihov sistem žal ni povezan z našim sistemom za vodenje konfiguracij, tako da imajo razvijalci pri izdaji sprememb dodatno delo, saj morajo spremembe vnesti tudi v njihov informacijski sistem.

Pred nameščanjem sprememb se ne opravljajo nobene posebne kontrole. Dejansko bi lahko med namestitvijo spremembe prišlo do pomanjkanja prostora na aplikacijskem strežniku ali strežniku baze podatkov, a se ta okolja vsakodnevno spremlja in skrbi, da je vedno na voljo dovolj prostega prostora. Za nameščanje sprememb ni potrebna prisotnost kakršnekoli posebne programske opreme, zato potreba po takšnih kontrolah odpade.

Umik sprememb

Umikov sprememb ne podpiramo in ne izvajamo. Enkrat, ko je določena sprememba nameščena v neko okolje je ena od možnosti, da jo razveljavimo, priprava dodatne spremembe, ki povrne aplikacijo v stanje pred prvo spremembo. Obstaja še možnost povrnitve celotnega sistema v predhodno stanje iz arhivskih in varnostnih kopij, a se niti ena niti druga možnost v praksi še nista pojavili. V primeru, da gre pri nameščanju določene spremembe kaj narobe, se lahko spremembo ponovno namesti ali pa kontaktira odgovornega

razvijalca ali administratorja baze podatkov na In2, ki potem pomaga pri nameščanju spremembe. V skrajnem primeru se lahko okolje obnovi iz varnostne kopije.

Nadzor nad procesi

Formalnega nadzora na procesi izdajanja se v podjetju In2 ne izvaja. V primeru napak ali neuspešnih izdaj se napake sicer hitro odpravijo, vendar se vkljub ponavljajočim se napakam procesi ne spremenijo. Meritve ključnih kazalcev uspešnosti se ravno tako ne izvajajo, zato ni možno ugotoviti, ali so izdajni procesi uspešni ali ne.

3.2. Prednosti

Dejstvo, da podjetje In2 uspešno deluje na domačem in se hkrati širi na tuje trge, potrjuje njegovo uspešnost pri razvoju programske opreme. Informacijske sisteme, razvite znotraj In2, vsakodnevno uporablja več tisoč uporabnikov velikih podjetij, kot so zavarovalnice in banke. Pri analizi razvoja in izdajanja programske opreme vidimo, da je ključna strategija za obvladovanje morebitnih težav enostavnost.

Kljub temu, da je zavarovalništvo kompleksno področje, sta tako arhitektura aplikacije kot razvoj posameznih modulov dokaj enostavna, zato se lahko razvoj aplikacije in njenih modulov osredotoča na zadovoljevanje poslovnih potreb naših strank. Celotna aplikacija uporablja skupno podatkovno bazo, kar omogoči enoten dostop do podatkov iz vseh modulov. Vsi obrazci so zgrajeni z enim orodjem za gradnjo obrazcev, vsa poročila z enim orodjem za gradnjo poročil, tako da je razvijalcem dovolj, da se dobro naučijo uporabljati po eno orodje.

Uporaba enotne razvojne baze poenostavlja verzioniranje programske opreme. Večina programske kode se namreč nahaja v obliki PL/SQL kode v bazi, kjer je vedno dostopna le zadnja verzija. Verzioniranje bazne sheme in aplikacijskega strežnika bi sicer omogočilo ločen razvoj aplikacije, hkrati pa zahtevalo dodatna orodja, procese ter strojno opremo, kar bi se odražalo na bolj kompleksnem razvojnem procesu. Zato se verzioniranje preprosto ne uporablja, vsak razvijalec pa skrbi za usklajenost verzij svojih modulov pri pošiljanju sprememb. Morebitne napake se ugotovijo v procesu testiranja in odpravijo.

Dostava vseh elementov konfiguracije vsem strankam poenostavlja izdajanje. Ni namreč potrebno voditi informacij o tem, katere module je katera stranka kupila, katere elemente konfiguracij posamezni moduli potrebujejo, in pri sestavljanju izdaje ni potrebno teh informacij upoštevati, tako da se strankam ne bi dostavljalo nepotrebnih elementov. Glavna prednost dostave manjšega števila elementov bi bili manjši prenosi podatkov, to pa pri današnji hitrosti mrežnih povezav in kapaciteti trdih diskov ni več problem.

Uvedba umikov sprememb bi ravno tako dodatno zakomplicirala razvoj in testiranje vseh sprememb. Ker se vsaka sprememba namesti v vsaj dve testni okolji, eno znotraj podjetja In2 in eno pri stranki, se morebitne napake sprememb ugotovijo pred nameščanjem v

produkcijsko okolje. V primeru, da do napak pri nameščanju vseeno pride, priskoči na pomoč administrator baze podatkov ali odgovorni razvijalec.

Uporaba informacijskega sistema za vodenje zahtev strank in povezav med zahtevami, spremembami in elementi konfiguracije omogoča podjetju In2 uspešno obvladovanje kompleksnega produkta. Možno je spremljanje napredovanja zahtev od analize, preko implementacije in testiranja do produkcijske rabe. Ob incidentu je možno hitro ugotoviti, kdo od razvijalcev je zadolžen za določen modul, ali za modul obstajajo kakšne znane napake, ali je bil za modul izdan kakšen popravek, ipd. Možno je spremljanje obremenjenosti posameznega razvijalca in planiranje bodočih sprememb, informacijski sistem olajša in centralizira tudi komunikacijo s strankami.

3.3. Slabosti

Pri primerjavi procesov sem ugotovil nekaj točk, kjer bi podjetje In2 lahko izboljšalo svoje delovanje. Za slabosti, strnjene v tem poglavju, so predlagane izboljšave opisane v naslednjem poglavju. Ker je upravljanje izdaj odvisno od kvalitete upravljanja sprememb in upravljanja konfiguracij, se nekaj točk nanaša tudi na ostali dve področji.

Sočasni razvoj aplikacije na skupni razvojni bazi je vzrok za težave s konsistentnostjo (Sadalage, 2004). Moduli so namreč med seboj povezani in razvoj modulov, ki traja dalj časa, ovira razvoj ostalih modulov, ki so od njega odvisni. Ker se verzioniranje ne uporablja, se ne da ločiti razvoja aplikacije v nekaj vej in jih ob končanem razvoju združiti (Schuh, 2002). Ravno tako se ne da nadzirati sprememb nad dejansko izvorno kodo, kajti vsakdo lahko kadarkoli spremeni katerikoli del aplikacije. Za kvaliteto izvorne kode sicer obstajajo standardi, a njihovega upoštevanja nihče ne preverja.

