Szoftverfejlesztés

Jeszenszky Péter
jeszeszky.peter@inf.unideb.hu

Utolsó módosítás: 2023. február 11.

Felhasznált irodalom

- A prezentáció az alábbi forrásokon alapul:
  - Ian Sommerville. *Software Engineering*. 10th ed. Pearson, 2015. https://software-engineering-book.com/
    - A 2006-ban megjelent 8. kiadás érhető el magyar nyelven:
      - Ian Sommerville. *Szoftverrendszerek fejlesztése*. Második, bővített, átdolgozott kiadás. Panem Könyvkiadó Kft., 2007.
  - Pierre Bourque (ed.), Richard E. (Dick) Fairley (ed.). *Guide to the Software Engineering Body of Knowledge, Version 3.0*. IEEE Computer Society, 2014. https://www.computer.org/education/bodies-of-knowledge/software-engineering
  - *Software and Systems Engineering Vocabulary (SEVOCAB)*. IEEE Computer Society, 2021. https://pascal.computer.org/

SWEBOOK

- Webhely: https://www.computer.org/education/bodies-of-knowledge/software-engineering
- A cím magyarul: „Útmutató a szoftverfejlesztés tudásanyagához”
- Az IEEE Computer Society projektje: https://www.computer.org/
- Útmutatóként szolgál a szoftverfejlesztés tudásanyagának általánosan elfogadott részéhez.
- 15 tudásterületet azonosít a szoftverfejlesztésen belül, melyek mindegyikét egy fejezet tárgyalja.
- ISO szabványként is jóváhagyásra került:
  - *ISO/IEC TR 19759:2015: Software Engineering – Guide to the software engineering body of knowledge (SWEBOK)* https://www.iso.org/standard/67604.html

Mi a szoftver?

- Programok és a hozzájuk tartozó dokumentáció.
- A szoftvertermékek fejlesztése történhet egy adott ügyfél vagy az általános piac számára.

Szoftvertermékek

- **Általános célú szoftverek**: Önálló rendszerek, melyeket a nyílt piacon értékesítenek bármely vevőnek, aki képes megvenni őket.
  - Például: szövegszerkesztők, grafikus programcsomagok, projektkezelő eszközök, könyvtári információs rendszerek, számviteli rendszerek, …
- **Egyedi szoftverek**: Egy adott ügyfél által megrendelt rendszer. A szoftverszállító kifejezetten ennek az ügyfélnek fejleszti a szoftvert.
  - Például: elektronikus eszközök vezérlő rendszerei, egy adott üzleti folyamatot támogató rendszerek, légiforgalmi irányítási rendszerek, …

A jó szoftver lényeges jellemzői

- **Karbantarthatóság**: A szoftvert úgy kell megírni, hogy az ügyfelek változó igényeinek megfelelően legyen továbbfejleszthető. Ez döntő fontosságú jellemző, mert a szoftverek változása elkerülhetetlen követelmény a változó üzleti környezetben.
- **Megbízhatóság és biztonság**: A megbízható szoftver nem okozhat fizikai vagy gazdasági károkat egy rendszerhiba esetén. A rosszindulatú felhasználóknak nem szabad hozzáférni a rendszerhez vagy károsítani azt.
- **Hatékonyság**: A szoftver nem használhatja pazarlóan a rendszer erőforrásait, mint például a memória és a processzoridő.
- **Elfogadhatóság**: A szoftver elfogadható kell, hogy legyen azon felhasználók számára, akiknek tervezték. Ez azt jelenti, hogy érthető, használható és az általuk használt más rendszerekkel kompatibilis kell, hogy legyen.

Mi a szoftverfejlesztés? (1)

- Definíció (SEVOCAB):
  - Tudományos és technológiai tudás, módszerek és tapasztalat szisztematikus alkalmazása szoftverek tervezéséhez, megvalósításához, teszteléséhez és dokumentálásához.
- A szoftverfejlesztés tehát egy mérnöki tudományág.

Mi a szoftverfejlesztés? (2)

- Definíció (Sommerville):
  - A szoftverfejlesztés (*software engineering*) mérnöki tudományág, mely a szoftverek létrehozásának valamennyi kérdésével foglalkozik a rendszer specifikáció korai szakaszaitól kezdve a rendszer használatba vétel után karbantartásáig.

