Rapport codekwaliteit EASY SWORD

*Datum: 25 september 2012*

*Auteur: Jan van Mansum*

# Inleiding

Dit document is het resultaat van een code-review die ik in het kader van het project Auto-Ingest Afronding (AIA) heb gedaan op de code van het EASY SWORD-project. Om een goede review te doen is het eigenlijk nodig om duidelijke criteria te hebben waaraan je het te reviewen product toetst. Er is bij DANS echter (nog) geen Coding Guide o.i.d. Daarom ben ik voor deze review uitgegaan van de *best practises* zoals ik ze uit de programmeerliteratuur ken. Wellicht kan deze review als aanzet tot een DANS Coding Guide dienen.

Een goed georganiseerd software-project moet op een aantal gebieden zijn zaken op orde hebben. Er zijn verschillende onderverdelingen te maken. Ik heb een groffe onderverdeling gemaakt o.b.v. belanghebbende.

* (Klant) Requirements – voldoet de software aan de functionele en overige requirements (voor zover deze duidelijk zijn)?
* (Programmeur) Development/maintenance:
  + Clean code – is de code “clean?” d.w.z. geschreven op een manier die de intentie duidelijk communiceert naar de programmeur en daardoor onderhoudbaar en uitbreidbaar is?
  + Tests – zijn er voldoende unit-tests?
  + Build – is de build triviaal?
* (Beheerder) Beheer – is er voldoende documentatie voor de applicatiebeheerder en technisch beheerder?

# (Functionele) requirements

(Opmerking vooraf: wellicht hoort dit hoofdstuk niet thuis in een code-review-document. De (afwijkingen) van de requirements zouden in de issue-database moeten staan. In dit document is eigenlijk alleen een samenvatting van de stand van zaken op zijn plaats, met verwijzingen naar de openstaande issues.)

*Zijn er functionele requirements?*

In deze sectie staat de vraag centraal: voldoet de software aan de requirements? Dit valt natuurlijk alleen goed te bepalen als de requirements ondubbelzinnig zijn vastgelegd. Bij het EASY SWORD zijn niet alle requirements als zodanig vastgelegd. De volgende bronnen van specificaties zijn beschikbaar:

* TestPlanSword.doc – een checklist van te testen zaken
* UserGuideSword.doc -
* SWORD v1.3 profile specs – dit is het protocol waarop de Auto-Ingest gebaseerd is. Het is een profile (aanpassing/verfijning) van het Atom Publishing Protocol (APP). Het specificeert hoe je geaggregeerde resources kunt aanmaken en bewerken via APP.
* Atom Publishing Protocol specs: RFC 5023 – de basis waarop SWORD gebouwd is.
* HTTP specs: RFC 2616 – APP is een RESTful protocol en leunt dus zwaar op het correcte gebruik van HTTP.

Een groot deel van de requirements zijn in bovenstaande documenten terug te vinden danwel eruit af te leiden.

*Welke requirements zijn vervuld?*

*Welke requirements zijn niet vervuld?*

# Development

# Beheer

# Voldoet de code aan de functionele requirements?

* Response-document bij ingest tegen de APP en SWORD specs houden

# Is de code leesbaar en onderhoudbaar?

* + De gegeneerde klassediagrammen kan ik niet openen. Ik heb de gratis versie van ObjectAid.
* Adequaat gebruik van library’s en frameworks? (te veel zelfgemaakt? libs en frameworks op wijze gebruikt die niet de bedoeling is?)
* Voldoende logging en met de juiste niveau’s?
* Naamgeving consistent, verduidelijkend?

# Is test-code netjes gescheiden van de productie-code?

# Bevat de code een adequaat commentaar?

* Javadocs
* TODO’s:

# Is het project voldoende self-contained?

* Is alle informatie aanwezig om het te bouwen?
  + welke tools? (Dit kan grotendeel gestandaardiseerd worden bij DANS: Eclipse, (welke plugins?), Maven 3.0.3, java 6, etc., Artifactory repository)

Het project bevat geen documentatie voor de ontwikkelaar. Beschreven zou moeten zijn wat een EASY-ontwikkelaar moet weten om het project schoon te bouwen. Ik zeg EASY-ontwikkelaar omdat het hebben van een EASY-ontwikkel-omgeving voorondersteld mag worden. (Het opzetten van zo’n omgeving is gedeeltelijk gedocumenteerd, maar deze documentatie zou eigenlijk ook beter georganiseerd moeten worden en op een vaste, bekende plek moeten staan, zodat de EASY SWORD documentatie ernaar kan verwijzen.)

Wat ontbreekt er?

* EASY-dependency’s moeten eerst gebouwd worden (ze moeten in de lokale Maven-cache staan).

# Is de build schoon?

* mvn clean install
  + Geen warnings in build-log
    - Er zijn nog verscheidene warnings aanwezig:
      * [WARNING] The POM for org.hibernate:jtidy-r8:jar:20060801 is missing, no dependency information available
      * In unit-tests: Not a valid storeId:
      * WARN 10:59:01,247 no submission mail sent for mock
  + Geen onnodige output in build-log
    - Er staat nog debug-level info in build-log.
* mvn site:site (voor het genereren van de rapportage)

# Is de code schoon?

* Geen warnings in bron-code
  + 7 warnings voor unused imports
  + Een hoop warnings over .cls-files (van ObjectAid). Gerelateerd aan het probleem dat ik ze niet kan openen?
* Geen sysouts in code
  + Er staan nog sysouts in TestFederativeAuthentication
  + Er staan nog e.printStackTrace() calls in: FederativeAuthentication, Start
* Geen uitgecommente code

# Zijn er voldoende unit-tests? (Code/branch-coverage, zijn belangrijke delen genoeg getest?)

# Nog onder te brengen punten

* Waarom wordt in EasyBusinessFacade (main code) een test-library gebruikt (EasyMock)? (nl in mockSubmittedDataset)
* Context als statische container van settings is volgens mij een anti-pattern. Spring laat je heel gemakkelijk de settings injecteren waar je ze nodige hebt. Doe dat dan ook!

# 