Software Improvement Group

Logius – Onderzoek naar Digipoort
Eindpresentatie

Inhoudsopgave

Achtergrond en context

Achtergrond

Logius biedt voor **publieke organisaties producten** en **diensten** aan waardoor burgers en bedrijven **digitale zaken** kunnen regelen met de **overheid**.

Digipoort, een dienst van Logius, fungeert als het **digitale knooppunt** voor de **uitwisseling** van **gegevens** tussen **bedrijven** en **overheidsinstanties** in Nederland.

Logius beheert zowel Digipoort **als** de **standaarden**. Daarnaast zorgt Logius voor de **beschikbaarheid**, **juiste werking**, **continuïteit** en **beveiliging** van Digipoort.

Het **Adviescollege ICT** (AcICT) heeft recent **onderzoek** gedaan naar het **herbouwde Digipoort**.

Context

Logius heeft **SIG gevraagd** om het **herbouwde Digipoort** te **beoordelen**. Men wil vaststellen wat de kwaliteit en de complexiteit van de software en de software-architectuur is. Mede gelet op de gevolgen die het kan hebben op de **beheerfase**.

Meer specifiek zal SIG de volgende aspecten onderzoeken:

- De **technische kwaliteit** van Digipoort;
- De **architectuur** van het systeem (zie volgende slide);
- De geschatte **jaarlijkse inspanning** voor het **technisch beheer** en onderhoud van het systeem.

Onderzoeksvragen

SIG heeft samen met Logius de volgende onderzoeksvragen opgesteld:

1. Wat is de **onderhoudbaarheid** van het Digipoort-systeem op basis van de ISO 25010-standaard voor softwarekwaliteit, in vergelijking met 20.000 andere systemen uit de SIG-benchmark?

2. Hoe **flexibel** is de **architectuur** van het systeem en in hoeverre zijn *best practices* toegepast zodanig dat veranderende businesswensen kunnen worden ondersteund?

3. Wat is de geschatte **jaarlijkse inspanning** voor **beheer** en **onderhoud** van het Digipoort-systeem?

4. **Welke zaken** dienen te worden **verbeterd** uit oogpunt van de huidige kwaliteit en de **ambities** van het Digipoort-systeem en **welke prioritering is nodig**?

Over Software Improvement Group

GETTING SOFTWARE RIGHT
FOR A HEALTHIER DIGITAL WORLD

Wetenschappelijk onderzoek

De onderzoeksafdeling van SIG ontwikkelt onze meetmodellen en draagt bij aan vooruitgang op het gebied van software engineering.

Sigrid

Het SIG software assurance platform biedt continu inzicht in de kwaliteit, kosten en risico's van software.

Benchmarking

De database voor softwareanalyse van SIG is de grootste ter wereld en bevat meer dan 200 miljard coderegels.

Technologiebedrijf met geïntegreerde adviesdiensten

SIG biedt op feiten gebaseerd advies om organisaties te helpen van hun software een groeimotor te maken.

TÜViT® Certificatie

SIG is het eerste bedrijf ter wereld met een door TÜViT geaccrediteerd lab dat software certificeert voor ISO 25010.

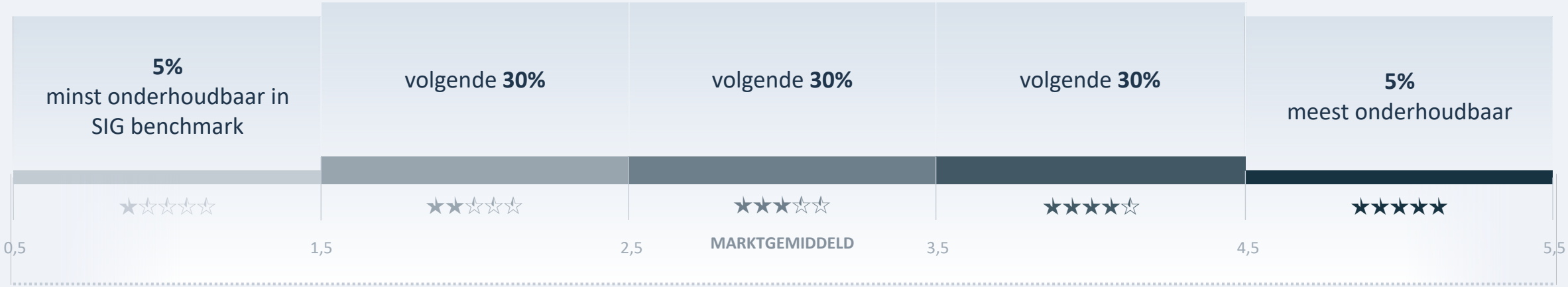
We meten volgens de ISO 25010 standaard voor Software Product Kwaliteit

De ISO 25010 is een **wereldwijde standaard** voor softwarekwaliteit en beschrijft kwaliteitseigenschappen van software op 8 verschillende aspecten. Software Improvement Group heeft modellen ontwikkeld welke 6 van deze 8 aspecten meetbaar maken.

Voor de certificering van het model voor onderhoudbaarheid werkt SIG samen met TÜViT. Door deze samenwerking heeft Software Improvement Group het **eerste volledig gecertificeerde laboratorium ter wereld** om te meten volgens de **ISO 25010 standaard**.

In aanvulling op deze modellen biedt SIG ook **vele andere diensten** die helpen om inzicht te krijgen in de risico's waaraan uw software (en bedrijf) worden blootgesteld.

Beoordeling van onderhoudbaarheid volgens ISO 25010 is gebaseerd op de SIG benchmark



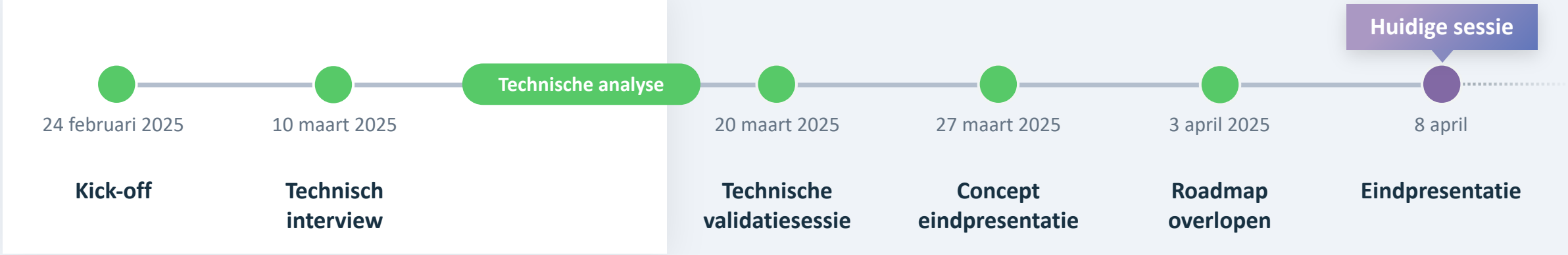
Onderhoudbaarheid wordt vastgesteld door geautomatiseerde broncode-analyse.