A szoftverfejlesztés története

- Szoftverfejlesztés (*software engineering*) címmel került megrendezésre 1968-ban egy NATO konferencia, melyet a szoftverkrízis megvitatásának szenteltek.
  - Világossá vált, hogy a programok fejlesztésére szolgáló egyedi módszerek nem alkalmazhatóak nagy és bonyolult szoftverrendszerek esetén.

Miért fontos a szoftverfejlesztés? (1)

- Az egyén és a társadalom egyre nagyobb mértékben függ a fejlett szoftverrendszerektől. Képesnek kell lennünk megbízható rendszerek gazdaságos és gyors előállítására.

Miért fontos a szoftverfejlesztés? (2)

- Hosszú távon általában olcsóbb szoftverfejlesztési módszereket és eljárásokat használni a szoftverrendszerekhez, mint csupán úgy megírni a programokat, mint ha egy személyes programozási projektről lenne szó. A legtöbb fajta rendszernél a költségek nagy része a szoftver használatba vétel utáni változtatásának költsége.
  - A szoftverköltségek nagyjából 60%-a fejlesztési költség, 40%-a karbantartási költség. Egyéni szoftvereknél az evolúciós költségek gyakran meghaladják a fejlesztési költségeket.

Sokféleség

- A szoftvereknek nagyon sok fajtája van.
- Nem létezik olyan univerzális szoftverfejlesztési módszer vagy eljárás, mely ezek mindegyikéhez alkalmazható.

Szoftverfolyamat (1)

- Egy szoftvertermék előállításához szükséges tevékenységeket jelenti.

Szoftverfolyamat (2)

- Sok különféle szoftverfolyamat létezik, de mindegyik tartalmazza az alábbi négy tevékenységet, melyek alapvető fontosságúak a szoftverfejlesztés szempontjából:
  - **Szoftverspecifikáció**: az ügyfelek és a mérnökök meghatározzák az előállítandó szoftvert és a működésére vonatkozó megszorításokat.
  - **Szoftverfejlesztés** (***software development***): a szoftver tervezése és megvalósítása (programozás).
  - **Szoftvervalidáció**: annak ellenőrzése, hogy a szoftver az-e, amit az ügyfél is szeretne.
  - **Szoftverevolúció**: a szoftver módosítása az ügyfél és a piac változó követelményeire reagálva.

Szoftverfolyamat (3)

- A szoftverfolyamatokat néha az alábbi két kategóriába sorolják:
  - **Terv-alapú** (***plan-driven***): a folyamat tevékenységeit előre megtervezik és az előrehaladást a tervhez képest mérik.
  - **Agilis** (***agile***): a tervezés inkrementális, könnyebb a folyamaton változtatni az ügyfelek változói igényei szerint.
- Nincs tökéletes szoftverfolyamat!

Szoftverfolyamat modellek

- Egy **szoftverfolyamat modell** (szoftverfejlesztési életciklus (SDLC) modell) egy szoftverfolyamat egyszerűsített ábrázolása.
- Vannak olyan nagyon általános folyamat modellek, melyek különféle szoftverfejlesztési módszerek leírására használhatók.
  - Például: vízesés modell, inkrementális fejlesztés

Szoftver újrafelhasználás (1)

