Software Improvement Group

Memo / confidentieel

Onderwerp	Inspectie onderhoudbaarheid Portero
Datum	25 augustus 2016
Auteur	Niels van der Zwan, Pepijn van de Kamp

1 Inleiding

1.1 Context

Portero, het kernsysteem van CIZ, is in afgelopen jaren ontwikkeld en beheerd door een externe organisatie. Het contract met deze huidige partner is eindigend en daarom is CIZ gestart met de voorbereiding van een nieuwe aanbesteding voor het beheer van Portero. Om een zo goed en betrouwbaar mogelijke aanbieding te ontvangen van aanbieders en vragen omtrent kwaliteit te minimaliseren is het voor CIZ wenselijk dat de onderhoudbaarheid van Portero door een onafhankelijke organisatie wordt vastgesteld. In dit kader is aan SIG gevraagd om een kwaliteitsanalyse uit te voeren naar de onderhoudbaarheid van Portero, conform het SIG/TÜViT kwaliteitsmodel. De uitkomst en de rapportage zal onderdeel uitmaken van de tenderdocumentatie, waardoor de markt een betrouwbaar inzicht verkrijgt in de onderhoudbaarheid van Portero en er een baseline voor kwaliteit ontstaat.

1.2 Onderzoeksvraag

In kader van de bovenstaande situatie is de onderzoeksvraag:

'Wat is de huidige kwaliteit van Portero op basis van het SIG/TÜViT kwaliteitsmodel voor onderhoudbaarheid?'

1.3 Aanpak

SIG heeft een initiële analyse van de source code van Portero gedaan. Deze analyse is gevalideerd met het ontwikkelteam van Portero tijdens een technische validatiesessie. Op basis van de bevindingen uit de technische validatiesessie heeft SIG de definitieve meting uitgevoerd. De resultaten van de definitieve meting zijn opgenomen in dit memo.

2 Scoping

Onderdeel van het onderzoek door SIG is het vaststellen van de scope waarop de analyse voor onderhoudbaarheid wordt uitgevoerd. Op basis van de ontvangen codebase (upload dd. 8-7-2016 door Erik Danckaarts) is de volgende scope vastgesteld. Deze scope is met het ontwikkelteam gevalideerd tijdens de technische sessie. Een overzicht van de gebruikte technologieën en omvang is opgenomen in TabelTabel 1: Gebruikte technologieën Portero en hun omvang (exclusief testcode)

.

Technologies	Lines of code	Man years
Groovy	82.923	14
Cordys BPM	1.569 (Activities)	5
GSP	23.929	5
Java	26.955	3
JavaScript	16.579	1
Total	**150.386 LOCs + 1.569 Activities**	**28 MY**

Tabel 1: Gebruikte technologieën Portero en hun omvang (exclusief testcode)

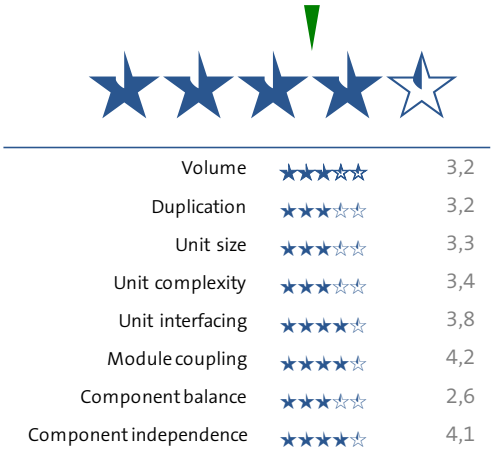
De theoretische herbouwwaarde van Portero op basis van marktgemiddelde productiviteit voor de gebruikte technologieën is 28 manjaar. Activiteiten die onderdeel zijn van deze herbouwwaarde zijn technisch design, *coding* en unit testen. Additionele inspanning is nodig voor functioneel design, projectmanagement en systeem-, integratie- en acceptatietesten. Met uitzondering van Cordys BPM zijn voor de gebruikte technologieën het aantal 'Lines Of Code' geteld. Voor de Cordys BPM technologie is het aantal procesactiviteiten geteld.