- SIG meet **een set van eigenschappen** van broncode, als gedefinieerd door de SIG/TÜViT Evaluation Criteria for Trusted Product Maintainability.
- De uitkomsten worden vergeleken met de **SIG Benchmark, welke jaarlijks wordt gekalibreerd uit een dataset** van duizenden systemen. Dit resulteert in een sterrenscore.

Planning

Fase 1: **Initiatie en analyse**

Fase 2: **Advies**



Digipoort is goed onderhoudbaar; architectuur is flexibel en schaalbaar

Logius heeft SIG gevraagd om inzicht te geven in de technische kwaliteit van de herbouwde Digipoort. SIG onderzocht hiervoor **onderhoudbaarheid**, **architectuur** en de **beheerlast** van de nieuwe applicatie, welke nog in ontwikkeling is en deels in productie. Hiervoor heeft SIG gebruik gemaakt van modellen voor onderhoudbaarheid, architectuurkwaliteit en toetsing aan *best practices* voor *service-oriented* applicaties.

Met een omvang van 11,1 persoon-jaren is de herbouwde Digipoort een klein systeem volgens de SIG benchmark. Het systeem bestaat uit Java-code in de back-end en Angular/Typescript-code in de front-end. Digipoort scoort boven-marktgemiddeld op **onderhoudbaarheid** (★★★★☆; 3,9). Belangrijkste verbeterpunt tav. onderhoudbaarheid is het beheer van bibliotheken en het bundelen van XSD's in bibliotheken.

Digipoort is ontwikkeld conform een *service-oriented* **architectuur**. De services zijn ontkoppeld, wat mede de onderhoudbaarheid, flexibiliteit en schaalbaarheid bevordert.

- Het Kafka event-streaming platform wordt gebruikt en verzorgt de communicatie (obv. events) tussen services. Ondanks dat de communicatie via Kafka het inrichten van complexe workflows toelaat, heeft dit geen effect op de architecturale verwevenheid.
- Documentatie is grotendeels op orde. De huidige eisen en de toekomstplannen (in de roadmap) van de herbouwde Digipoort vragen om flexibiliteit en schaalbaarheid en geven een rechtvaardiging voor de keuze van een service-oriented architectuur.
- Het gebruik van ActiveMQ is niet conform de technologiekeuzes van Digipoort (i.e. Kafka); hierdoor kan extra complexiteit ontstaan in het systeem en het voortbrengingsproces.
- Microservices worden momenteel niet geautomatiseerd uitgerold; administratieve handelingen zorgen ervoor dat releases slechts één keer per maand plaatsvinden. Volledige automatisering is een *best practice*.

Sterke fundamenten aanwezig voor functionele groei

De geschatte **onderhoudslast*** bedraagt ±2-3 FTE's per jaar (±4-5 FTE per jaar bij oplevering van de herbouwde Digipoort op basis van de verwachte groei); dit is de inspanning om de huidige codebase in stand te houden en maakt deel uit van de totale werkzaamheden rondom Digipoort, zonder vergroting van de technische schuld en zonder grootschalige projectmatige uitbreidingen. Momenteel zijn er ±37 FTE's betrokken bij de ontwikkeling van de huidige codebase.

De ontwikkeling van de herbouwde Digipoort is nog gaande. De gekozen oplossing is zeer flexibel en schaalbaar en biedt daardoor de fundamenten om functioneel en niet-functioneel te groeien.

SIG adviseert het volgende om de kwaliteit van Digipoort in de toekomst te blijven borgen:

- ○ **[Roadmap]** Documenteer de doelstellingen, de afstemming op de bedrijfsdoelen, interne en externe afhankelijkheden, de beschikbaarheid van middelen en mogelijke risico's, zodat de haalbaarheid en strategische afstemming worden gewaarborgd.
- ○ **[Migratie]** Borg dat na de oplevering van Digipoort de belangen van Logius (zoals *business requirements*) behartigd blijven. Het juist beleggen van het product-ownerschap is hier een onderdeel van.
- ● **[Onderhoudbaarheid]** Bundel XSD's in bibliotheken om zowel duplicatie als onderlinge afhankelijkheden tussen de microservices te verminderen. Documenteer kwaliteitseisen en neem deze actief mee in elke sprint. Gebruik tooling (ie.: SonarQube en/of Sigrid) om de kwaliteit inzichtelijk te maken en de kwaliteitseisen te toetsen.

* Onderhoudslast omvat enkel technische updates, bugfixes en kleine functionele verbeteringen, zoals beleidsgestuurde aanpassingen; overhead zoals vergaderingen, planning of rapportage vallen hier niet onder.

Niet-functionele eisen behoeven prioriteit; stel een plan op om ActiveMQ te vervangen door Kafka

- [Niet-functionele eisen] Prioriteer de adressering van de volgende niet-functionele eisen te adresseren in relatie tot Digipoort: adresseer stabiliteitsproblematiek door gebruik te maken van KRaft (Kafka Raft), gelijkgestemde omgevingen (ontwikkelomgeving vs. productieomgeving), CPU/RAM-resource gebruik, etc.

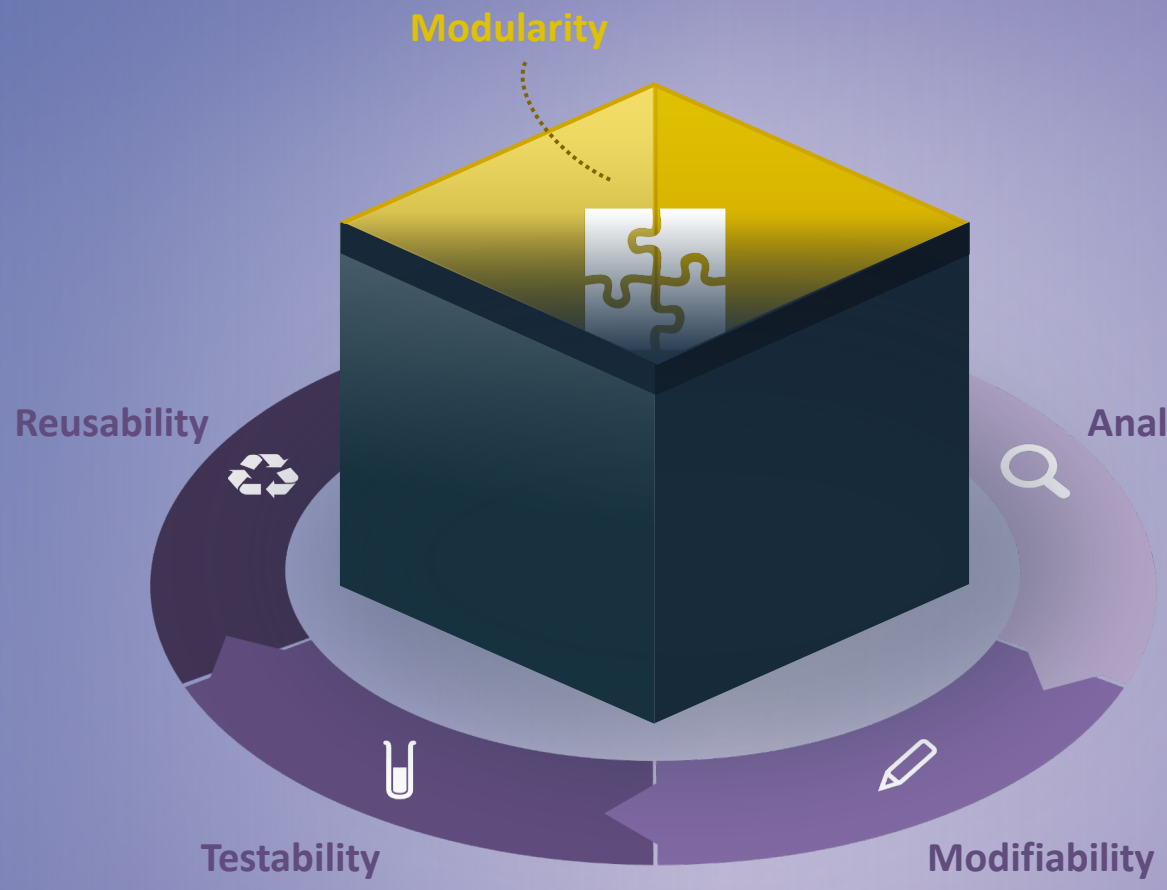
- [Life-cycle management] ActiveMQ maakt deel uit van de aanwezige technologieën door de afhankelijkheid op een intern Logius-component dat niet onder het beheer van het Digipoort-team valt. Stel een plan op om deze technologie te vervangen door Kafka, wat aansluit op de huidige werking van Digipoort en de bredere technologiewensen van Logius.