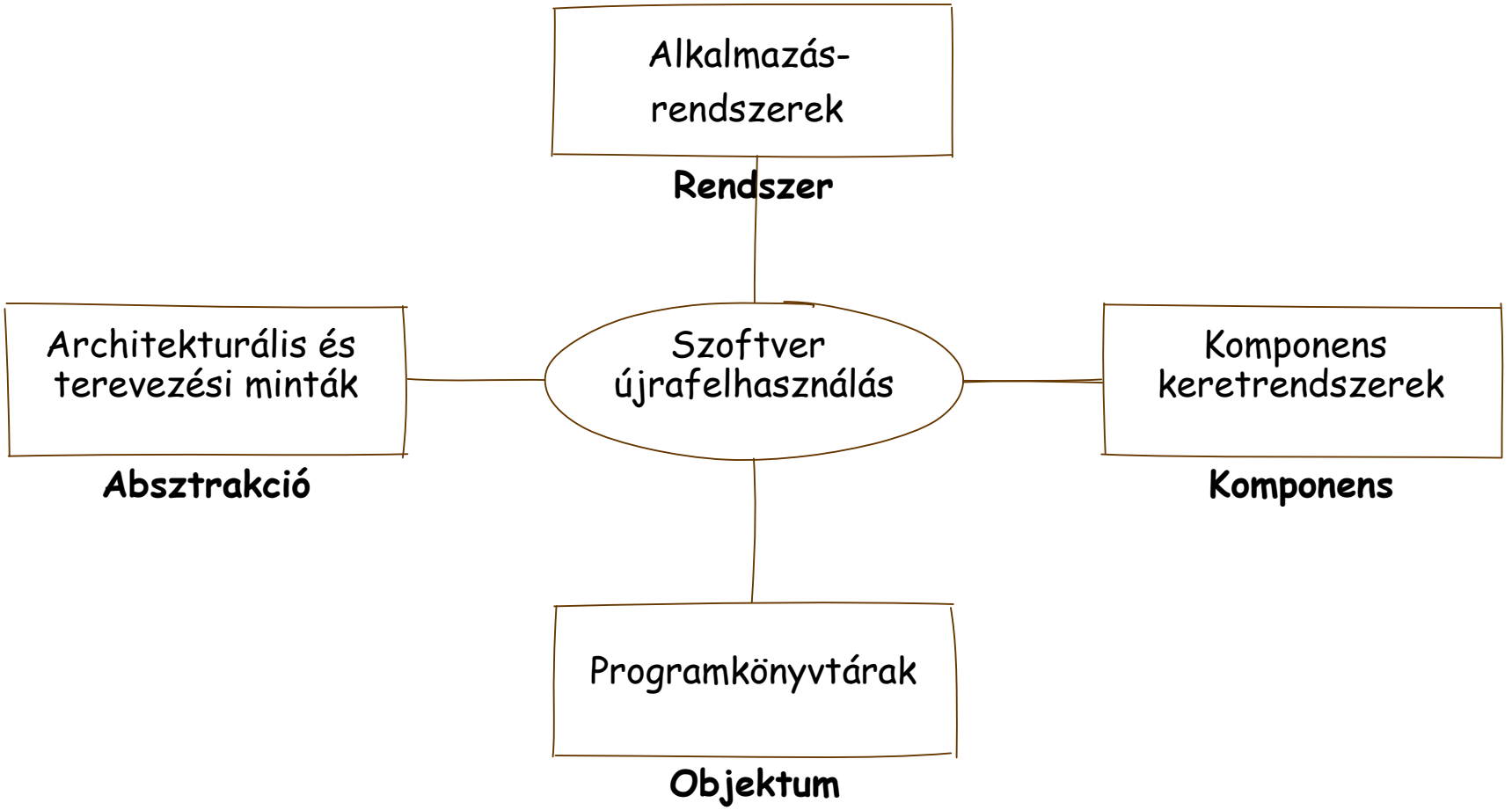
- Az 1960-as évektől az 1990-es évekig a legtöbb új szoftvert a nulláról kezdve (*from scratch*) írták.
  - Az egyetlen számottevő szoftver újrafelhasználás a programkönyvtárak használata volt.
- Azonban 2000-től kezdve széles körben elterjedtté váltak a létező szoftverek újrafelhasználására összpontosító szoftverfejlesztési folyamatok (újrafelhasználás-alapú szoftverfejlesztés).

Szoftver újrafelhasználás (2)

- Napjainkban a legtöbb modern szoftverrendszer létrehozása létező komponensek vagy rendszerek újrafelhasználásával történik.
- A nyílt forrású mozgalomnak köszönhetően hatalmas kódbázis áll rendelkezésre újrafelhasználáshoz.
- Létező szoftverek újrafelhasználásával gyorsabban, kevesebb kockázattal és kisebb költséggel fejleszthetők új rendszerek.

Szoftver újrafelhasználás (3)

- A szoftverek újrafelhasználása különböző szinteken lehetséges:



Programkönyvtárak (1)

- Definíció (SWEBOOK):
  - Egy programkönyvtár szoftverek és kapcsolódó dokumentáció olyan együttese, melyet abból a célból terveztek, hogy a szoftverfejlesztést, használatot és karbantartást segítse.
- Egy könyvtár erőforrásokat tartalmaz, mint adatok és kód, és egy jól meghatározott interfésze (API-ja) van.

