5 IZDELAVA PROCESA

Eksekucija procesov je veliko več kot upoštevanje korakov po determiniranem vrstnem redu. Proces potrebuje podporo infrastrukture, ki omogoča njegove omejitve in hkrati dovolj fleksibilnosti, da ne ovira kreativnost uporabnikov (Jablonski 2010, 393). Izdelava procesa pomeni definiranje procesnega modela, ki predstavlja zgradbo sistema (Jablonski 2010, 394). Za definiranje procesa je potrebna globoka metodologija skupaj z močnim in ekspresivnim jezikom procesnega modeliranja (Jablonski 2010, 393). Jeziki modeliranja so bili izdelani na podlagi programerskih jezikov, Petrijevih mrež in jezikov, ki temeljijo na pravilih (García-Borgoñón in dr. 2014, 104). Uporabnost jezikov modeliranja se razlikuje glede na njihovo paradigmo. Nekateri so bolj primerni za nadzorovanje procesa, objektno-orientirani za nadzor funkcionalnosti in kode, podatkovni pa za nadzor in definiranje podatkovnih baz (Osterweil 2011, 295). Procesi sami so tudi programska oprema (Osterweil 2011, 291). To pomeni, da lahko procese predstavimo na podoben način kot programsko opremo. Pri izgradnji se pogosto uporablja termin programiranje procesov pri čemer se beseda programiranje uporablja za implikacijo obsežnosti programskega inženiringa, ne le kodiranja. Čeprav strokovnjaki zagovarjajo algoritemski zapis procesov je potrebno upoštevati dejstvo, da razvoj programske opreme zajema ustvarjalne vidike, ki so inherentno ne algoritmični. Procesi so deležni posebnih, nepričakovanih iregularnosti, ki onemogočajo slednje algoritemskim predpisom (Osterweil 2011, 293).

Programsko opremo lahko predstavimo s podatki, kar nam omogoča apliciranje entiteto-relacijskega[[1]](#footnote-1) modeliranja. Entitete nam predstavijo podatke in relacije njihove medsebojne odvisnosti. Zaradi elementarne narave ER koncepta modeliranja ga je mogoče aplicirati na katerekoli podatkovne strukture (Jablonski 2010, 395).

Procesi so časovno razporejene zbirke aktivnosti, ki so usmerjene v cilj ali izid (Peters 2008, 103). Podrobneje jih lahko opišemo kot zbirko deloma decidiranih korakov s podskupino medsebojno povezanih artefaktov, fizičnih in virtualnih virov, organizacijskih struktur in omejitev z namenom proizvajanja in vzdrževanja programske opreme (García-Borgoñón in dr. 2014, 104). Osnovna oblika vseh življenjskih ciklov je prikaz napredka od problema do rešitve (Peters 2008, 108). Potrebno je potrebno identificirati, načrtovati, organizirati in spremljati vse aktivnosti, ki so potrebne za izgradnjo novega sistema (Satzinger, Jackson, in Burd 2011, 5). Življenjski cikel razvoja identificira vse aktivnosti, ki so potrebne za izgradnjo, zagon in vzdrževanje informacijskih sistemov. Čeprav je mnogo variacij procesov znotraj življenjskih ciklov veljajo naslednje za osnovne;

1. Identifikacija problema ali potreb in odobritev za nadaljevanje,
2. načrtovanje in spremljanje projekta,
3. odkrivanje in razumevanje podrobnosti problema ali potreb,
4. oblikovanje sistemskih komponent, ki rešijo problem ali zadovoljijo potrebam,
5. izgradnja, testiranje in integracija sistemskih komponent,
6. testiranje celotnega sistema in prenos v delovno okolje (Satzinger, Jackson, in Burd 2011, 6).

Modeli so zgrajeni na podlagi inženiringa imperativov in inženiringa oblikovanja (Rolland in Ufr 1994, 1). Inženiring imperativov vključuje identifikacijo problemov ali potreb in oblikovanje specifikacije sistema. Na podlagi slednje inženiring oblikovanja poskrbi za izgradnjo in implementacijo v delovni sistem (Rolland in Ufr 1994, 2).