Inhoudsopgave

Explanation: The ISO 25010 standard for Maintainability has 5 sub-characteristics

Maintainability ISO 25010

- Analyzability
- Modifiability
- Testability
- Modularity
- Reusability



The ISO 25010 standard refers to **maintainability** as:

"The degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements."

Maintainability according to the ISO 25010 standard has five sub-characteristics. These sub-characteristics can be seen as a **representation of the phases** that are passed when performing maintenance work.

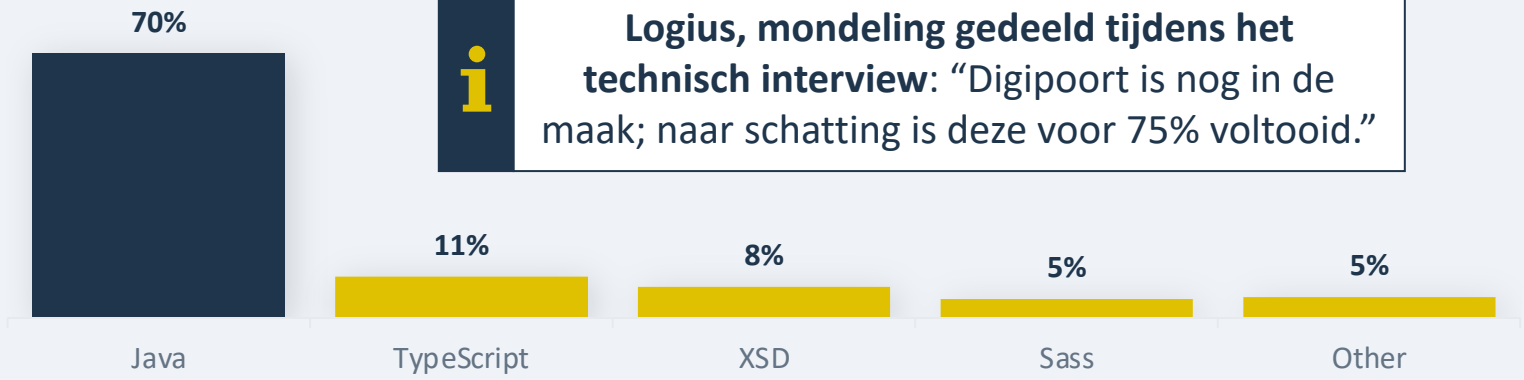
Digipoort heeft een boven-marktgemiddelde onderhoudbaarheid (★★★★☆)

- Digipoort bestaat uit Java-code (back-end) en TypeScript (front-end).
- De mate van duplicatie (19%) in Digipoort is martktgemiddeld.
- Op code-niveau scoort Digipoort boven het marktgemiddelde.
  - Het systeem bestaat voornamelijk uit korte, niet-complexe methodes.
  - De interfaces van methodes zijn klein en vormen geen risico.
- Modules zijn boven marktgemiddeld ontkoppeld.
- Gedeelde bibliotheken zorgen voor enkele afhankelijkheden tussen software-componenten, dit beïnvloedt de scores negatief.
- Digipoort wordt uitvoerig getest; er is een test-tot-productiecode ratio van 124%.

Volume	★★★★☆	4.3
Duplication	★★★☆☆	2.8
Unit size	★★★★☆	4.0
Unit complexity	★★★★☆	4.0
Unit interfacing	★★★★☆	3.7
Module coupling	★★★★☆	4.0
Component entanglement	★★★☆☆	3.4
Component independence	★★★★★	4.5

Digipoort bestaat uit Java code met Angular en TypeScript voor de front-end, omvang is 11 PJ

- De back-end componenten zijn geschreven in Java met het Spring framework.
- De front-end portalen zijn geschreven Angular en TypeScript.
- XML Schema definities zijn gedefinieerd in XSD voor het structureren en valideren van XML documenten (input/output).



Logius, mondeling gedeeld tijdens het technisch interview: “Digipoort is nog in de maak; naar schatting is deze voor 75% voltooid.”

Technologie	LOC	Herbouwgrootte (PM)
Java	70.124	93
TypeScript	9.980	15
XSD	11.380	11
Sass	4.332	7
Other (Angular Templates, XSLT, WSDL, JS)	5.873	7