V informacijskem sistemu se sicer vodijo elementi konfiguracije in njihove povezave s spremembami, kompleksne povezave med elementi konfiguracije pa niso nikjer eksplicitno zapisane. Vnos povezav med elementi konfiguracije in spremembami se opravlja ročno, nekateri razvijalci pa teh informacij preprosto ne zapisujejo. Knjižnica veljavne programske opreme nima vseh potrebnih lastnosti, od katerih izstopata verzioniranje in centraliziranost (obrazci so shranjeni v orodju Oracle Designer, poročila na mrežnem disku, bazni del aplikacije pa na razvojni shemi).

Za sestavljanje in pošiljanje sprememb so zadolženi razvijalci. Ko vsebina določene spremembe vsebuje spremembe baznega dela aplikacije, se ročno pripravi sql skripta. Pri velikem številu sprememb, kompleksnih odvisnostih in sočasnem razvoju na eni bazni shemi je ročni postopek sestavljanja sprememb neustrezen in vodi k napakam. Takrat je potrebno zaradi neustrezno sestavljene spremembe narediti novo spremembo, ki odpravi napake v prejšnji.

Ker je potrebno spremembe ločeno poslati vsaki posamezni stranki, bo to ob naraščajočem številu strank zahtevalo vedno več ponavljajočega dela. Ob nameščanju spremembe v neko okolje se ne preverja, ali so v to okolje bile nameščene vse predhodno potrebne spremembe. Dejansko bi se to lahko preverjalo le, v kolikor bi bile poznane odvisnosti med spremembami, a se tudi teh odvisnosti v internem informacijskem sistemu ne vodi, in v kolikor bi se za vsako okolje vedelo, katere spremembe so bile nameščene, a se tudi teh informacij ne vodi.

4. Optimizacija skrbništva izdaj

Upoštevanje dobrih praks upravljanja informacijske tehnologije postaja vse pomembnejše zaradi številnih vzrokov:

- Organizacije zahtevajo boljši izkoristek investicij v informacijsko tehnologijo, od nje pričakujejo vedno večjo dodano vrednost.
- Vlada prepričanje, da se stroški informacijske tehnologije konstantno povečujejo.
- Informacijski sistemi morajo ustrezati zakonodaji, ki predpisuje vedno več zakonov: npr. Zakon o varstvu osebnih podatkov.
- Vpliv na izbiro zunanjega izvajalca storitev in upravljanje zunanjih izvajalcev.
- Vedno bolj kompleksna tveganja, povezana z informacijsko tehnologijo.
- Iniciative po upoštevanju nadzornih ogrodij in dobrih praks, da se omogoči merjenje in izboljševanje kritičnih informacijskih aktivnosti.
- Potreba po zmanjševanju stroškov z uporabo standardiziranih pristopov.
- Dozorevanje in posledično širši sprejem ogrodij kot so npr. COBIT, ITIL, CMMI, PRINCE2, ISO standardi, ipd.
- Potreba po primerjanju organizacije in njenih konkurentov pri upoštevanju splošno sprejetih standardov.

Po analizi ogrodij ITIL in COBIT ter ostale literature sem ugotovil, da bi uvedba nekaterih dobrih praks lahko izboljšala procese izdajanja programske opreme podjetja In2. Ker je upravljanje izdaj močno povezano z upravljanjem sprememb in konfiguracij, je smiselno, da se dogradi interni informacijski sistem, v katerem se že vodijo incidenti, spremembe in konfiguracije. Tako bi bile na enem mestu zbrane in povezane informacije o konfiguracijah, problemih, spremembah in izdajah.

Na začetku poglavja so obravnavani incidenti, ki so se v podjetju In2 zgodili v zadnjem letu in so povezani z izdajanjem programske opreme. Incidenti so bili moje izhodišče za ugotavljanje možnih izboljšav, saj sem za vsak incident ugotovil, katera izboljšava bi preprečila pojavljanje takega tipa incidentov. Predlogi izboljšav so podrobno opisani v nadaljevanju poglavja, pri čemer je ocenjen čas, potreben za uvedbo predloga, in navedene bistvene prednosti, ki bi jih podjetje z uvedbo posamezne izboljšave pridobilo. Na koncu poglavja so predlogi izboljšav primerjani med seboj, glede na časovno zahtevnost in pričakovane prednosti je predstavljen tudi plan vpeljave izboljšav.

4.1. Primeri incidentov

Navedem naj nekaj konkretnih primerov incidentov, ki so se zgodili v preteklem letu. Primeri ponazarjajo nekatere slabosti, opisane v poglavju 3.3, hkrati pa so dobra podlaga za ugotavljanje možnih izboljšav procesov, s katerimi bi zagotovili, da se takšni incidenti ne bi več pojavljali.

4.1.1. Primer 1

Nameščanje sprememb v produkcijsko okolje se je zgodilo med delovnim dnevom, nameščanje je opravil kar razvijalec sam, ki poleg tega ni poznal posledic spreminjanja baznih objektov na uporabo aplikacije že prijavljenih uporabnikov. Nekaj uporabnikov je zaradi tega moralo ugasniti aplikacijo in se ponovno prijaviti, nekaj uporabnikov pa se je zaradi neveljavnih baznih paketov začelo med seboj zaklepati. Tudi te uporabnike je bilo potrebno nasilno odjaviti in še enkrat prevesti neveljavne pakete. Škoda, ki je pri tem nastala, bi se lahko definirala s časom, ki so ga uporabniki porabili, da so se ponovno prijavili v aplikacijo ter opravili spremembe, ki jih prej niso shranili. Nekateri uporabniki so prijavili incidente, kar je dodatno obremenilo osebje za podporo uporabnikom, zmanjšal pa se je tudi ugled podjetja.

Predlog 1: Nameščanje sprememb bi moralo biti časovno omejeno na termine, ko aplikacijo uporablja malo ali nič uporabnikov, da se motnje v delovanju storitev ne zazna (Carzaniga et al., 1998).
Predlog 2: Za nameščanje mora biti odgovorna oseba z ustreznim strokovnim znanjem, ki pozna posledice nameščanja in razume, zakaj se tega ne počne med delovnim časom.

