Procesni model ni enostaven linearen model ampak vključuje potrebo po povratnih informacijah od koraka do koraka (Sommerville 2010, 31).

Parts of the system which are difficult to

specify in advance, such as the user interface, should always be developed using an

incremental approach.(Sommerville 2010, 30)

The waterfall model forms the foundation of many software development

methodologies in use today. However, it has some limitations and, if followed

too strictly, can lead to the following types of problems: (Lethbridge in Laganiere 2005, 429)

V uvod

Ne obstaja idealen proces in večina organizacij je razvila svoje procese razvoja (Sommerville 2010, 28). Čeprav ne obstaja idealen proces, velja, da v veliko organizacijah obstaja prostor za izboljšave. Procesi lahko uporabljajo zastarele tehnike ali pa ne izkoriščajo najnovejših in najboljših praks. Veliko organizacij še vedno ne uporablja najnovejših pristopov pri razvoju lastne programske opreme. Proces razvoja je lahko izboljšan s standardizacijo procesov. To vodi v izboljšano komunikacijo, manjšo porabo časa pri uvajanju in zmanjšanje stroškov avtomatiziranega vodenja. (Sommerville 2010, 29).

Pressman (2014) deli modele glede na njihovo strukturo in namembnost na predpisujoče, specializirane, enotne, osebne in ekipne.

Navadno je najdlje trajajoča aktivnost vzdrževanje programske opreme (Sommerville 2010, 31).

Življenjski cikel razvoja programske opreme predstavlja organizacijsko shemo procesa razvoja (Glass 2002).

Proces razvoja programske opreme je skupek aktivnosti, ki vodijo v produkcijo programske opreme. (Pressman in Maxim 2014, 15).

Inženiring procesa razvoja programske opreme omogoča racionalen in časovno sprejemljiv razvoj programske opreme (Pressman in Maxim 2014, 15).

Also, the earlier admonitions regarding changes were based

on a Big Bang style of development. That is, work went on for months with no running code,

and then, at some point, a build happened. (Peters 2008, 117).

This software was developed by large teams working for different companies. Teams

were often geographically dispersed and worked on the software for long periods of

time. An example of this type of software is the control systems for a modern aircraft,

which might take up to 10 years from initial specification to deployment. These plandriven

approaches involve a significant overhead in planning, designing, and documenting

the system. This overhead is justified when the work of multiple development teams

has to be coordinated, when the system is a critical system, and when many different

people will be involved in maintaining the software over its lifetime.

However, when this heavyweight, plan-driven development approach is applied

to small and medium-sized business systems, the overhead involved is so large that it

dominates the software development process. More time is spent on how the system

should be developed than on program development and testing. As the system

requirements change, rework is essential and, in principle at least, the specification

and design has to change with the program.

Dissatisfaction with these heavyweight approaches to software engineering led a

number of software developers in the 1990s to propose new ‘agile methods’. (Sommerville 2010, 58).

These

allowed the development team to focus on the software itself rather than on its design

3.1 \_ Agile methods **59**

and documentation. Agile methods universally rely on an incremental approach to software

specification, development, and delivery. They are best suited to application development

where the system requirements usually change rapidly during the development

process. They are intended to deliver working software quickly to customers, who can

then propose new and changed requirements to be included in later iterations of the system.

They aim to cut down on process bureaucracy by avoiding work that has dubious

long-term value and eliminating documentation that will probably never be used. (Sommerville 2010, 59)

Tudi uvoddddddddddd:::::::LKJLČČKMČLKČPJHJIG

Čeprav je industrija razvoja iger sprva prevzela najboljše prakse tradicionalnega razvoja programske opreme je osnovna razlike v tem, da se razvoj iger osredotoča bolj na uporabniške izkušnje kot na produkt sam (O’Hagan, Coleman, in O’Connor 2014, 182). Razvoj igre je bolj podobno izdelku, ustvarjenem s prepletanjem aspektov, umetnosti, glasbe, programiranja, igranja, poslovnega upravljanja, integriranih v eno celoto. (Ramadan in Widyani 2013, 95). Potrebno je razširiti tradicionalne tehnike razvoja, da bi lahko podprli kreativni proces razvoja elektronskih iger („What went wrong? A survey of problems in game development“ 2017, 19). Za načrtovanje in upravljanje takšnih kompleksnih multidisciplinarnih projektov je potrebna metodologija, kjer ad hoc načini upravljanja ne pridejo v poštev (Aslan in Balci 2015, 307).

Proces video iger ni podrobno definiran. (McAllister in White 2015, 14). Dejstvo je, da je več študij narejenih na akademski strani, poročila s strani industrije iger pa se nahajajo v sivih literaturah, revijah, spletnih straneh. (O’Hagan, Coleman, in O’Connor 2014, 190). Večina raziskav pa se osredotoča na konec življenjskega cikla razvoja iger, kar je dobro za manjše spremembe ni pa to efektivno za spremembo ključne mehanike v igri (McAllister in White 2015, 14).

Ni procesnega modela za razvoj video iger, ki bi vključeval najboljše prakse. Takšen model bi bil zelo koristen za industrijo iger, saj bi lahko skrajšal čas razvoja in vstopa na trg in izboljšal kvaliteto video iger (O’Hagan in O’Connor 2015, 15). Potrebno je boljše razumevanje življenjskega cikla razvoja iger (McAllister in White 2015, 14). Raziskava na temo razvojnih procesov video iger bi bila koristna (O’Hagan in O’Connor 2015, 14).

• začni delati s tem kar imaš  
• prizadevaj si za izboljšanje na podlagi evolucijskih sprememb  
•spodbujaj

UPRAVLJANJE SPREMEMB

ZAGOTAVLJANJE STORITEV