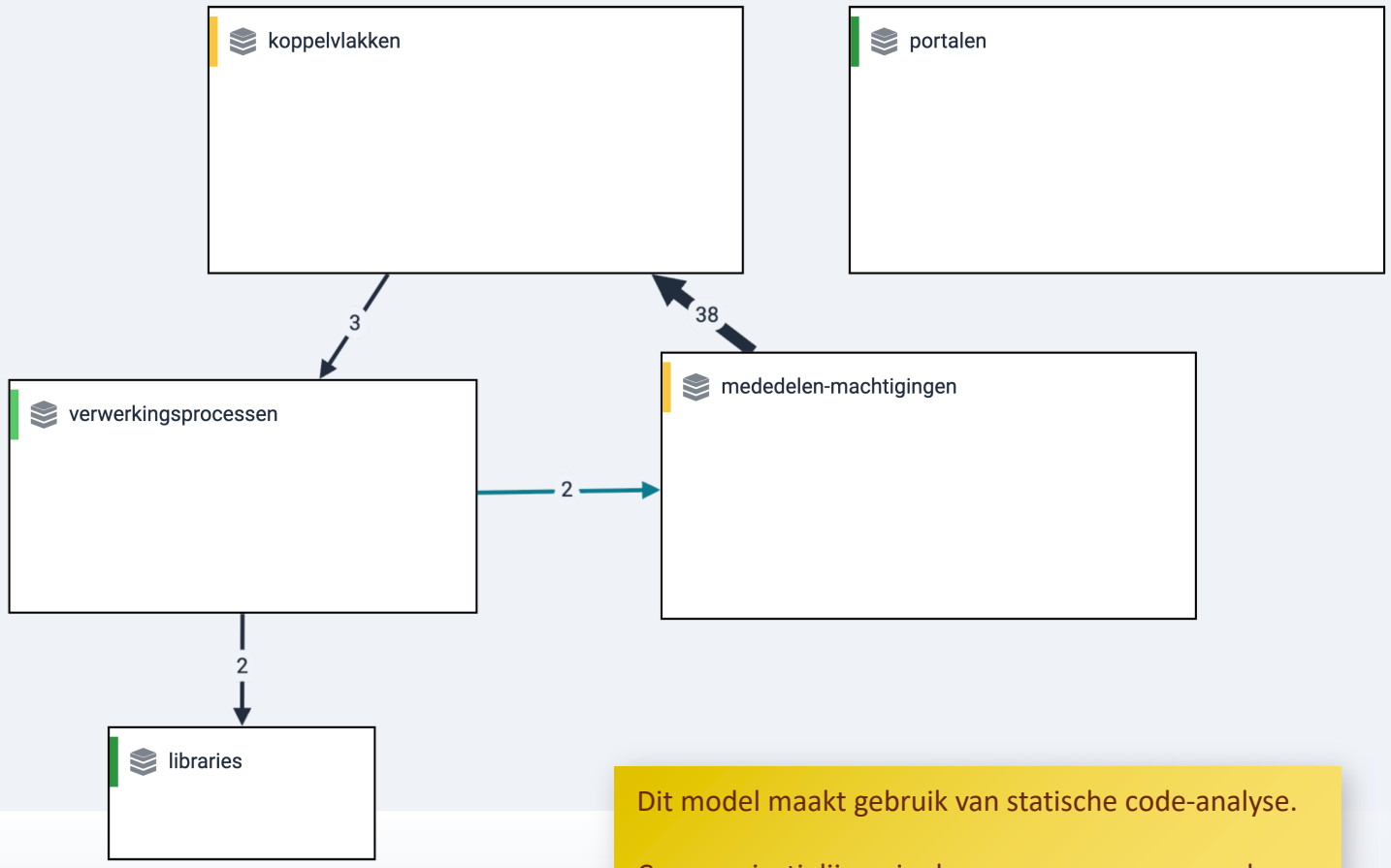
Volume 4.3 ★★★★☆

Totaal systeemvolume: **101.689 LOC** **11.1 PJ**

PM = Persoon-maanden, **PJ** = Persoon-jaren

De component entanglement wordt beïnvloed door inconsistent gebruik van bibliotheken

- Er is een indirecte cyclische afhankelijkheid tussen verwerkingsprocessen en mededelen-machtigingen.
- Het bibliotheken-component wordt enkel aangeroepen door verwerkingsprocessen.
- Mededelen-machtigingen gebruikt bibliotheken die zijn gedefinieerd in koppelvlakken.



Component entanglement 3.4 ★★★☆☆

Dit model maakt gebruik van statische code-analyse.

Communicatielijnen in de vorm van queues worden niet meegenomen in de onderhoudbaarheidsscores.

SIG's raamwerk voor het beoordelen van een *Service-oriented* architectuur

Het hier gebruikte raamwerk beschrijft 10 eigenschappen die bijdragen aan een robuuste, schaalbare en toekomstvaste *Service-oriented* architectuur.

Deze eigenschappen hebben invloed op:

- **Onderhoudbaarheid**: Het gemak van het updaten, aanpassen of vervangen van individuele services.
- **Wendbaarheid**: De flexibiliteit waarmee de architectuur snel kan reageren op veranderende bedrijfsbehoeften.
- **Schaalbaarheid**: Services kunnen onafhankelijk worden geschaald, waardoor groei wordt ondersteund.
- **Vereenvoudigde integratie**: Het kan zorgen voor compatibiliteit en gestroomlijnde integratie tussen services of externe systemen.

Eigenschap	Beschrijving
Ontkoppeling	Services werken onafhankelijk, met minimale afhankelijkheden.
Geautomatiseerde uitrol	Services kunnen onafhankelijk en geautomatiseerd worden uitgerold naar productie.
Interoperabiliteit	Services communiceren via gestandaardiseerde protocollen (e.g.: REST, Kafka, RabbitMQ, etc.).
Herbruikbaarheid	Services zijn ontworpen voor hergebruik in meerdere applicaties.
Ontdekbaarheid	Diensten zijn gemakkelijk te vinden via registers of directories.
Abstractie	Services verbergen interne logica en geven alleen de vereiste interfaces weer.
Statelessness	Services behandelen verzoeken onafhankelijk zonder sessie-informatie op te slaan.
Samenstelbaarheid	Complexe functies worden samengesteld door meerdere eenvoudigere services te combineren.
Gestandaardiseerde contracten	Duidelijk gedefinieerde interfaces specificeren inputs, outputs en gedrag van services.
Documentatie	Verduidelijkt het doel van services, wat helpt bij de afstemming tussen technische implementatie en bedrijfsstrategie. Daarnaast borgt documentatie kritieke informatie en vermindert de afhankelijkheid van individuele expertise.

Flexibiliteit en schaalbaarheid via Kafka

De **architectuur scoort goed** op het gebied van **ontkoppeling**. **Microservices** functioneren **onafhankelijk** mede dankzij communicatie via Kafka en het gebruik van **unieke databases per service**, wat bijdraagt aan schaalbaarheid en flexibiliteit. Hoewel er sprake is van expliciete code-aanroepen via bibliotheken tussen componenten, vormt dit geen significante beperking.

Hoewel microservices **individueel uitgerold** kunnen worden, zijn ze **functioneel afhankelijk** van elkaar. Administratieve handelingen zorgen er voor dat **releases** slechts **één keer per maand** plaatsvinden; volledige automatisering wordt niet optimaal benut.

Kafka zorgt voor **interoperabiliteit** en **samenstelbaarheid**, wat de flexibiliteit verhoogt bij het ontwerpen van complexe workflows. Ook is sprake van een **goede abstractie** en ***statelessness***, wat bijdraagt aan **eenvoudiger onderhoud**, **robuustheid** en **schaalbaarheid**.

Gestandaardiseerde contracten via Kafka-events en XSD-schema's dragen bij aan **data-integriteit** en hierdoor zijn **services niet afhankelijk** van de **volgorde** waarin **events** verwerkt worden.

De **documentatie** is **grotendeels op orde** middels Architecture Decision Records (ADR's). De **huidige eisen** en de **toekomstplannen** (in de roadmap) van de herbouwde Digipoort **vragen** om **flexibiliteit** en **schaalbaarheid** en geven een **rechtvaardiging** voor de **keuze** van een ***Service-oriented* architectuur**.