4.1.2. Primer 2

Nameščanje sprememb v produkcijsko okolje se je zgodilo po delovnem času, a zaradi slabo pripravljenega in nepreverjenega popravka je produkcijsko okolje postalo delno neuporabno. Uporabniki so naslednji dan to takoj občutili, saj niso mogli uporabljati dela aplikacije, sledilo je izredno nameščanje dodatnih sprememb med delovnim časom.

Predlog 1: Vsaka sprememba mora biti preverjena v testnem okolju. Napaka v omenjenem popravku bi se namreč ugotovila že pri nameščanju v testno okolje (Carzaniga et al., 1998).
Predlog 2: Nepreverjenih sprememb naj ne bo možno namestiti v produkcijsko okolje. Oseba, ki namešča spremembe, mora namestiti le preverjene in avtorizirane spremembe – česar pa trenutno ne vodimo.
Predlog 3: Uvedba umikov sprememb. Če vsi ostali postopki zatajijo in se napaka le prikrade v produkcijsko okolje, bi lahko z umikom spremembe vzpostavili prvotno stanje.

4.1.3. Primer 3

Sprememba ene od masovnih obdelav podatkov je vključevala logiko, ki je ob vsaki napaki poslala e-sporočilo na naslov razvijalca. Ker sprememba ni bila dobro preverjena v testnem okolju, je v produkciji ta logika poslala nekaj 10.000 e-sporočil preko strežnika za odhajajočo pošto. Strežnik je pričel pošiljati sporočila drugemu strežniku, kjer je imel svoj poštni predal razvijalec, ta pa je izvorni strežnik kmalu postavil na seznam pošiljateljev neželene elektronske pošte, kar je ohromilo pošiljanje vseh ostalih e-sporočil iz strankinega strežnika.

Predlog: Sprememba mora biti pregledana in odobrena s strani drugega razvijalca ali vodje razvijalca, s čemer bi se v prvem koraku lahko preprečilo vgrajevanje takšne logike.

4.1.4. Primer 4

Pri nameščanju izdaj se večkrat zgodi, da se spremembe med seboj »povozijo«. Nek element konfiguracije, ki je vsebovan v različnih spremembah, se lahko namesti v okolje v nepravilnem zaporedju. Najprej se namesti novejša verzija tega elementa, nato pa še starejša verzija. Vzrok za takšne incidente so kompleksne povezave med elementi konfiguracije (npr. modul A je odvisen od modula B), spremembami (npr. sprememba C mora biti nameščena pred spremembo D) ter med elementi in spremembami (npr. sprememba E vsebuje verzijo 1.3 elementa F). Vodenja nekaterih povezav trenutni informacijski sistem ne omogoča, nekatere povezave pa niso vedno popolne, ker jih razvijalci ne vnašajo, orodja, ki bi to avtomatsko počelo, pa nimamo.

Predlog: Voditi je potrebno odvisnosti med spremembami in elementi konfiguracije (Dolstra, Visser, Jonge, 2004). Za vsako okolje je potrebno vedeti, katere spremembe so bile nameščene in katere ne. Iz tega lahko sklepamo, katere spremembe lahko namestimo in katerih še ne. Glede na odvisnosti je potrebno določiti ustrezni vrsti red nameščanja.

4.2. Predlogi izboljšav sistema

Po primerjavi trenutnih aktivnosti upravljanja izdaj s svetovno uveljavljenimi najboljšimi praksami sem ugotovil določene slabosti. Našteti incidenti potrjujejo, da slabosti obstajajo, skupaj s primerjavo pa predstavljajo osnovo za ugotavljanje možnih izboljšav. V nadaljevanju poglavja so predlogi izboljšav podrobno opisani, pri čemer je ocenjen čas, potreben za uvedbo predloga in navedene bistvene prednosti, ki bi jih podjetje z uvedbo posamezne izboljšave pridobilo.

4.2.1. Politika izdaj

Politika izdajanja, ki sicer obstaja v glavah zaposlenih, bi morala biti zapisana. Tako bi se lahko vsak posameznik nedvoumno seznanil s smernicami izdajanja, novincem pa bi olajšali učenje in razumevanje procesov izdajanja (Erenkrantz, 2003). Vsak posameznik bi jasno razumel svojo vlogo in avtoriteto ter vloge ostalih, ki sodelujejo v povezanih procesih.

Interni informacijski sistem, kjer spremljamo spremembe, bi z nekaj nadgraditvami lahko uporabili tudi za merjenje ključnih kazalcev uspešnosti izdajnih procesov. Za vsako stranko in okolje bi lahko spremljali število sprememb v različnih časovnih intervalih, ugotavljali odzivne čase testerjev, spremljali število napak pri namestitvah, ipd.

4.2.2. Obveščanje o spremembah

Nekaj dni pred uvedbo izdaje bi lahko ustreznim osebam poslali obvestilo s seznamom vseh sprememb, ki se bodo namestile. Tako bi bili ključni uporabniki bolje seznanjeni s spremembami, njihova pričakovanja bi bila bolj realna (Bays, 2003). Osebje, ki skrbi za podporo, bi se lahko bolje pripravilo na morebitne težave, hkrati pa bi vedelo, katere težave se ne bodo več pojavljale.

Tudi po namestitvi bi bilo potrebno obvestiti ustrezne osebe, še posebej v primeru napak ali neuspešne uvedbe kakšne spremembe. Tako bi se namreč preprečilo prijavljanje incidentov zaradi nedelovanja novih funkcionalnosti, ki jih uporabniki sicer pričakujejo z izdajo, a se uvedba izdaje ne zgodi zaradi napak pri nameščanju.

4.2.3. Vodenje sprememb na nivoju produkta

Čeprav vsaka stranka dobi popolno aplikacijo, torej vse elemente konfiguracije, se spremembe vseeno obravnava ločeno za vsako stranko. Vsako spremembo mora torej razvijalec vnesti kot več enakih sprememb, za vsako stranko po eno. To bo ob naraščajočem številu strank zahtevalo vedno več ponavljajočega dela. Postopek upravljanja sprememb je potrebno preurediti tako, da se sprememba obravnava kot sprememba aplikacije in ne kot sprememba več produkcijskih okolij.