De database code en testcode vallen buiten de metingen en zijn niet meegenomen in de analyse van de onderhoudbaarheid.

3 Resultaten

3.1 Eindscore

Portero scoort ★★★★☆ (3,5) op onderhoudbaarheid op basis van het SIG/TÜViT kwaliteitsmodel voor onderhoudbaarheid. Dit betekent dat de applicatie boven markgemiddeld onderhoudbaar is. In onderstaande tabel zijn de scores voor de onderliggende metrieken weergegeven.



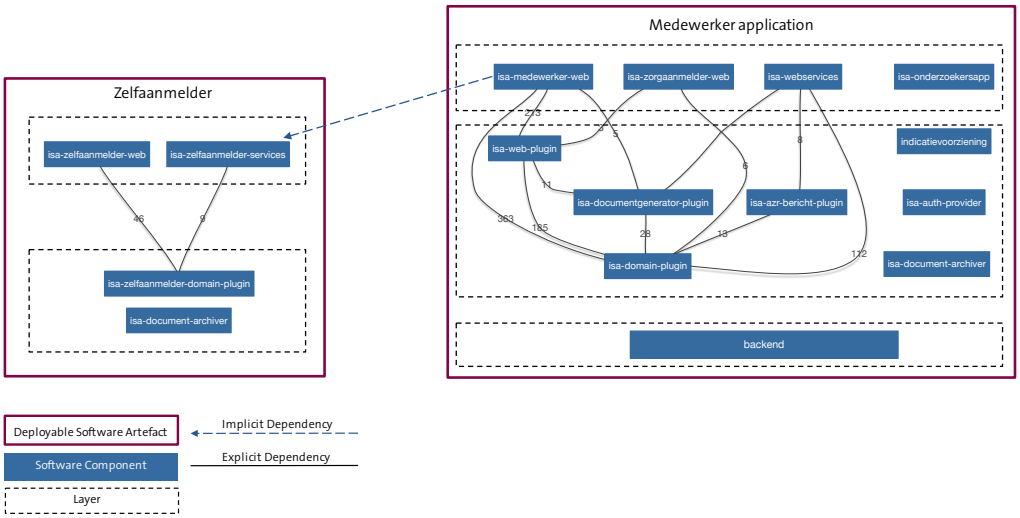
Figuur 1: Onderhoudbaarheid van Portero en de onderliggende metrieken

3.2 Onderliggende metrieken

In deze paragraaf worden de scores van de individuele metrieken besproken.
In hoofdlijnen is de Portero broncode op te splitsen in twee subsystemen die onafhankelijk van elkaar worden gedeployed:

- Medewerker-subsysteem
- Zelfaanmelder-subsysteem

Er zijn geen directe afhankelijkheden tussen de broncode van beide subsystemen. Vanuit het Zelfaanmelder-subsysteem wordt een REST-interface gefaciliteerd welke wordt gebruikt door het Medewerker-subsysteem voor het ophalen van gegevens. De Cordys BPM backend wordt alleen gebruikt door het Medewerker-subsysteem. Figuur 2 geeft een overzicht van de applicaties, layers, componenten en onderlinge afhankelijkheden binnen het Portero systeem.



Figuur 2: De twee Portero sub-systemen en de afhankelijkheden tussen de Portero componenten

3.2.1 Volume (3.2)

In vergelijking met de systemen uit de benchmark van SIG is Portero een systeem van gemiddelde omvang. Figuur 3 geeft een overzicht van het volume en de onderhoudbaarheid van het Portero systeem ten opzichte van de benchmark van SIG.