Flexibiliteit en schaalbaarheid via Kafka

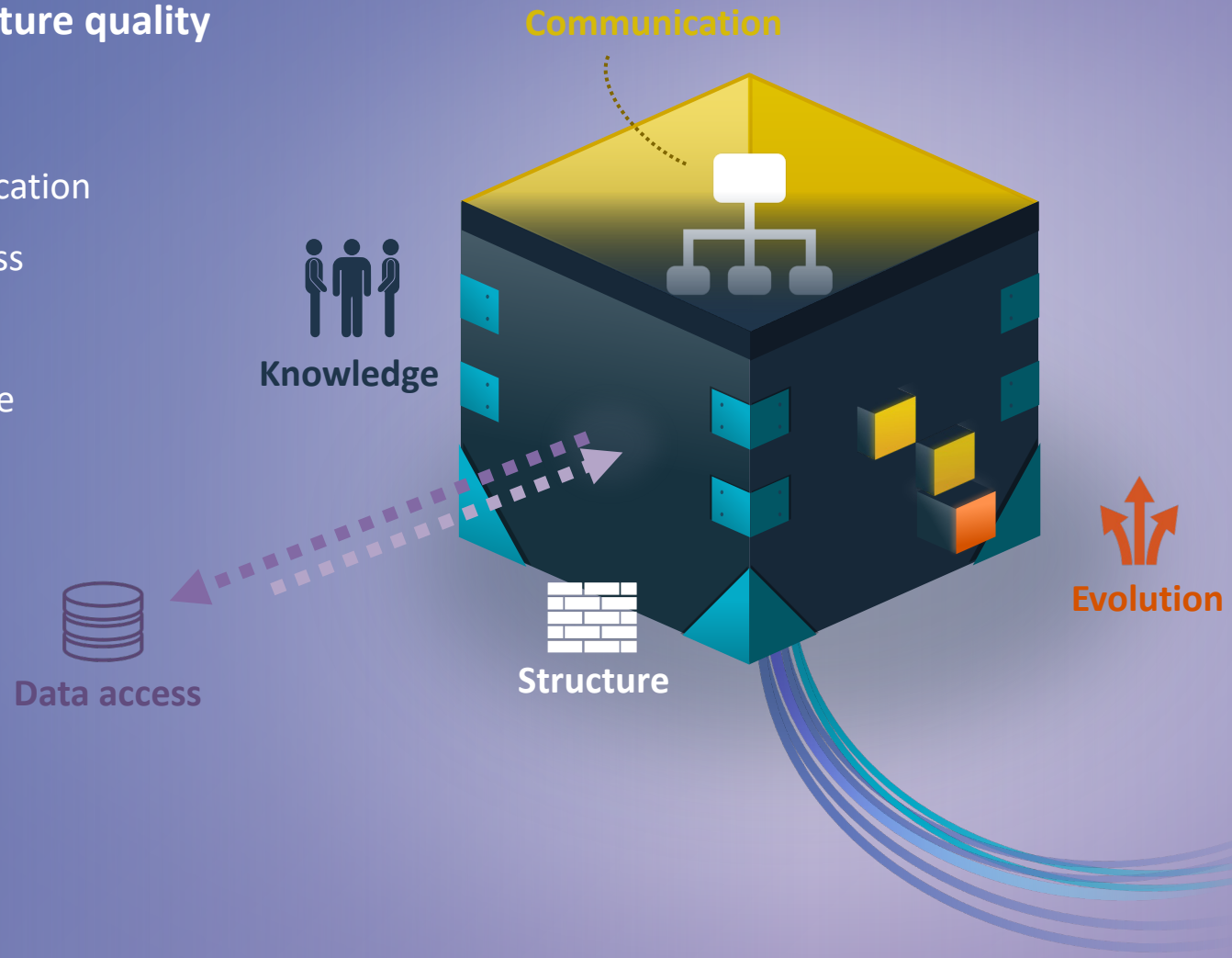
Eigenschap	Toegepast?	Toelichting
Ontkoppeling	±	+ De aanwezige services werken onafhankelijk van elkaar en communicatie is boven-marktgemiddeld gecentraliseerd binnen componenten. + Kafka wordt gebruikt om tussen services te communiceren, wat eenvoudige en complexe stappenreeksen toelaat. + Dataopslag is losgekoppeld met unieke databases voor elke service. ± Expliciete code-aanroepen tussen componenten gebeuren doordat ze bibliotheek-functionaliteiten uit andere componenten gebruiken.
Geautomatiseerde uitrol	±	± Microservices kunnen onafhankelijk van elkaar worden uitgerold, maar hebben elkaar nodig om te fungeren. ± Geautomatiseerde uitrol is mogelijk, maar administratieve handelingen resulteren in een handmatige release en uitrol (1x/maand).
Interoperabiliteit	+	+ Services communiceren met elkaar gebruikmakende van Kafka.
Herbruikbaarheid		N.v.t.
Ontdekbaarheid	+	+ De routeringservice zal voor een naadloze transitie zorgen van het oude naar het nieuwe Digipoort.
Abstractie	+	+ Services verbergen interne logica. + Inter-service-communicatie wordt door Kafka gefaciliteerd; de nodige consumers en producers zorgen voor een gestroomlijnde interface.
Statelessness	+	+ De microservices van Digipoort werken *stateless*, wat onder meer een *active-active* setup ondersteunt. + Zie bevinding over gecentraliseerde communicatie bij “Ontkoppeling”.
Samenstelbaarheid	+	+ Kafka topics worden gebruikt om procedurele stappen te definiëren; stappen worden afgehandeld door een service. Op deze manier kunnen complexere stappenreeksen worden geprogrammeerd.
Gestandaardiseerde contracten	+	+ Alle Kafka events in een specifiek proces volgen hetzelfde schema, met hetzelfde interface. Dit zorgt ervoor dat de individuele iServices niet afhankelijk zijn van de volgorde waarin ze uitgevoerd worden. + XSD’s zorgen voor een gestructureerde datavalidatie.
Documentatie	+	+ Documentatie wordt uitgebreid bijgehouden met onder andere Architecture Decision Records (ADR’s). + Argumentatie betreffende de keuze voor een *service-oriented* architectuur is aanwezig. + Roadmap beschrijft (niet-)functionele groei (korte, middellange en lange termijn).