Predlog izboljšanega postopka je torej, da razvijalec vnese eno spremembo, potem pa se ta sprememba distribuira vsem strankam. Na ta način se izognemo ponavljajočemu se delu, hkrati pa preprečimo možnost, da določeni stranki določene spremembe ne namestimo. Kot se proces trenutno izvaja se namreč večkrat zgodi, da razvijalec pošlje spremembo le eni od strank, drugim pa ne. Problem nastane, ko se k drugim strankam nameša nove spremembe, ki so odvisne od predhodnih, vendar nekatere od predhodnih niso nameščene, ker jih razvijalec ni poslal.

Zaradi enostavnejšega postopka vnosa sprememb bi se izboljšala produktivnost razvijalcev, hkrati pa bi se povečala zanesljivost nameščanja sprememb, saj bi predhodne spremembe bile nameščene oz. bi se vedelo, da še niso bile nameščene (glej poglavje 4.2.4). Po drugi strani bi to pomenilo, da bi vsaka stranka kar naenkrat pričela dobivati spremembe, ki niso posledica njenih zahtev ali napak. To težavo bi odpravili z uvedbo povezovanja sprememb (glej poglavje 4.2.5).

4.2.4. Nameščene spremembe in odvisnosti med spremembami

Če bi želeli pri nameščanju sprememb v dano okolje preveriti, ali so bile nameščene vse predhodno potrebne spremembe, bi morali v informacijskem sistemu za upravljanje konfiguracij voditi:

- odvisnosti med posameznimi spremembami (Hoek, Wolf, 2001; Jansen, Ballintijn, Brinkkemper, 2004),
- informacije o nameščenih spremembah za vsako okolje (Abran et al., 2004).

Tako bi za vsako spremembo, ki bi jo želeli namestiti v okolje, lahko hitro dobili seznam sprememb, ki morajo biti nameščene pred njo, potem pa za vsako od teh sprememb preverili, ali je že nameščena. Če bi vse potrebne spremembe bile nameščene, potem bi lahko nadaljevali z nameščanjem, sicer pa bi se postopek ustavil.

Vodenje odvisnosti med spremembami se lahko opravlja ročno oz. delno avtomatizirano na podlagi elementov konfiguracije (Jansen, 2005). Za vsak element konfiguracije, ki ga sprememba vsebuje, morajo namreč biti naložene vse spremembe, ki vsebujejo eno od predhodnih verzij tega elementa. Če bi razvijalec ugotovil, da je sprememba odvisna še od kakšne druge spremembe, bi ročno vnesel še to odvisnost. S povečevanjem števila elementov, sprememb in medsebojnih povezav pa bo nujna popolna avtomatizacija (Jansen, Ballintijn, Brinkkemper, 2005a).

Vodenje informacij o nameščenih spremembah je ravno tako možno opravljati ročno, dokler se ročno opravlja tudi nameščanje sprememb – ob vsaki namestitvi bi bilo potrebno to zavesti v informacijskem sistemu. Število potrebnih ročnih vnosov se lahko zmanjša z združevanjem sprememb v izdaje: ko se izdaja označi kot nameščena, pomeni da so bile nameščene vse spremembe te izdaje. Ob uvedbi avtomatskega nameščanja sprememb pa se lahko v informacijski sistem avtomatsko zapisuje tudi podatke o nameščanju.

Ta izboljšava bi bistveno vplivala na nameščanje sprememb, ki bi tako postalo bolj odporno na napake zaradi nameščanja sprememb v napačnem vrstnem redu. Posledično bi se dosegla večja učinkovitost razvijalcev, ki se ne bi več ukvarjali z napakami zaradi neustreznega nameščanja. Manj bi bilo tudi izgubljenega časa zaradi ponovnega nameščanja že nameščenih sprememb. Pogosto se namreč ena in ista sprememba namesti večkrat, ker je trenutno bolj enostavno namestiti vse še enkrat, kot pa preveriti, katere spremembe so in katere še niso nameščene.

4.2.5. Povezava sprememb v izdaje

Ob naraščanju števila sprememb bi bilo smiselno uvesti pojem izdaje, kot ga definirajo različna ogrodja, ki izdajo smatrajo kot nek skupek sprememb. Spremembe lahko v izdajo združujemo po različnih kriterijih – lahko gre za planiranje izdaj na nek časovni rok, npr. na vsake tri mesece, lahko gre za izdajo, ki združuje funkcionalno povezane elemente konfiguracije, lahko gre za razvoj novih modulov, ki so med seboj odvisni (Microsoft Solutions for Management: Release Management, 2007).

Združevanje sprememb v izdaje bi nam koristilo predvsem kot odgovor na večje število sprememb, ki bi jih stranke dobivale zaradi predloga po vodenju sprememb na nivoju produkta. Neko spremembo, ki jo je zahtevala določena stranka, sedaj pošljemo le tej stranki, ostalim pa ne. Ko nato drugim strankam pošiljamo nove spremembe, ki so odvisne od prej narejene spremembe, moramo sestaviti takšno spremembo, ki zajema tudi že prej narejene, a še ne poslane spremembe. Tako preprečimo nepotrebno pošiljanje sprememb strankam, ki sprememb niso zahtevale.

Sedaj pa bi se lahko takšne spremembe akumulirale v našem sistemu na nivoju produkta, a hkrati bi se za vsako stranko posebej vedelo, ali je sprememba že bila nameščena ali ne. V trenutku, ko bi neka sprememba za stranko bila odvisna od ene od vnesenih in še ne nameščenih sprememb, bi se pripravila izdaja, ki bi vsebovala konsistenten nabor sprememb. Torej bi se postopek odvijal tako kot do sedaj, le da se neposlane spremembe ne bi vodile v glavah razvijalcev ali na njihovih delovnih postajah, ampak centralizirano v informacijskem sistemu. Ker bi povezave med spremembami bile zapisane (poglavje 4.2.4.), bi bilo enostavno sestaviti konsistenten nabor soodvisnih sprememb v enotno izdajo za določeno stranko (Jansen, Brinkkemper, 2005).