Figuur 3: Vergelijking van het Portero systeem ten opzichte van de SIG benchmark

3.2.2 Duplicatie (3.2)

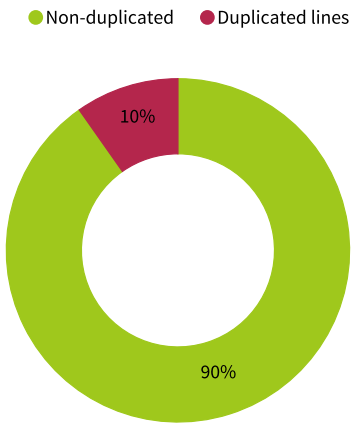
Ongeveer 10% van de applicatie bestaat uit gedupliceerde regels code. Tabel 2 geeft een overzicht van de hoeveelheid duplicatie per component. Een code-duplicaat is een stuk code van meer dan zes regels dat meermaals voorkomt in de broncode; het betreft dubbele regels die exacte kopieën zijn van elkaar. Eén van deze duplicaten kan worden beschouwd als 'origineel' en de rest is redundante code. Duplicatie wordt berekend als het percentage van de

gehele code dat een code-duplicaat is. De duplicatie van het Portero systeem scoort marktgemiddeld.

De componenten *isa-zelfaanmelder-domain-plugin*, *isa-web-plugin*, *isa-documentgenerator-plugin*, *isa-zorgaanmelder-web*, *isa-zelfaanmelder-web*, *isa-zelfaanmelder-services*, *indicatievoorziening* en *servicemodel* bevatten relatief meer duplicatie ten opzichte van het systeem als geheel. Een aantal van deze componenten is recentelijk gebouwd (*zelfaanmelder* en *document-archiver*).

Component	Duplicatie
back-end	8%
indicatievoorziening	**32%**
isa-auth-provider	0%
isa-azr-bericht-plugin	4%
isa-document-archiver	9%
isa-documentgenerator-plugin	**15%**
isa-domain-plugin	4%
isa-medewerker-web	9%
isa-onderzoekersapp	8%
isa-web-plugin	**14%**
isa-webservices	1%
isa-zelfaanmelder-domain-plugin	**11%**
isa-zelfaanmelder-services	**22%**
isa-zelfaanmelder-web	**21%**
isa-zorgaanmelder-web	**14%**
servicemodel	**84%**

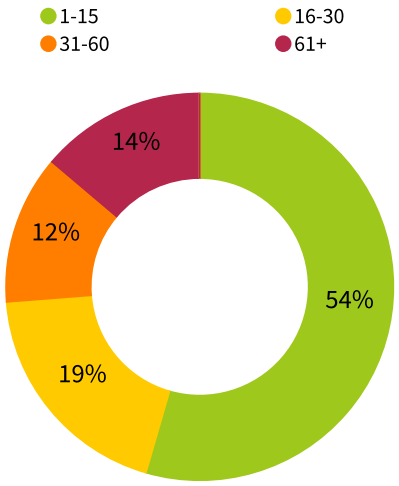
Tabel 2: Duplicatie per component



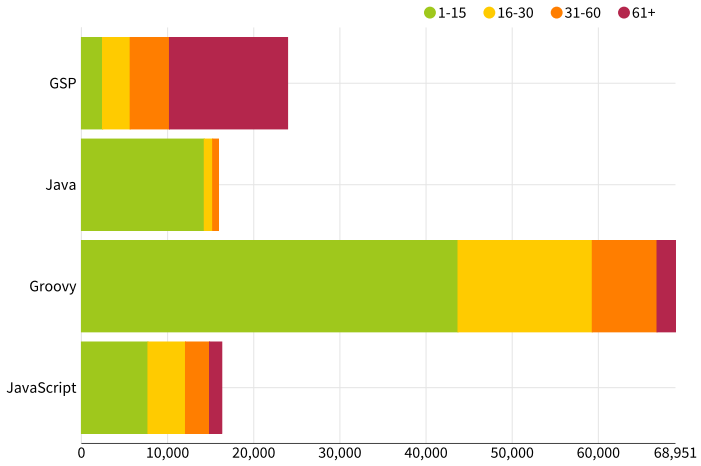
Figuur 4: Percentage gedupliceerde code in Portero

3.2.3 Unit Size (3.3)

Uit de metingen blijkt dat 26% van de Portero code zich bevindt in grote tot zeer grote units (> 30 lines of code). De score voor unit size van het Portero systeem is marktgemiddeld.



