581358 Ohjelmistoarkkitehtuurit

Harjoitus 2

1. Pohdi ja selosta, miten yleiset suunnitteluperiaatteet Separation of Concerns ja Information Hiding ilmenenvät Kolmistasoarkkitehtuurissa (N-Tier) ja MVC-arkkitehtuurissa (Web-MVC tai alkuperäinen MVC).
2. Tutustu Microsoftin MVVM (Model-View-ViewModel) malliin ja vertaa sitä alkuperäiseen MVC:hen. Miten MVVM eroaa perus-MVC:stä? Mitä tarkoittaa ”data binding”?
    - http://en.wikipedia.org/wiki/Model_View_ViewModel
3. Edellisten harjoitusten 4. tehtävässä pohdittiin harjoitustehtävien kirjausjärjestelmän tuote- ja projektiriskejä. Riskianalyysin tuloksena on nyt saatu lista merkittävimmistä riskeistä (alla). Pohdi, minkälaisilla arkkitehtuuritason suunnitteluratkaisuilla voisi lieventää näitä riskejä. Ota huomioon, että ratkaisusi ovat suhteessa järjestelmän luonteeseen (kyseessä ei ole kriittinen järjestelmä, lähinnä apuväline). Miten arkkitehtuuria voisi muuten käyttää riskien hallinnan apuna?
    - Tuoteriskit
        i. Monen alustan tukeminen voi osoittautua vaikeaksi
        ii. Muutokset toimintaympäristössä/muitten järjestelmien rajapinnoissa
        iii. Miten taataan saatavuus 24/7? Oltava jokin luotettava varajärjestelmä.
        iv. Tietojen siirto Kurkeen on hankalaa. Suoraa yhteyttä ei saada järjestelmien välille, vaan tiedon siirto tapahtuu tiedostojen kautta.
    - Projektiriskit
        i. Ohtu-ryhmien osaaminen vaihtelee paljon. Tehtävä voi osoittautua liian haastavaksi joiltain osiltaan.
        ii. Tiedon kulku eri toteutusryhmien välillä (toteutus peräkkäisissä ohtu-projekteissa).
        iii. Rinnakkain etenevien tiimien kommunikointi ja synkronointi, jos järjestelmä toteutetaan kahden ohtu-ryhmän yhteistyönä.
4. SEMAT (Software Engineering Method and Theory) on hanke, jossa pyritään uudelleen löytämään ohjelmistokehityksen ydinasiat eli *Essence* (http://en.wikipedia.org/wiki/SEMAT, semat.org). Ytimen (Kernel) ympärille on määritelty joukko *peruskäytäntöjä* (Practice):

    ” A practice defines a simple, light-weight solution to a specific software development problem. The Essential Practices focus on the essential things which need to be done in order to ensure a development effort stays on track. Each practice consists of Things to Produce, Things to Do, Competencies required and Patterns which can be applied.”[1]

    Tutustu *Architecture Essentials* –käytäntöön osoitteessa http://www.ivarjacobson.com/uploadedFiles/Content/Resources/Practices/Architechural_Essentials.pdf

    Mitkä piirteet käytännössä tukevat mielestäsi ketterää kehitystä? Entä miten yhteensopiva tämä käytäntö on Fairbanksin *Risk-driven approach* –idean kanssa?

    [1] http://www.ivarjacobson.com/Practices/