Napake zaradi nenameščenih odvisnih sprememb se sedaj odkriva in rešuje z uporabo različnih testnih okolij, kar pa zahteva dodaten čas, saj je ob vsaki takšni napaki najprej potrebno ugotoviti, kaj še ni nameščeno, nato poslati dodatno spremembo, osebe, odgovorne za testiranje, pa morajo testni scenarij še enkrat preveriti. Prihaja do nepotrebnega obremenjevanja razvijalcev, testerjev, uporabnikov in oseb, ki spremembe nameščajo.

Ta izboljšava je po eni strani nujna, če želimo uvesti izboljšavo »vodenje sprememb na nivoju produkta« in hkrati ne preobremenjevati strank z velikim številom sprememb. Po drugi strani ta izboljšava le informatizira to, kar že sedaj opravljamo, pa ni nikjer eksplicitno zapisano. Postopek bi torej postal bolj pregleden in robusten, tudi v odsotnosti razvijalca bi se vedelo, kaj mora biti nameščeno, da bo izdaja celovita. Zaradi manjšega števila napak bi se zmanjšala količina dela, potrebnega za odpravo teh napak.

4.2.6. Pregled sprememb/izdaj

Trenutno se v informacijskem sistemu sicer vodijo spremembe, a zaradi ostalih izboljšav bo potrebno voditi več informacij o vsaki spremembi oz. uvesti pojem izdaje in zanje voditi več informacij. Predvsem je smiselno, da se za vsako izdajo vodijo podatki o njenem nameščanju v razna okolja, opravljenih testiranjih v teh okoljih in avtorizaciji za nameščanje v naslednja okolja. Tako bi recimo vedeli, kdaj je bila neka izdaja nameščena v testno okolje, kdaj in kdo jo je tam preveril ter kdaj in kdo je dal avtorizacijo, da se izdaja lahko namesti v produkcijsko okolje.

Tako In2 kot naše stranke bi tako lahko učinkovito identificirali izdaje, ki jih je potrebno še testirati, in izdaje, ki so že pretestirane, a še niso naložene v produkcijsko okolje. Trenutno je to možno ugotoviti le iz tekstovnega dopisovanja, možnosti učinkovitega nadzora torej ni. Kot že omenjeno, si je ena od naših strank že razvila ločeno aplikacijo, kjer spremljajo podobne informacije, a z naraščanjem števila strank mora takšna funkcionalnost obstajati za vsako našo stranko in biti hkrati integrirana s procesom vodenja sprememb in izdaj. Logično mesto za vodenje takšnih informacij je torej obstoječi informacijski sistem za vodenje sprememb.

4.2.7. Avtomatsko nameščanje sprememb

Ker je nameščanje sprememb dokaj rutinsko opravilo, ki se trenutno izvaja ročno, bi ga bilo smiselno avtomatizirati. Večinoma gre za poganjanje baznih skript in kopiranje izvršilnih datotek obrazcev in poročil na aplikacijski strežnik, za kar bi lahko razvili posebno orodje, ki bi na podlagi sprememb ali izdaj, vnesenih v informacijski sistem, le te avtomatsko namestil. Pogoj bi seveda bila avtorizacija nameščanja spremembe/izdaje s strani odgovorne osebe. Ker nameščanja ne bi več opravljal človek, bi se nameščanje lahko izvedlo izven delovnega časa, npr. ponoči. Orodje za avtomatsko nameščanje sprememb bi lahko informacije o poteku postopka nameščanja (npr. trajanje nameščanja posamezne spremembe, morebitne napake, ipd.) vpisal nazaj v informacijski sistem, kjer bi lahko upravitelj nameščanja pregledoval informacije o opravljenih nameščanjih. Orodje bi bilo torej dvosmerno povezano s sistemom za vodenje konfiguracij (Hoek, 2001).

Pred nameščanjem bi tako orodje lahko preverilo, ali so res nameščene vse potrebne predhodne spremembe, po namestitvi pa bi lahko preverilo, ali so spremembe bile uspešno nameščene. Ob napaki bi orodje lahko poslalo elektronsko sporočilo ali SMS na ustrezen naslov. Nameščanje baznih sprememb se lahko zaustavi, če je npr. procedura, ki jo želimo spremeniti, pravkar v uporabi. Orodje bi lahko pred nameščanjem opravilo kontrole in v takšnem primeru preprečilo nameščanje ali bi ga zakasnilo za nekaj časa.

Razvoj takšnega orodja bi sicer zahteval nekaj časa, a z naraščanjem števila strank in sprememb bi se vloženi čas obrestoval. Oseba, odgovorna za nameščanje, bi bila osvobojena rutinskega ročnega dela, zaradi avtomatizacije bi se zmanjšala verjetnost napak, spremembe bi se lahko nameščalo hkrati v več okolij, ob raznih za človeka »čudnih« urah, ob tem pa bi se lahko zajelo in v informacijski sistem shranilo več uporabnih informacij, ki se jih sedaj ne (Jansen, Ballintijn, Brinkkemper, 2005).

4.3. Vrednostna analiza predlaganih izboljšav

Izboljševanje slabega upravljanja izdaj je zelo zahteven podvig, ki lahko zahteva veliko sredstev v obliki človeškega časa, denarja za orodja in prinaša veliko sprememb v delovanje organizacije. Po drugi strani pa lahko ignoriranje težav upravljanja izdaj vodi v drage napake, ki lahko zelo negativno vplivajo na poslovno uspešnost organizacije (Aiello, 2007; Jansen, Brinkkemper, 2006). Stroški vzpostavitve upravljanja izdaj so torej neprimerno nižji od

potencialnih stroškov napak zaradi neprimernega planiranja, upravljanja in kontroliranja novih izdaj.

A napake in odprava posledic napak niso edino, kar podjetje In2 sili k uvedbi upravljanja izdaj. Menim, da podjetje ne bo sposobno uspešno delovati, dokler ne bo sposobno obvladati velikega števila sprememb programske opreme ne da bi trpela kakovost. Večje število strank namreč pomeni več ljudi, ki sodeluje pri razvoju programske opreme, več sprememb in več izdaj v izbranem časovnem obdobju. Raziskave kažejo, da so podjetja z ustreznim upravljanjem izdaj sposobna obvladati veliko število kupcev, brez ustreznih postopkov pa lahko učinkovitost in uspešnost podjetja upade (Jansen, 2006; Jansen, Brinkkemper, Ballintijn, 2005).