Programkönyvtárak (2)

- Példák:
  - **C/C++**: Boost libraries, GNU C Library (glibc), OpenCV, OpenSSL
  - **Java**: Apache Log4j 2, Apache Commons, Guava, Jackson
  - **JavaScript**: Chart.js, jQuery, React, Three.js
  - **Python**: Matplotlib, NumPy, Requests, seaborn

Alkalmazásprogramozási interfész (API) (1)

- Definíció (SWEBOOK):
  - Egy alkalmazásprogramozási interfész (API) egy könyvtár vagy egy keretrendszer által a felhasználók számára alkalmazások írásához exportált és elérhető szignatúrákat jelenti.
    - A szignatúrák mellett egy API mindig közléssel kell, hogy szolgáljon a programok hatásáról és/vagy viselkedéséről.
- Definíció (SEVOCAB):
  - Szoftverkomponens, mely lehetővé teszi, hogy szoftveralkalmazások egymással kommunikáljanak.

Alkalmazásprogramozási interfész (API) (2)

- Egy API specifikáció egy API-t definiáló és leíró dokumentum.
- Azt mondjuk egy API specifikációnak megfelelő szoftverre, hogy implementálja az API-t.
- A szoftvereknek lehet egyedi API-ja, azonban különböző szoftverek implementálhatják ugyanazt azt API-t, ami lehetővé teszi az interoperabilitást.
- Az API-k akár szabványosíthatók is.
  - Példák szabványos API-kra:
    - DOM, JDBC, Jakarta Mail, Jakarta Persistence, Python Database API, WebGL, WebSocket, …

Keretrendszerek (1)

- Definíció (SWEBOOK):
  - Egy keretrendszer egy részlegesen befejezett szoftverrendszer, mely bizonyos kiterjesztések (például *plugin*-ek) megfelelően történő példányosításával terjeszthető ki.
- Definíció (Schmidt et al.):
  - Egy keretrendszer szoftvertermékek (például osztályok, objektumok és komponensek) olyan integrált együttese, melyek abból a célból működnek együtt, hogy egy újrafelhasználható architektúrát biztosítsanak hasonló alkalmazások egy családjához.

Keretrendszerek (2)

- Egy keretrendszer egy félkész alkalmazás.
  - A fejlesztők a keretrendszer újrafelhasználható komponenseinek kiterjesztésével és testreszabásával alkotnak teljes alkalmazásokat.
- A keretrendszerek egy csontváz architektúrát biztosítanak az alkalmazásokhoz.
- A keretrendszerek jellemzően mintákon alapulnak, mint az architekturális minták és tervezési minták.

Keretrendszerek (3)

- Egy rendszer egy keretrendszerrel történő megvalósításához konkrét osztályokat hozunk létre, melyek a keretrendszer absztrakt osztályaitól örökölnek műveleteket.
- Emellett *callback* metódusokat definiálunk, melyek válaszként kerülnek meghívásra a keretrendszer által felismert eseményekre.
- Inkább a keretrendszer objektumok felelnek a rendszerben a vezérlésért, nem pedig az alkalmazás-specifikus objektumok.

Keretrendszerek (4)

- Hátrányuk, hogy természetüktől fogva bonyolultak és hónapokba telhet a megtanulásuk.
- Egy keretrendszer magában foglalhat más keretrendszereket, ahol minden egyes keretrendszert abból a célból terveztek, hogy az alkalmazás egy részének fejlesztését támogassa.
  - Egy keretrendszer használható egy teljes alkalmazás létrehozásához vagy annak csupán egy részének, például a grafikus felhasználói felületnek a megvalósításához.

Keretrendszerek (5)

- Példák:
  - **C/C++**: Boost.Test, GoogleTest, Qt
  - **Java**: Hibernate ORM, JUnit 5, Mockito, Play Framework, Spring Framework
  - **JavaScript**: Angular, Express, Jest, Mocha, Next.js, Vue.js
  - **Python**: Django, Flask, unittest