LEGENDA: Toegepast — Niet toegepast

The Architecture Quality model has 5 sub-characteristics

Architecture quality

- Structure
- Communication
- Data access
- Evolution
- Knowledge



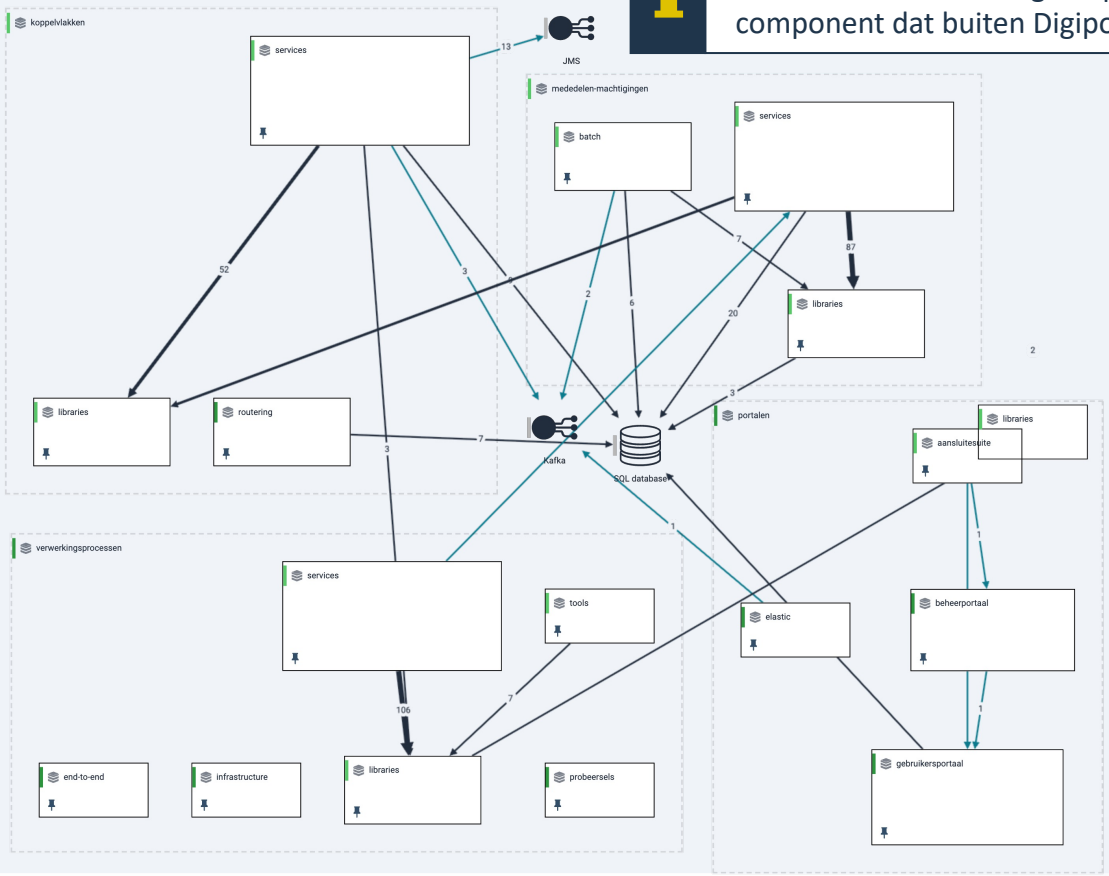
The SIG **Architecture Quality** model provides insight into the ability for an application to evolve as business needs change.

Each of the 5 sub-characteristics represent properties of software architecture that **effect the velocity in which foundational changes can be made** to a software system.

Het percentage interne code in componenten ligt boven het marktgemiddelde

- Enkele componenten hebben relatief veel communicerende code door het hele component, verspreid over het hele component:
  - Aanlever-validation-persistance-library (26%)
  - Bcr-ftp-integration-library (57%)
  - Machtiging-api (33%)

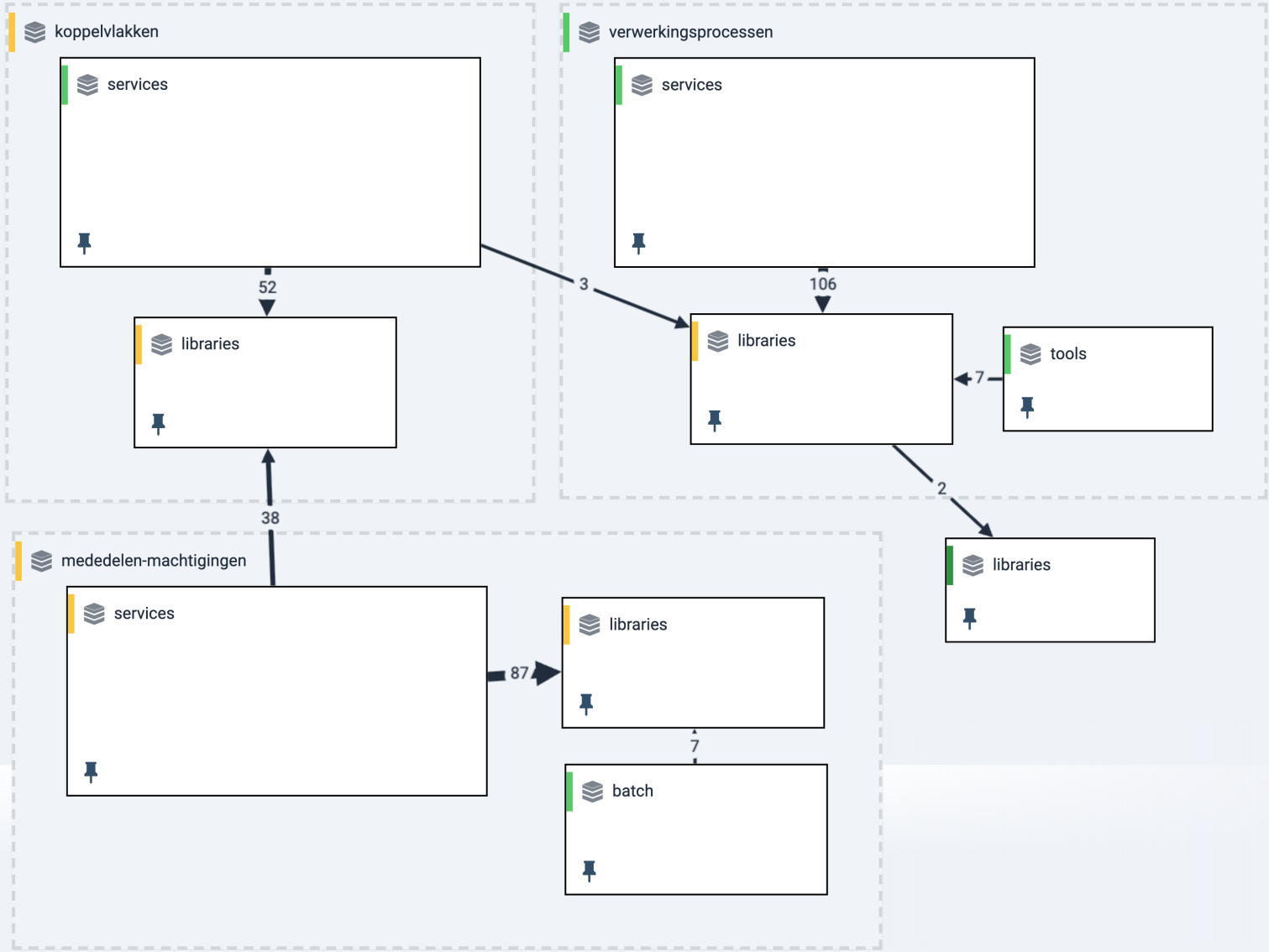
Logius: “ActiveMQ is nodig voor OSDBK (Open-Source Dienst Beveiligd Koppelvlak); een Logius-component dat buiten Digipoort wordt ontwikkeld.”