Čeprav raziskave kažejo, da si v splošnem uporabniki ne želijo pogostega nameščanja novih izdaj programske opreme (Jansen, Brinkkemper, 2006a), lahko pogosto in kvalitetno nadgrajevanje informacijskega sistema predstavlja konkurenčno prednost organizacije. Procesov, ki so zapleteni, zamudni in povzročajo veliko napak, si nihče ne želi izvajati pogosto. Z optimiziranimi procesi upravljanja izdaj pa pogosto nameščanje novih izdaj postane sprejemljivo (Storm, 2006).

Ugotavljanje količine potrebnih sredstev za uvedbo vsake predlagane izboljšave in primerjanje porabljenih sredstev s pričakovano donosnostjo je zato skoraj nesmiselno, kajti neuvedba predlaganih izboljšav lahko zaradi neučinkovitega obvladovanja procesov v najslabši možni varianti pomeni konec podjetja. Kljub temu pa se da vsaj približno oceniti in primerjati med seboj različne predloge, tako z vidika potrebnih sredstev kot z vidika pričakovanih koristi.

Stroški pri uvedbi upravljanja izdaj bodo nastali kot posledica:

- razvoja in definicije praks ter postopkov upravljanja izdaj,
- izobraževanja o politiki izdaj, postopkih in orodjih,
- osebja, potrebnega za razvoj in uporabo postopkov in orodij,
- strojne in programske opreme.

Glede na predvidene izvire stroškov in ob upoštevanju možnosti lastnega razvoja internega informacijskega sistema so torej glavni stroški ljudje, zato sem za predloge podal oceno potrebnega časa, ki bi ga en človek z ustreznim znanjem porabil za implementacijo celotnega predloga.

Tabela 2: predlagane izboljšave.

Predlagana izboljšava	**Ocenjeni stroški**	**Pričakovane koristi**
Politika izdaj	2 meseca dela	Povečano razumevanje Merjenje uspešnosti
Obveščanje o spremembah	1 mesec dela	Večja seznanjenost Realna pričakovanja
Vodenje sprememb na nivoju produkta	2 meseca dela	Povečana zanesljivost Višja produktivnost
Nameščene spremembe in odvisnosti med spremembami	2 meseca dela	Povečana zanesljivost Višja produktivnost
Povezava sprememb v izdaje	1 mesec dela	Obvladovanje večjega števila sprememb
Pregled sprememb/izdaj	1 mesec dela	Učinkovit pregled zgodovinskih podatkov
Avtomatsko nameščanje sprememb	2 meseca dela	Povečana zanesljivost

Vir: lasten.

Skupaj bodo predlagane izboljšave prinesle:

- Boljšo kvaliteto storitev zaradi konsistentnega procesa izdajanja.
- Višji odstotek uspešno izdane programske opreme.
- Zmanjšano število motenj v storitvah.
- Povečano produktivnost vpletenega osebja.
- Povečano zaupanje uporabnikov v informacijski sistem.
- Učinkovit pregled zgodovinskih podatkov o vsaki izdaji.
- Prihranek zaradi centraliziranega nadzora nad programsko opremo.
- Znano stanje programske opreme v vsakem nadzorovanem okolju.

Tveganja in možni problemi, ki se lahko pojavijo pri uvajanju upravljanja izdaj so:

- Pomanjkanje sredstev za uvedbo novih procesov upravljanja izdaj. S karseda malo spremembami je potrebno doseči začetne koristi, nato pa uvesti še ostale spremembe, ki zahtevajo več sredstev in sprememb.
- Odpor in nepripravljenost na spremembe vpletenih oseb. Ljudje so navajeni delati po obstoječih postopkih in jim novi procesi predstavljajo nepotrebno breme. Zato je potrebno razložiti koristi novih postopkov, pri uvajanju sprememb pa sodelovati z vpletenim osebjem.
- Posamezniki lahko poizkušajo zaobiti postopke upravljanja izdaj ali pa jih ne upoštevajo v celoti. Vpletene osebe je potrebno izobraziti, jim razložiti koristi in prepričati, da bodo novi postopki zagotovili višjo kvaliteto storitev. Kasneje je potrebno nadzirati procese in ob neupoštevanju dogovorov primerno ukrepati.
- Posameznike lahko premaga skušnjava in zaobidejo standardne procedure, da bi lahko namestili izredno izdajo, kar mora biti prepovedano in onemogočeno s pomočjo varnostnih pravil.

4.4. Plan uvedbe izboljšav

Predlagane izboljšave sem razvrstil v vrstni red uvedbe glede na enostavnost implementacije izboljšave, pričakovane koristi izboljšave in odvisnosti med posameznimi izboljšavami. Predlogi so razvrščeni v štiri skupine, pri čemer bi bilo potrebno predloge znotraj skupine uvesti hkrati, same skupine pa postopoma eno za drugo. Pri uvedbi vsake skupine predlogov bi bilo potrebno izobraziti vpleteno osebje.

4.4.1. Skupina 1 – Politika, smernice, komunikacija

V prvem koraku bi definirali politiko izdaj, nakazali smernice pri izdajanju in v informacijski sistem vgradili merjenje ključnih kazalcev uspešnosti. Informacijski sistem bi dogradili s pošiljanjem elektronskih sporočil pred uvedbo in po uvedbi sprememb.

Prvi korak bi postavil temelje za ostale spremembe in sodelujočim nakazal pričetek uvajanja sprememb.

4.4.2. Skupina 2 – Vodenje sprememb

V drugem koraku bi razširili nabor informacij, ki jih vodimo za vsako spremembo:

- odvisnosti med posameznimi spremembami,
- informacije o namestitvi spremembe v posamezno okolje,
- informacije o testiranju sprememb in avtorizaciji za nameščanje.

Te informacije bi izkoriščali pri nameščanju sprememb, saj bi natančno vedeli, katere spremembe so prestale testiranje v določenem okolju in se jih lahko namesti v naslednje okolje, hkrati pa zanje ne obstaja odvisna sprememba, ki še ni bila preverjena ali nameščena. Tako bi zmanjšali število napak, ki nastajajo zaradi nedokumentiranih odvisnosti sprememb, zaradi nameščanja sprememb v napačnem vrstnem redu in zaradi netestiranih sprememb.

Drugi korak bi torej že prinesel določen del pričakovanih koristi, čeprav bi bile spremembe še vedno ločene po strankah.