Procesni model ustreza načinu dela, ki ga predpisuje metodologija v uporabi. Model lahko razumemo tudi kot koncept načrta. Za oblikovanje takšnih načrtov je potrebna uporaba abstrakcije procesa, ki jo predstavlja meta model (slika 5.1). Pri uporabi procesnega modela v praksi je ta instanca meta modela (Rolland in Ufr 1994, 3). Pri procesnem modeliranju je velikokrat spregledano dejstvo, da je procesni model abstraktna predstavitev procesa in kot taka predstavlja proces iz drugačne perspektive, ki lahko definira potek dela kateri še ni v praksi (Peters 2008, 105).

Slika 5.1 nivoji abstrakcije pri procesnem modeliranju

  
vir: (Rolland in Ufr 1994, 3)

Meta model predstavlja ključne aktivnosti, ki v razvojnem procesu razlagajo kdaj se kaj zgodi in zakaj (Rolland in Ufr 1994, 10).

Procesni model definira zbirko generičnih aktivnosti. Metodološkim inženirjem pomaga definirati način dela določene metodologije, ki jo je potrebno razviti. Način dela podane metodologije pa bo pridobljena z inicializacijo generičnih aktivnosti procesnega modela (Rolland in Ufr 1994, 10).

Inženiring imperativov sestavljajo situacije, odločitve, argumenti in dejanja. Meta model prepozna kontekst v katerem se pojavi odločitev, potreba po argumentaciji in njen izid. Nato poveže proces in končen izdelek s predstavitvijo imperativnih artefaktov kot vhodnih in izhodnih informacij procesa (Rolland in Ufr 1994, 11).

Agilno modeliranje definira štiri prakse, ki podpirajo iterativni in inkrementalni prostop k modeliranju:

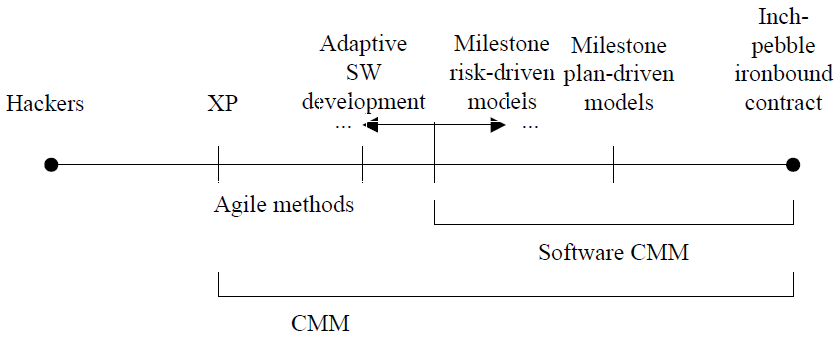
1. Apliciranje pravih artefaktov,
2. izdelava več vzporednih modelov,
3. iteracija do naslednjega artefakta,
4. izvajanje manjših inkrementov (Ambler 2002, 45).

Artefakte predstavljajo UML grafikon stanja, izvorna koda, diagram poteka podatkov, primeri uporabe in drugi. Vsak artefakt ima svoje prednosti pri uporabi v določenih situacijah (Ambler 2002, 45). V nekaterih primerih je bolj efektivno uporabiti diagram kot pa napisati 1024 vrstic kode. Pri modeliranju je pomembno razumevanje kdaj je smotrno uporabiti določen artefakt in kdaj ne (Ambler 2002, 46).

