Esimerkki riskilähtöisestä arkkitehtuurityöstä ja mallinnuksesta

Luento 11 (osa 2)

SUD

- *"The Home Media Player is a computer that plays media (like music, videos, and pictures) on a television and stereo. It is a normal computer like a laptop with a single audio and video output that is hooked up to a television and optionally to a stereo.*
- *This Home Media Player is able to play media in multiple formats from its local hard disk or streamed from the internet. It can simultaneously play music and display a slideshow of pictures, or show a video and browse information about that video.*
- *Third parties can build extensions to enable the system to play streaming media or collect metadata (e.g. song lyrics or actor biographies) from internet sites."*

Fairbanks, 4. Luku

SUD

- Esimerkin taustalla samantapainen todellinen järjestelmä
- Esimerkissä yhdistyvät tyypilliset "systeemi" ja "IT" –asiat
  - Suorituskyky ja luotettavuus
  - Monimutkaisen datan käsittely
- Lähtötilanne: tiimi on tehnyt järjestelmästä prototyypin, ja nyt on tarkoitus rakentaa oikea tuote

Haasteet

1. *Suunnittelun kommunikointi*
   - Projektiin on tulossa mukaan uusia kehittäjiä toisessa toimipisteessä. Pelätään, että uudet kehittäjät eivät ehkä ymmärrä suunniteltua arkkitehtuuria eivätkä pysty osallistumaan täysipainoisesti tai voivat vahingossa rikkoa arkkitehtuurin.
2. *COTS-komponenttien integrointi*
   - Prototyyppi rakennettiin vain yhtä käyttöjärjestelmää ajatellen, mutta nyt halutaan käyttää myös (kaupallisia) valmiskomponentteja, jotta järjestelmä saataisiin toimimaan useammassa ympäristössä
3. *Metadatan esityksen yhtäpitävyys*
   - Metadatan sisäisen esityksen epäyhteensopivuus internetissä käytettyjen metadatamuotojen kanssa voi estää kolmansien osapuolten laajennosten toteuttamisen.

Haasteet

- Huom - vain 1. haaste eli *suunnittelun kommunikointi* käydään läpi luennolla, lue loput itse kurssikirjasta

Esimerkkitapauksen rajaukset ja oletukset

- Oletetaan vaatimukset tunnetuiksi
- Ei kiinnitetä huomiota kehitystiimin eri rooleihin
- Ei ota kantaa kehitysprosessin luonteeseen
- Kaikki osapuolet ovat yhtä mieltä laatuattribuuttien priorisoinnista
- Tapauksen kuvauksen fokus on suunnittelussa

Haaste 1

SUUNNITTELUN KOMMUNIKOINTI

Prototyypistä tuotteeksi

- Mediasoittimen prototyyppi kehitettiin start-up –tyyliin
  - Tiivis ryhmä teki pitkiä päiviä yhteisessä työtilassa
  - Kaikki osallistuivat suunnittelupäätöksiin ja arkkitehtuuri on jokaisella ”nahkakansissa” (omassa päässään)
- Prototyyppi on päätetty tuotteistaa, ja yritys on palkkaamassa lisää työntekijöitä projektiin
  - Uudet kehittäjät eivät tule olemaan samoissa tiloissa eivätkä tunne sovellusaluetta entuudestaan

Prototyypistä tuotteeksi

- Miten estetään arkkitehtuurin eroosio (*architectural drift*), jota helposti tapahtuu, jos ja kun tuotekehitysprojektin paineessa uudet kehittäjät (tarkoittamattaan) kirjoittavatkin suunnittelupäätösten kanssa ristiriitaista koodia?
- Täytyy siis jotenkin kommunikoida arkkitehtuuri ja tehdyt suunnittelupäätökset koko laajennetulle tiimille
  - Staattinen, dynaaminen, operatiivinen olomuoto

Riskilähtöinen kommunikointistrategia

- Aloitetaan yksinkertaisista ja suhteellisen halvoista keinoista
- Otetaan monimutkaisempia keinoja käyttöön, vain jos yksinkertaisilla ei näytä saatavan riskiä tarpeeksi vähennettyä
  - Subjektiivinen arvio
- Lopetetaan, kun ei saada enää riittävää lisäarvoa suhteessa mallien/dokumentoinnin vaatimaan työmäärään

Entä jos vain luetaan koodia?