4.4.3. Skupina 3 – Vodenje izdaj

V tretjem koraku bi postopek upravljanja sprememb preuredili tako, da bi se spremembe aplikacije obravnavale na nivoju aplikacije in ne bi bile več ločene po strankah. Ker bi stranke občutile večje število sprememb in ker se nameščanje od stranke do stranke časovno razlikuje bi hkrati uvedli izdaje kot način združevanja sprememb.

V tretjem koraku bi še povečali produktivnost razvijalcev ter preprečili napake zaradi nenameščenih sprememb. Ta skupina sprememb bi podjetju In2 omogočila učinkovito upravljanje izdaj tudi ob povečanju števila strank.

4.4.4. Skupina 4 – Avtomatsko nameščanje izdaj

O razvoju orodja za avtomatsko nameščanje izdaj je smiselno razmišljati šele po uvedbi vseh ostalih predlogov. Orodje namreč potrebuje določene informacije, ki bi jih uvedli v drugem koraku, hkrati pa bi se vložena sredstva obrestovala šele pri večjem številu izdaj, ki bi jih bilo potrebno nameščati, torej ob povečanju števila strank. Zmanjšanje števila napak zaradi neustreznega nameščanja je do uvedbe takšnega orodja možno doseči z natančnim definiranjem postopka nameščanja.

Z uvedbo takšnega orodja bi se predvideno izboljševanje upravljanja izdajanja sicer zaključilo, z analizo uvedenih izboljšav pa bi lahko izmerili dejanske koristi, jih primerjali s pričakovanimi in določili nadaljnje izboljšave.

5. Zaključek

Potreba po razvoju novih IT storitev in spreminjanju obstoječih, je jasna. IT storitve predstavljajo ključ do uspeha mnogih podjetij, saj zagotavljajo konkurenčno prednost na vedno bolj zahtevnem trgu. Po drugi strani se od informacijske tehnologije pričakuje neprekinjeno delovanje, saj lahko že kratkotrajna motnja v delovanju predstavlja ogromno poslovno izgubo ali celo resno grožnjo obstoju marsikaterega podjetja. Torej je uspešen prenos programske opreme iz razvoja v produkcijo ključni element informacijskih storitev, za katere skrbijo vzdrževalci programske opreme. Celovito upravljanje prenosov skrbi za racionalnost, uspešnost in učinkovitost vseh aktivnosti v povezavi z izdajo in nameščanjem programske opreme, kar podjetjem omogoča hitro prilagajanje konstantnim spremembam.

Podjetje In2 uspešno deluje na domačem in se hkrati širi na tuje trge, kar potrjuje njegovo uspešnost pri razvoju programske opreme. Informacijske sisteme, razvite znotraj In2, vsakodnevno uporablja več tisoč uporabnikov velikih podjetij, kot so zavarovalnice in banke. Pri analizi izdajanja programske opreme sem ugotovil, da je ključna strategija podjetja pri obvladovanju morebitnih težav enostavnost. Kljub temu, da je zavarovalništvo kompleksno področje, sta tako arhitektura aplikacije kot razvoj posameznih modulov dokaj enostavna, zato se lahko razvoj aplikacije in njenih modulov osredotoča na zadovoljevanje poslovnih potreb naših strank.

Prvemu vprašanju magistrskega dela (*Katere so svetovno uveljavljene najboljše prakse izdajanja programske opreme?*) je posvečeno drugo poglavje, v katerem je preučena raznovrstna literatura. Po seznanitvi z ogrodjema ITIL in COBIT ter ostale literature sem analiziral aktualne procese izdajanja programske opreme podjetja In2 ter tako odgovoril na drugo vprašanje magistrskega dela (*Kakšni procesi, povezani z izdajanjem programske opreme, so trenutno v uporabi v podjetju In2?*). Ugotovil sem, da bi uvedba nekaterih dobrih praks lahko izboljšala procese izdajanja programske opreme podjetja In2. Ker je upravljanje izdaj močno povezano z upravljanjem sprememb in konfiguracij, je smiselno, da se dogradi interni informacijski sistem. Tako bi bile na enem mestu zbrane in povezane informacije o konfiguracijah, problemih, spremembah in izdajah.

Incidenti, povezani z izdajanjem programske opreme, ki so se v podjetju zgodili v zadnjem letu, so bili moje izhodišče za ugotavljanje možnih izboljšav, saj sem za vsak incident ugotovil, katere izboljšave bi preprečile pojavljanje takega tipa incidentov. Predloge izboljšav sem ovrednotil glede na sredstva, potrebna za implementacijo, in prednosti, ki jih predlogi prinašajo, ter zanje predlagal vrstni red njihove uvedbe, s čemer sem odgovoril na tretje vprašanje magistrskega dela (*Po katerih praksah bi se lahko zgledovali, jih poizkušali vpeljati ali prilagoditi našim potrebam in tako izboljšati naše procese?*). Menim, da lahko uvedba predlaganih izboljšav odpravi omejitvene dejavnike v aktivnostih izdajanja programske opreme, izboljša kvaliteto storitev podjetja In2, poveča produktivnost zaposlenih ter tako omogoči širjenje poslovanja podjetja.