Vzporedno modeliranje omogoča simultani zajem informacij več različnih artefaktov (Ambler 2002, 48). V primeru, da informacije postanejo neprimerne za določen artefakt se izvede iteracija do naslednjega artefakta. S tem se informacije prenesejo na drug artefakt in posledično omogočijo napredovanje v procesu. Proces se izvaja inkrementalno, kar predstavlja temelj agilnega modeliranja. Agilno modeliranje si prizadeva k fragmentaciji kompleksnejših nalog v manjše obvladljive entitete, ki se izvajajo v krajših intervalih. Daljši intervali se lahko zgodijo vendar so zgolj izjeme. Daljše kot je obdobje brez povratne informacije večja je nevarnost, da je bilo opravljeno delo zaman (Ambler 2002, 51). Daljša iteracija ne pomeni, da je bilo narejeno več dela. Agilni procesi se fokusirajo na efektivne ure razvoja kot pa njihovo število (Unger in Novak 2011, 180).

Slika 5.2 prikazuje spekter različnih pristopov na kateri so hekerji postavljeni na levo in podrobnih mejnikov na desno (Abrahamsson in dr. 2017, 14).

Slika 5.2 spekter pristopov po Boehm



Vir: (Abrahamsson in dr. 2017, 14)

Ena metodologija ne more ustrezati celotnemu spektru različnih projektov. Vodstvo projekta mora identificirati naravo projekta in nato izbrati primerno razvojno metodologijo (Abrahamsson in dr. 2017, 14).

Cockburn temeljna pravila agilnega razvoja definira kot lahka vendar učinkovita z uporabo človeško in komunikacijsko usmerjenih pravil. Predlagal je naslednje prijeme katerih uporaba povečuje uspešnost projekta;

1. Dva do osem razvijalcev v eni sobi,
2. uporabniki na lokaciji,
3. kratki inkrementi,
4. avtomatizirani regresijski testi,
5. izkušeni razvijalci (Abrahamsson in dr. 2017, 15).

Miller predlaga naslednje karakteristike, ki omogočajo skrajšanje življenjskega cikla projekta;

1. Modularnost na stopnji razvoja,
2. iteracije s kratkimi cikli za omogočanje hitre verifikacije in popravkov,
3. časovno vezani iteracijski cikli od 1 do 6 tednov,
4. varčevanje z namenom odstranjevanja nepotrebnih aktivnosti,
5. prilagodljivost na morebitna tveganja,
6. inkrementalni pristop, ki omogoča apliciranje funkcionalnosti v kratkih korakih,
7. konvergenten pristop za zmanjšanje tveganj,
8. naklonjenost ljudem nad tehnologijo,
9. delo na način sodelovanja in komuniciranja (Abrahamsson in dr. 2017, 16).

5.1 OPTIMIZACIJA PROCESA

Optimizacija pomeni razumevanje trenutnega procesa in njegovo spreminjanje z namenom povečanja kvalitete produktov, zmanjšanja stroškov in časa razvoja. Za optimizacijo se uporabljata zrelostni in agilni pristop (Sommerville 2010, 706).

*Zrelostni pristop* je fokusiran na izboljšave procesov, vodenje projektov in implementacijo boljših inženirskih praks v organizacijo. Nivo zrelosti odraža obseg adaptacije tehničnih in vodstvenih praks v proces razvoja programske opreme. Cilj pristopa je izboljšanje kvalitete produktov in predvidljivost procesa (Sommerville 2010, 706).

*Agilni pristop* je fokusiran na iterativni razvoj s krčenjem presežkov v razvojnem procesu. Temeljne karakteristike agilnih metod so hitre izdaje funkcionalnosti in visoka odzivnost na spremembe (Sommerville 2010, 706).

Zrelostni pristop je zakoreninjen v načrtno usmerjenem razvoju in navadno proizvede veliko presežkov v smislu, da definira več aktivnosti kot je potrebno. Agilni pristop pa se fokusira na kodo in se namensko izogiba formalnostim in dokumentaciji (Sommerville 2010, 706). Za večje projekte in kompleksne sisteme je priporočena uporaba zrelostnega pristopa, medtem, ko je velika verjetnost, da bo izbira agilnega pristopa manjšim in srednje velikim projektom predstavljala najboljšo strategijo optimizacije procesa (Sommerville 2010, 706).

1. (angl.) Entity-relationship – ER. [↑](#footnote-ref-1)