Figuur 5: 26% van de Portero code is te vinden in grote tot zeer grote units

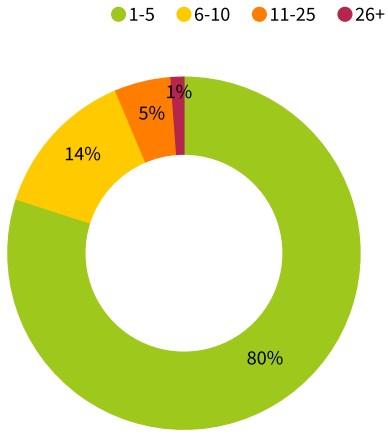


Figuur 6: Verdeling van de unit size per technologie

Uit Figuur 6 blijkt dat het grootste deel van de grote en zeer grote units te vinden is in de GSP code. Dit is vooral het gevolg van de grote hoeveelheid Embedded Javascript code in de GSP code. Van de gebruikte Cordys BPM technologie is 2% van de procesactiviteiten onderdeel van grote tot zeer grote processen (> 30 procesactiviteiten).

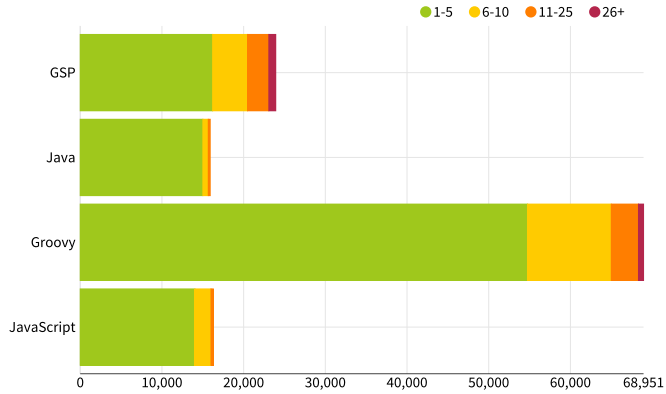
3.2.4 Unit Complexity (3.4)

Uit de metingen blijkt dat 6% van de Portero code zich bevindt in units met een hoge tot zeer hoge complexiteit (> 10 unieke codepaden). De unit complexity van het Portero systeem scoort marktgemiddeld.



Figuur 7: 6% van de Portero broncode is te vinden in complexe en zeer complexe units

Figuur 8 geeft een overzicht van de verdeling van complexe code over de verschillende technologieën. Code met een hoge tot zeer hoge complexiteit is met name te vinden in de GSP-technologie. De hoge complexiteit in de GSP code wordt onder andere veroorzaakt door de hoeveelheid embedded JavaScript code.



Figuur 8: Verdeling van de complexe code over de verschillende technologieën

3.2.5 Unit Interfacing (3.8)

Uit de metingen blijkt dat 1% van de Portero code zich bevindt in units met een grote tot zeer grote interface (> 4 parameters). De unit interfacing van het Portero systeem scoort boven marktgemiddeld.