Communication Centralization **4.7** ★★★★★

Componenten binnen Digipoort zijn boven-marktgemiddeld losgekoppeld

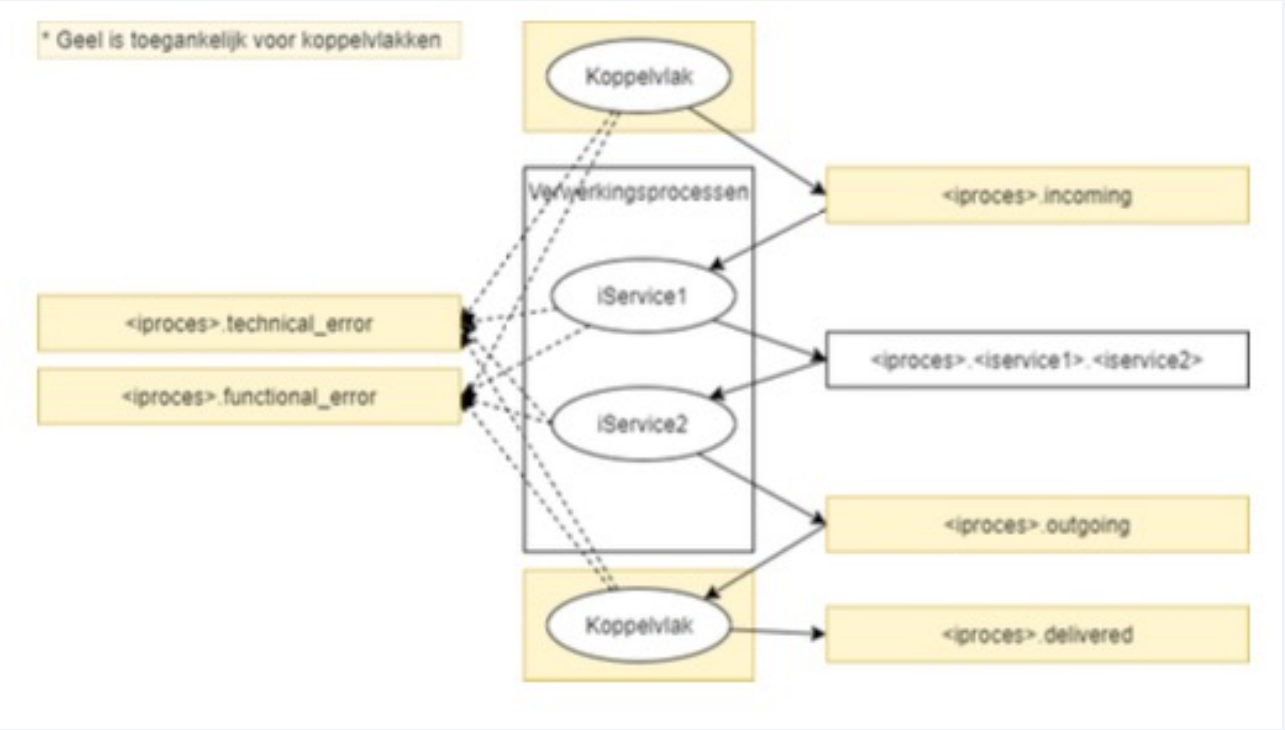
- Gedeelde functionaliteiten van services zijn gedefinieerd in bibliotheken.
- Het gebruik van bibliotheken in hoofdcomponenten is niet altijd beperkt tot enkel dat hoofdcomponent.
  - Mededelen-machtigingen gebruikt bibliotheken in koppelvlakken.
  - Koppelvlakken gebruikt bibliotheken in verwerkingsprocessen.
- Componenten in services hebben geen afhankelijkheden van elkaar in de vorm van directe code of interface calls.



Component Coupling 4.0 ★★★★☆

Tussen iServices is de communicatie procedureel gedefinieerd aan de hand van Kafka topics

- Elk berichttype maakt gebruik van specifieke Kafka-topics voor communicatie.
- Koppelvlakken sturen berichten naar verwerkingsprocessen via *.incoming en ontvangen berichten via *.outgoing.
- Binnen verwerkingsprocessen wisselen iServices berichten uit via unieke topics voor elke combinatie van berichttype, verzendende service en ontvangende service.
- Hierdoor ontstaat een vaste volgorde van iServices die elk berichttype moet doorlopen.



- Alle Kafka events in dit proces volgen hetzelfde schema, met hetzelfde interface. Eventuele MinIO referenties zijn immutable. Het enige dat wijzigt in de Kafka metadata tussen de topics is een status flag.
- Kafka events met hetzelfde schema zorgen er voor dat de individuele iServices niet afhankelijk zijn van de volgorde waarin ze uitgevoerd worden. De reden voor een vaste volgorde heeft zijn oorzaak in externe requirements (metrokaart), maar kan op technisch vlak losgelaten worden.

Onderhoudslast bedraagt 2-3 FTE per jaar; gemiddelde aantal auteurs per component is net voldoende

- Het grote aantal auteurs per component kan duiden op betrokkenheid van meerdere experts, actieve kennisdeling en/of verloop van ontwikkelaars.
- Het gemiddeld aantal auteurs bij een complete herbouw met een voldoende kennisdeling wordt geschat op 16,1.
- SIG schat dat er 2-3 FTE nodig zijn om de huidige codebase te onderhouden***.
- SIG schat dat op basis van de verwachte groei en oplevering van Digipoort (ie.: broncode) de onderhoudslast 4-5 FTE per jaar bedraagt.

Logius meldt dat er op dit moment (ie.: ontwikkelfase) ±37 FTE (1 platform teams en 3 feature teams) werkt aan Digipoort.

Component	Lijnen code	Aantal auteurs*	Gemiddeld aantal auteurs**
Koppelvlakken	35K	25	**3,2**
Mededelingen-Machtigingen	34K	12	**2,3**
Portalen	32K	30	**6,4**
Verwerkingsprocessen	28K	33	**3,8**
Bibliotheken	438	6	**0,4**
		Totaal	**16,1 auteurs**

* Het totale aantal personen dat heeft bijgedragen aan dit component binnen de geselecteerde tijdsperiode. Elke persoon dat een vastlegging heeft gedaan voor dit deel van de codebase wordt meegenomen in deze telling.

** Het lopende gemiddelde van het personeel dat actief bijdraagt aan de component binnen de geselecteerde tijdsperiode. SIG neemt het gemiddelde als een schatting van de grootte van het kernteam om het effect van auteurs die slechts weinig of sporadisch bijdragen aan de component te verminderen.