- Prototyyppi ei ole kovin iso, joten voisi tietysti vain pyytää uusia kehittäjiä lukemaan sen koodia
  - Ei vaadi mitään ylimääräistä vaivaa tiimin vanhoilta jäseniltä
- Esimerkiksi koodin nykyinen hakemistorakenne vastaa jossain määrin moduulijakoa

- Application
- External_libs
  - Audio_codecs
  - Audio_player
  - Video_codecs
  - Video_player
  - Remote_controls
- Gui
- Media_player
- Util

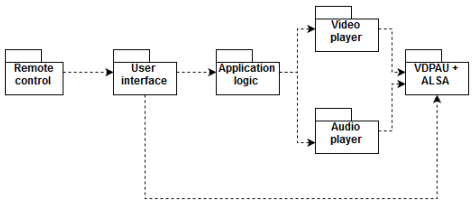
Ongelmia

- Hakemistorakenteesta ei näe moduulien välisiä riippuvuuksia
- Prototyypin koodi ei kaikilta osin ole kovin hyvää, eivätkä arkkitehtuuriratkaisut oikein näy siitä (*model-code gap*)
- Koodin lukemiseen menee myös paljon aikaa, jonka uudet kehittäjät voisivat käyttää paremminkin
  - Tärkeimpien asioiden dokumentointi vie kuitenkin todennäköisesti vähemmän henkilötyötunteja
- Erityisesti ajonaikaisten rakenteiden päättely koodista on hyvin hankalaa

Moduulimalli

- Päätetään laatia ainakin moduulimalli
  - Aloitetaan siis yksinkertaisesta, lähellä koodia olevasta näkymästä (staattinen olomuoto, moduulinäkymätyyppi)
  - Moduulit, riippuvuudet, mahdolliset rajoitteet
  - Kattaa koko ohjelmiston (komponentit), ei vain systeemitason rajapinnat
- Kaavion lisäksi (seuraavalla dialla) lisätään linkit tärkeimpien käytettyjen teknologioiden kotisivuille

Moduulimalli



Jäljellä olevan riskin arviointi

- Moduulimalli auttaa nyt hahmottamaan perustavanlaatuisen järjestelmän osituksen staattisessa olomuodossa
  - Hyvin valitut nimet ja riippuvuudet voivat auttaa arvaamaan oikein perusskenaarioiden kulun
- Se ei kuitenkaan kerro kovin paljon
  - Suoritusaikaisista rakenteista ja järjestelmän käyttäytymisestä (dyamiikka)
  - Laatuvaatimuksista ja niiden priorisoinnista sekä näihin liittyvistä suunnittelupäätöksistä

Laatuattribuutit

- Tiimi on kokemuksensa perusteella päätynyt laatuattribuuttien seuraavaan priorisointiin
  - KL:n responsiivisuus > Audion/videon toiston sulavuus (ajoitus) > luotettavuus > muunneltavuus > toiston absoluuttinen suorituskyky (framerate) > siirrettävyys
- *Responsiivisuus* ja *toiston sulavuus* ovat erittäin tärkeitä kilpailutekijöitä
- *Siirrettävyys* on vähemmän tärkeää, koska ohjelmisto ja laitteisto paketoidaan yhdeksi integroiduksi tuotteeksi

Tasapainottelua

- *Siirrettävyys* ja *Toiston sulavuus*
  - Aidosti siirrettävä ratkaisu vaatisi yleensä eri käyttöjärjestelmien ja laitteistojen erot piilottavan ylimääräisen API-kerroksen (fasaadi, adapteri patterni) käyttöä arkkitehtuurissa
  - Tämä ylimääräinen kerros lisäisi suoritusaikaista viivettä ja voisi ajoittain vaikuttaa huonontavasti äänen toiston laatuun
  - Siksi päätettiin käyttää suoraan alustakohtaisia (natiivi-) API:ja, mikä heikentää vastaavasti siirrettävyyttä

Taspainottelua

- *Toiston suorituskyky* ja *Muunneltavuus*
  - Median toiston suorituskykyä voi yleensä parantaa koodekki- tai mediadatakohtaisilla optimoinneilla
  - Toisaalta suorituskyky tuntui olevan riittävä valitulla laitteistolla ilmankin optimointeja, jotka olisivat monimutkaistaneet uusien koodekkien ja dataformaattien lisäämistä, mikä haluttiin pitää mahdollisimman yksinkertaisena muunneltavuuden vuoksi

Arkkitehtuuriajurit

- Alkuperäinen tiimi ei ollut dokumentoinut yhtään laatuattribuuttiskenaariota, mutta tiimi käytti toistuvasti kahta skenaariota suunnittelun ja testauksen mittatikkuina
- Nämä kaksi skenaarioita olivat siis prototyypin *arkkitehtuuriajureita* (architectural driver), ja ne päätettiin nyt kirjoittaa auki