6. Literatura in viri

6.1. Literatura

1. Alain Abran et al.: Guide to the Software Engineering Body of Knowledge. IEEE, Los Alamos, California, ZDA, 2004. 202 str.
2. Alain April et al.: Software Maintenance Maturity Model (SMmm): The software maintenance process model. University of Kentucky, Kentucky, ZDA, 2004. 30 str.
3. André Van der Hoek: Integrating Configuration Management and Software Deployment. University of California, Irvine, California, ZDA, 2001. 4 str.
4. André Van der Hoek, Alexander L. Wolf: Software release Management for Component Based Software. University of California, Irvine, California, ZDA, 2001. 24 str.
5. André Van der Hoek et al.: Software Release Management. Department of Computer Science, University of Colorado, Boulder, Colorado, ZDA, 1996. 20 str.
6. Antonio Carzaniga et al.: A Characterization Framework for Software Deployment Technologies. Department of Computer Science, University of Colorado, Boulder, Colorado, ZDA, 1998. 24 str.
7. Eelco Dolstra, Eelco Visser, Merijn de Jonge: Imposing a Memory Management Discipline on Software Deployment. 26th International Conference on Software Engineering, IEEE, 2004. str. 583–592.
8. Eelco Dolstra: The Purely Functional Software Deployment Model. PhD thesis, University of Utrecht, Nizozemska, 2006. 281 str.
9. Gerco Ballintijn: A Case Study of the Release Management of a Health-care Information System. Proceedings of the IEEE International Conference on Software Maintenance, IEEE, 2005. 10 str.
10. Hans Sassenburg: Design of a Methodology to Support Software Release Decisions. PhD thesis, University of Groningen, Nizozemska, 2005. 302 str.
11. IT Governance Institute: COBIT 4.1: Framework, Control Objectives, Management Guidelines, Maturity Models. ISACA, Rolling Meadows, ZDA, 2005. 213 str.
12. Justin R. Erenkrantz: Release Management Within Open Source Projects. Proceedings of the 3rd Workshop on Open Source Software Engineering, 2003. str. 51-55.
13. Michael E. Bays: Software Release Methodology. Prentice Hall PTR, 2003. 247 str.
14. Office of Government Commerce: Itil Service Support. Stationery Office Books, London, Velika Britanija, 2000. 326 str.
15. Office of Government Commerce: Itil Service Transition. Stationery Office Books, London, Velika Britanija, 2007. 261 str.
16. Peter Schuh: Agility and the Database. 3rd International Conference on XP and Agile Processes in Software Engineering, 2002. str. 125-129.

17. Remy Jansen, Gerco Ballintijn, Sjaak Brinkkemper: Software release and deployment at Exact: a case study report. Centrum voor Wiskunde en Informatica, Amsterdam, Nizozemska, 2004. 40 str.
18. Richard S. Hall et al.: The software Dock: A Distributed, Agent-based Software Deployment System. Department of Computer Science, University of Colorado, Boulder, Colorado, ZDA, 1997. 23 str.
19. Richard S. Hall, Dennis Heimbigner, Alexander L. Wolf: A cooperative Appoach to Support Software Deployment Using the Software Dock. Proceedings of the International Conference on Software Engineering, IEEE, 1999. str 174-183.
20. Robert Arthur Smith: Analysis and design for a next generation software release management system. Master's thesis, University of Colorado, Boulder, Colorado, ZDA, 1999. 114 str.
21. Sjaak Brinkkemper, Paul Klint: Intelligent Software Knowledge Management and Delivery. Centrum voor Wiskunde en Informatica, Amsterdam, Nizozemska, 2003. 15 str.
22. Slinger Jansen: Alleviating the Release and Deployment Effort of Product Software by Explicitly managing Component Knowledge. Proceedings of the Workshop on Development and Deployment of Product Software, 2005, str. 21-30.
23. Slinger Jansen: Improving the Customer Configuration Update Process by Explicitly managing Software Knowledge. Centrum voor Wiskunde en Informatica, Amsterdam, Nizozemska, 2006. 4 str.
24. Slinger Jansen, Gerco Ballintijn, Sjaak Brinkkemper: A Process Model and Typology for Software Product Updaters. Proceedings of the 9th European Conference on Software Maintenance and Reengineering, 2005. str. 265-274.
25. Slinger Jansen, Gerco Ballintijn, Sjaak Brinkkemper: Release and Deployment at Planon: A Case Study. Centrum voor Wiskunde en Informatica, Amsterdam, Nizozemska, 2005a. 20 str.
26. Slinger Jansen, Sjaak Brinkkemper: Definition and Validation of the Key Process Areas of Release, Delivery and Deployment for Product Software Vendors: turning the ugly duckling into a swan. Utrecht University, Utrecht, Nizozemska, 2005. 11 str.
27. Slinger Jansen, Sjaak Brinkkemper: Evaluating the Release, Delivery, and Deployment Process of Eight Large Product Software Vendors applying the Customer Configuration Update Model. Utrecht University, Utrecht, Nizozemska, 2006. 4 str.
28. Slinger Jansen, Sjaak Brinkkemper: Modelling Deployment using Feature Descriptions and State Models for Component-Based Software Product Families. 3rd International Working Conference on Component Deployment, 2005, 15 str.
29. Slinger Jansen, Sjaak Brinkkemper: Ten Misconceptions about Product Software Release management explained. First International Workshop on Software Product Management, IEEE, 2006a. 7 str.
30. Slinger Jansen, Sjaak Brinkkemper, Gerco Ballintijn: Integrated Development and Maintenance of Software Products to Support Efficient Updating of Customer

Configurations: A Case Study in Mass Market ERP Software. Proceedings of the 21st International Conference on Software Maintenance, IEEE, 2005. 10 str.
31. Software Engineering Institute: CMMI for Development. Pittsburgh, ZDA, 2006. 573 str.
32. Steve Beattie et al.: Timing the Application of Security Patches for Optimal Uptime. Proceedings of LISA 2002: 16th Systems Administration Conference, 2002. 110 str.
33. Tijs van der Storm: Lightweight incremental application upgrade. Centrum voor Wiskunde en Informatica, Amsterdam, Nizozemska, 2006. 9 str.

6.2. Viri

1. Aiello Bob: When the build fails - the Business of Software Development [URL: http://www.cmcrossroads.com/articles/behaviorally-speaking/when-the-build-fails-%11-the-business-of-software-development.html] 24.9.2007
2. ISO/IEC 12207.0-1996: Standard for Information Technology - Software life cycle processes. 1998. 87 str.
3. ISO/IEC TR 15504-2: Information Technology - Software process assessment, Part 2: A reference model for processes and process capability. 1998. 46 str.
4. ISO/IEC 20000-1: Information Technology – Service management, Part 1: Specification. 2005. 24 str.
5. ISO/IEC 20000-2: Information Technology – Service management, Part 2: Code of practice. 2005. 44 str.
6. Microsoft TechNet, Microsoft Solutions for Management: Release Management [URL: http://www.microsoft.com/technet/itsolutions/cits/mo/smf/smfrelmg.mspx] 9.9.2007
7. Sadalage Pramod: Database Agility [URL:http://tech.groups.yahoo.com/group/agileDatabases/files/AgileDatabases_Presentation.pdf] 9.9.2007
8. Wikipedia: Release management [URL: http://en.wikipedia.org/wiki/Release_Management] 9.9.2007
9. Wikipedia: Software release life cycle [URL: http://en.wikipedia.org/wiki/Software_release] 9.9.2007