*** Gebaseerd op een marktgemiddelde 15% van de code die jaarlijks wordt aangeraakt bij actief onderhoud. Onderhoudslast omvat technische updates, bugfixes en kleine functionele verbeteringen, zoals beleidsgestuurde aanpassingen. De ontwikkeling van nieuwe functies en overhead zoals vergaderingen, planning of rapportage vallen hier niet onder.

Knowledge Distribution 3.1 ★★★☆☆

PM = Persoon-maanden, **PJ** = Persoon-jaren

Er is gekozen voor een micro-services architectuur met generieke koppelvlakken

De volgende uitgangspunten in de architectuur zijn gekozen om te voldoen aan de eisen van schaalbaarheid, efficiëntie, veiligheid en toekomstbestendigheid, welke kritisch zijn voor Digipoort:

1. Koppelvlakken moeten schaalbaar kunnen worden ingezet, zodat ze flexibel kunnen omgaan met variaties in berichtenvolumes. De huidige Digipoort behandelt ongeveer 75 miljoen berichten per jaar.
   - De eisen van de herbouwde Digipoort: 50 berichten per seconde (± 4 MB berichtgrootte), 5 berichten per minuut (±100 MB berichtgrootte)
   - Er worden ±19 nieuwe berichtsoorten toegevoegd (zie volgende slide), wat gepaard gaat met een toename van toekomstige berichtenstromen.
2. Container technologie zorgt voor flexibiliteit en schaalbaarheid, wat leidt tot beter resource management en kostenbesparend kan werken.
3. Informatie wordt decentraal beheerd vanuit de services. Er is een single point of truth; een uitgangspunt zowel van NORA* als Logius.
4. Logius wil decompositie in afgebakende functionaliteit door services en API's. De microservices kunnen onafhankelijk van elkaar onderhouden, incrementeel doorontwikkeld en gereleased worden. Dit ondersteunt continue verbetering en aanpassing aan nieuwe initiatieven. Daarnaast leveren teams waarde aan de gebruiker en zijn verantwoordelijk voor hun eigen (deel)product, dit verhoogt wendbaarheid en efficiëntie.
5. Technische aansluitingen generiek neerzetten voor Gegevensuitwisseling en Logius Voorzieningen (initiele scope: Digipoort, BBO) bevordert hergebruik, efficiëntie, consistentie en interoperabiliteit tussen ketendeelnemers.
6. Het "Design for Failure"-principe zorgt bij uitval voor minimale impact en snel herstel; geïsoleerd falen en heropstarten van microservices zijn mogelijk door het gebruik van container technologie. Daarnaast zijn er hoge eisen gesteld aan betrouwbaarheid, foutbestendigheid en recovery (RTO: 4 uur, RPO: 0; synchrone database-replicaties over verschillende beschikbaarheidszones).

* Nederlandse Overheid Referentie Architectuur, NAP12

Bron: Appendix C - Solution Intent_v2.1.pdf, Solution Architectuur Digipoort-v55-20250224_144419.pdf

Beoogde (niet-)functionele groei ondersteunen de keuze voor de huidige architectuur

Logius heeft een roadmap opgesteld (zie rechts in hoofdlijnen) waarin de korte, middellange en lange termijn ontwikkelingen zijn opgenomen.

De roadmap duidt op een toekomstige, (niet-)functionele groei. De architectuurkeuzen (vorige slide) kunnen deze groei faciliteren:

- De herbouwde Digipoort heeft een *service-oriented* architectuur,
- Generieke koppelvlakken,
- De componenten zijn configureerbaar,
- Gebruik van containertechnologie,
- De technische kwaliteit van het systeem is boven-marktgemiddeld.

SIG heeft de juistheid van de roadmap verder niet beoordeeld. Om de haalbaarheid en strategische afstemming voor deze roadmap te waarborgen, moeten ook duidelijke doelstellingen, de afstemming op de bedrijfsdoelen, interne en externe afhankelijkheden, de beschikbaarheid van middelen en mogelijke risico's worden vastgelegd en getoetst.

Korte termijn ontwikkelingen (nu – Q4 2025)

- **Berichtsoorten toevoegingen/aanpassingen**: ± 19 nieuwe berichtsoorten in totaal 5 op korte termijn.
- **Portaal ontwikkelingen**: Webfunctionaliteit inzien & intrekken machtigingen Digipoort, DigiD assessment op B2 intrekkingsportaal
- **Voorzieningen en infra.**: Technische en functionele migratie Digipoort, diverse rapportages en beheerportalen herbouw
- **Regelgevingen en taxonomieën**: CSRD - Corporate Sustainability Reporting Directive, CbCR - Country-by-Country Reporting, …

De middellange termijn ontwikkelingen (Q1 – Q4 2026)

- **Regelgevingen en taxonomieën**: CSRD – MKB, eIDAS, WTMO, …
- **Voorzieningen en infra.**: IPv6 ondersteuning, Inloggen via Mijn DUO Zakelijk, …
- **Berichtsoorten en koppelvlakken**: DAC8 - EU Richtlijn Cryptobezittingen, *(continuering van korte termijn ontwikkelingen)*
- **Koppelvlakstandaarden**: Implementeren ebMS3/AS4, implementeren REST-API Digikoppeling voor Closed Data G2G uitwisseling

De lange termijn ontwikkelingen (2027 – 2030)

- **Berichtsoorten toevoegingen en aanpassingen**: nieuwe berichtsoorten
- **Regelgevingen en taxonomieën**: ESAP – European Single Access Point, SWIFT / G-rekeningen, *(continuering van middellange termijn ontwikkelingen)*
- **Machtigingen**: Doorontwikkeling B2 Machtigingsvoorziening, optimaliseren machtigingen, opheffen schijn van vertegenwoordiging, …
- **Afsprakenstelsel**: SBR Taxonomie ontwikkeling en releases, …
- **Koppelvlakstandaarden**: *(continuering van middellange termijn ontwikkelingen)*

Bron: 20250401 - Roadmap Domein GU - verwacht en gepland werk v17.xlsx

Colofon en disclaimer

Colofon

Zoals verzocht door Logius heeft SIG een Software Risk Assessment op Digipoort uitgevoerd tussen 24 februari – 8 april 2025. Formele opdrachtgever van deze opdracht is de directie van Logius. Tijdens deze opdracht vertegenwoordigde de directie Logius als klant naar SIG.

Documentstatus

Alleen het eindrapport is een formeel product van deze opdracht die de conclusies en aanbevelingen van SIG bevat. Alle andere rapporten vertegenwoordigen tussenproducten van de opdracht.

Disclaimer

SIG heeft dit rapport opgesteld op basis van informatie beschikbaar gesteld door Logius. Deze informatie is door SIG beoordeeld als zijnde geschikt voor het doel, de scope en diepgang van de opdracht. Hoewel SIG deze informatie op best-effort basis heeft gevalideerd, kan SIG de nauwkeurigheid, compleetheid, actualiteit en rechtmatigheid niet garanderen.

Amsterdam, 8 april 2025


Software
Improvement
Group .com
GETTING SOFTWARE RIGHT FOR A HEALTHIER DIGITAL WORLD