Arkkitehtuuriajurit

1. Järjestelmän pitää tuottaa vaste käyttäjän KL:n kautta antamiin komentoihin 50ms:n kuluessa komennon antamisesta. Tilanteissa joissa aikarajaa ei voida millään saavuttaa (kuten aloitettaessa videotiedoston toistoa), järjestelmän pitää kuitenkin antaa palaute, joka osoittaa arvion jäljellä olevasta odotusajasta
2. Referenssinä käytetyn H.264/MPEG-4 AVC videon pitäisi toistua sujuvasti paikalliselta kovalevyltä luettuna

Tärkeimmät suunnittelupäätökset

- Laatuattribuuttien priorisoinnin ja arkkitehtuuriajureiden dokumentoinnin lisäksi katsottiin aiheelliseksi vielä kirjoittaa esiin tärkeimmät laatuvaatimuksien toteutumiseen vaikuttavat suunnittelupäätökset
  - joita on vaikea nähdä pelkästään koodista

Suunnittelupäätökset

#	Kuvaus
1	Luotettavuuden lisäämiseksi jokaista päätason komponenttia suoritetaan omassa prosessissaan
2	Media Rendering (toisto) kommuninkoi Media Buffer:n (puskurin) kanssa jaettua muistia käyttäen viiveen minimoimiseksi
3	Sulavan toiston takaamiseksi mediadata luetaan levyltä tai streamista ensin RAM-muistissa olevaan puskuriin, ennen kuin sitä aletaan toistaa
4	Kaikki toistettavan median metadata talletetaan järjestelmän Metadata Repository:yn, vaikka sama data sisältyisikin itse mediaan
5	Vain Media Player Core –komponentti saa päivittää metadataa Metadata Repository:ssä

Riskin uudelleenarviointi

- Tähän mennessä on siis laadittu *moduulimalli* sekä kirjattu keskeiset *laatunäkökohdat* ja niitä suoraan koskevat *suunnittelupäätökset*, mikä pienentää riskiä jo selvästi
- Ohjelmiston dynaamisen olomuodon kuvaus kuitenkin käytännössä puuttuu, mikä kasvattaa virhetulkintojen mahdollisuutta
- Jäljellä olevan riskin pienentämiseksi katsotaan tarpeelliseksi mallintaa vielä järjestelmän toimintaa

Ajonaikaiset mallit

- Eniten lisäarvoa vähimmällä vaivalla tuottava keino suoritusaikaisten rakenteiden ymmärtämiseksi on luetella arkkitehtuurin komponentit ja konnektorit sekä selostaa niiden vastuut

Komponentit ja konnektorit

Elementti	Vastuut
Media Rendering	Komponentti joka toistaa mediatiedostoja audio- ja videoulostulojen kautta. Vastaa myös käyttöliittymäelementtien kuten valikkojen näyttämisestä.
Media Player Core	Mediasoittimen keskeiset sovellustoiminnot, käyttöliittymätoiminnot, sekä koko toiminnallisuuden koordinointi
Media Buffer	Puskuroi mediatiedostojen sisältöä muistiin toiston ajoitushäiriöiden välttämiseksi. Tarjoaa pääsyn dataan jaetun muistin kautta.
Messaging Connector	Asynkroninen kaksisuuntainen viestinvälityspalvelu
...	...

Muut ajoaikaiset näkymät

- Vastuiden kuvauksen perusteella saa jo melko hyvän kuvan järjestelmän dynamiikasta
- Kuvaa voidaan vielä täydentää
  - Komponenttikokoonpanolla, joka näyttää pääkomponentit, niiden portit sekä konnektorit
  - Tarkennetulla toiminnallisella skenaariolla, joka kuvaa komponenttien toiminnot tärkeimmissä käyttötapauksissa
- (Kaavio ja skenaario löytyvät kurssikirjasta)

Reflektio

- Edellä kuvatut tekstuaaliset ja graafiset mallit on laadittu vaiheittain aloittaen yksinkertaisista mutta paljon lisäarvoa tuovista esityksistä
  - Kattavat sekä staattisen että dynaamisen olomuodon (operatiivinen ei kovin tärkeä tälle järjestelmälle)
- Jokaisen mallin laatimisen jälkeen arvioitiin uudestaan jäljellä olevaa riskiä ja harkittiin mutkikkaampien kuvausten tarvetta
- Mallintaminen lopetettiin, kun arkkitehtuurin kommunikointiin liittyvän riskin arvioitiin käyneen riittävän pieneksi – mallien/dokumenttien laatimiseen kuluu aina jonkun työaikaa, ja dokumentaatiota täytyy myös pitää ajan tasalla
- Katso kahden muun haasteen analyysi ja laaditut mallit/kuvaukset kurssikirjasta