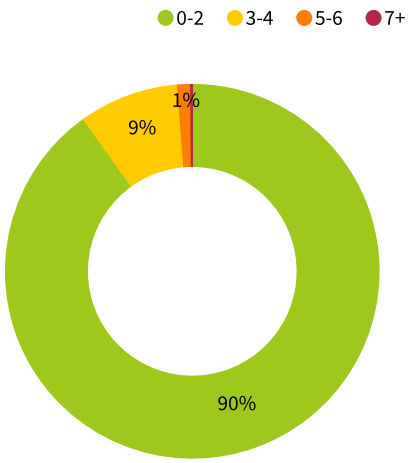
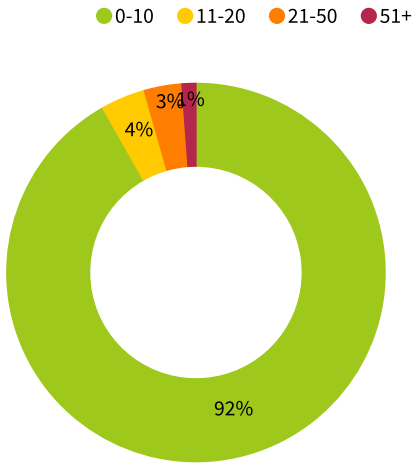


Figure 9: Unit-interfacing scoort boven marktgemiddeld

3.2.6 Module Coupling (4.2)

Uit de metingen blijkt dat 4% van de code zich bevindt in modules met een hoog tot zeer hoog aantal inkomende afhankelijkheden (> 20 inkomende afhankelijkheden). De modules binnen het Portero systeem kennen een boven marktgemiddelde mate van ontkoppeling.

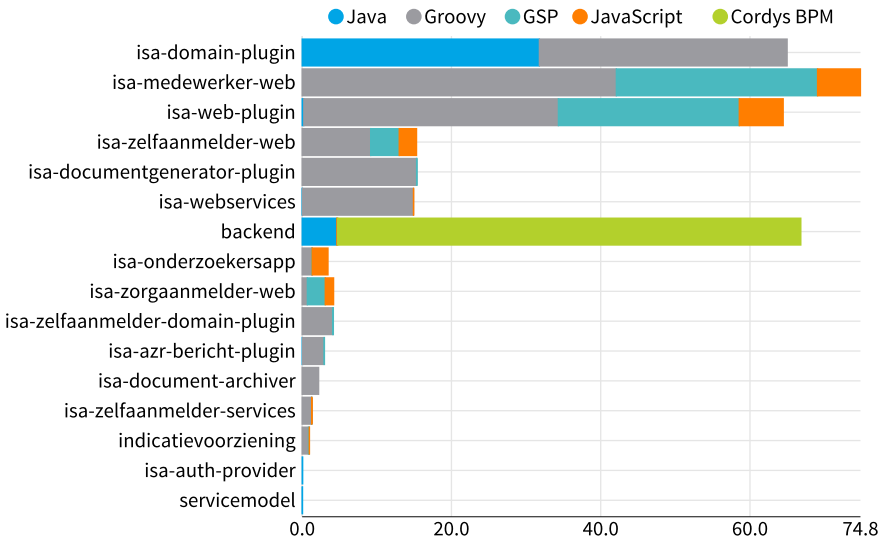


Figuur 10: Module coupling for the Portero applicatie

3.2.7 Component balance (2.6)

De code van het Portero systeem is verdeeld over 16 componenten. In vergelijking met de benchmark van SIG kent het Portero systeem een marktgemiddelde verdeling van code over componenten.

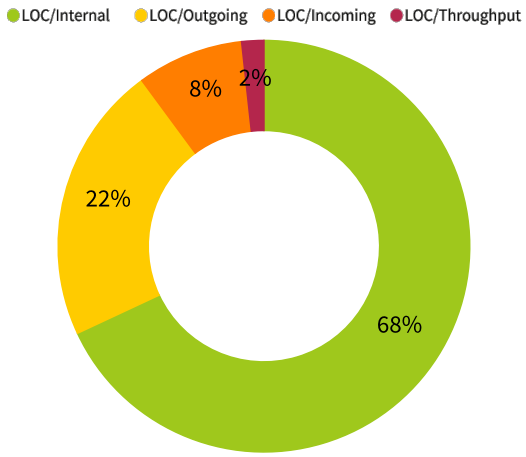
Figuur 11 geeft een overzicht van de verdeling van het codevolume (in manmaanden) over de verschillende componenten. De componenten *isa-domain-plugin*, *isa-medewerker-plugin*, *isa-web-plugin* en *backend* zijn relatief groot in vergelijking met de overige componenten. Daarnaast zijn de componenten *isa-onderzoeksapp*, *isa-zelfaanmelder-web*, *isa-zelfaanmelder-domain-plugin*, *isa-azr-bericht-plugin*, *isa-document-archiver*, *isa-zelfaanmelder-services*, *indicatievoorziening*, *isa-auth-provider* en *servicemodel* relatief klein ten opzichte van de overige componenten.



Figuur 11: Componenten van de Portero applicatie, hun omvang in manmaanden en technologieën

3.2.8 Component independence (4.1)

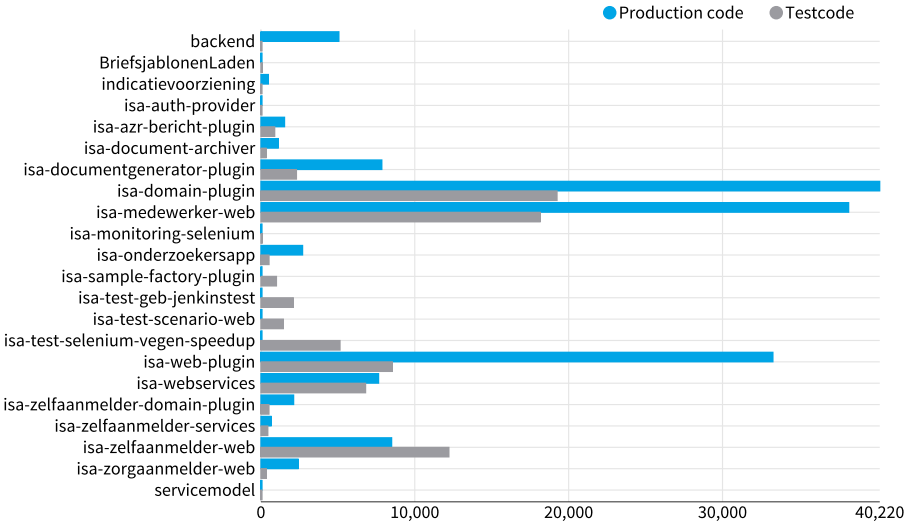
Uit de metingen blijkt dat 10% van de code aangeroepen wordt vanuit andere componenten (som van *Incoming* en *Throughput*). De componenten binnen Portero zijn boven marktgemiddeld onafhankelijk van elkaar.



Figuur 12: Component independence van Portero

3.2.9 Omvang van de testcode

De omvang van de testcode van het Portero systeem is 80.325 Lines of Code. De omvang van de testcode is 53% van de omvang van de productiecode. Figuur 13 geeft een overzicht van de verhouding tussen testcode en productiecode per component.



Figuur 13: Verhouding testcode en productiecode per component

3.2.10 Omvang van de database code

Het Portero systeem maakt gebruik van een Microsoft SQL Server database voor persistentie van de gegevens van de applicatie. De database bevat 221 tabellen, 6 views en 13 stored procedures welke onderdeel zijn van de productiecode. De gegeven omvang van de database code is gebaseerd op de export van tabellen, views en stored procedures welke onderdeel zijn van de applicatie in productie. Deze export is niet meegenomen in de analyse van de onderhoudbaarheid.

Artefact	Aantal	Omvang (lines of code)
Tables (definitie)	221	9.674
Views (definitie)	6	453
Stored Procedures	13	2.268

Tabel 3: Overzicht van tabellen, views en stored producedures

Voor communicatie met de database wordt gebruik gemaakt van 2 verschillende *Object Relational Mapping frameworks*, te weten GORM en Hibernate.

3.2.11 Gebruikte talen – Nederlands en Engels door elkaar heen

De code die onderdeel is van het probleemdomein is grotendeels geschreven met Nederlandstalige variabelen, modulenamen en commentaar. De code die onderdeel is van het oplossingsdomein is grotendeels geschreven in het Engels.