

Kiesraad - Beoordeling startfase Abacus **Eindpresentatie voor de Kiesraad**



SIG team

Context en managementsamenvatting

Bijlagen:

- Gebruikte modellen
- Hoofdbevindingen
- Conclusies en advies

Achtergrond

- De Kiesraad is een onafhankelijk bestuursorgaan in Nederland dat staat voor een eerlijk, transparant en controleerbaar verkiezingsproces.
- Het speelt een centrale rol bij het organiseren en controleren van verkiezingen en treedt op als centraal stembureau voor nationale en Europese verkiezingen.
- Daarnaast adviseert de Kiesraad de overheid over kiesrecht en verkiezingswetgeving en beheert het een uitgebreide databank met historische verkiezingsgegevens. Het doel van de Kiesraad is het bevorderen van een betrouwbaar en goed functionerend democratisch proces.

Context

- De applicatie Abacus is de beoogde opvolger van de module OSV-U en heeft als doel het invoeren en optellen van verkiezingsuitslagen en het berekenen van de zetelverdeling.
- De software wordt in huis bij de Kiesraad gebouwd en is in november 2024 in de startfase van de ontwikkeling. Hiervoor is een nieuw eigen team samengesteld bestaande uit zowel interne als externe specialisten. Abacus zal na oplevering in eigen beheer blijven van de Kiesraad.
- Voor de ontwikkeling hanteert de Kiesraad een eigen softwareontwikkelingsfilosofie, waarbij de software volledig open source ontwikkeld wordt en beschikbaar wordt gesteld op Github.
- Hiernaast beschikt het project over een interne repo zonder broncode, maar met enkele documenten voor het projectmanagement.

Analyseer het huidige ontwikkel- en voortbrengingsproces van Abacus en geef advies over verbeteringen die kunnen worden geïmplementeerd om dit proces te optimaliseren.

Deelvragen:

1. Geef inzicht in de meest relevante technische kwaliteitsaspecten, te weten onderhoudbaarheid en architectuurkwaliteit.
2. Geef inzicht in de volwassenheid van het ontwikkelproces, inclusief testdekking en gebruikte tooling;
3. Geef inzicht in de eventuele risico's die uit het bovenstaande voortvloeien;
4. Geef advies over risico mitigerende maatregelen, met inachtneming van prioriteit en benodigde ontwikkelingsinspanning.

Het voorliggende rapport is gebaseerd op de volgende artefacten:

- Broncode ontvangen in november 2024
- Planning ontvangen in februari 2025
- Gegevens in Sigrid uit februari 2025

Halverwege fase 1 is Abacus nog een klein systeem gebouwd volgens moderne best practices

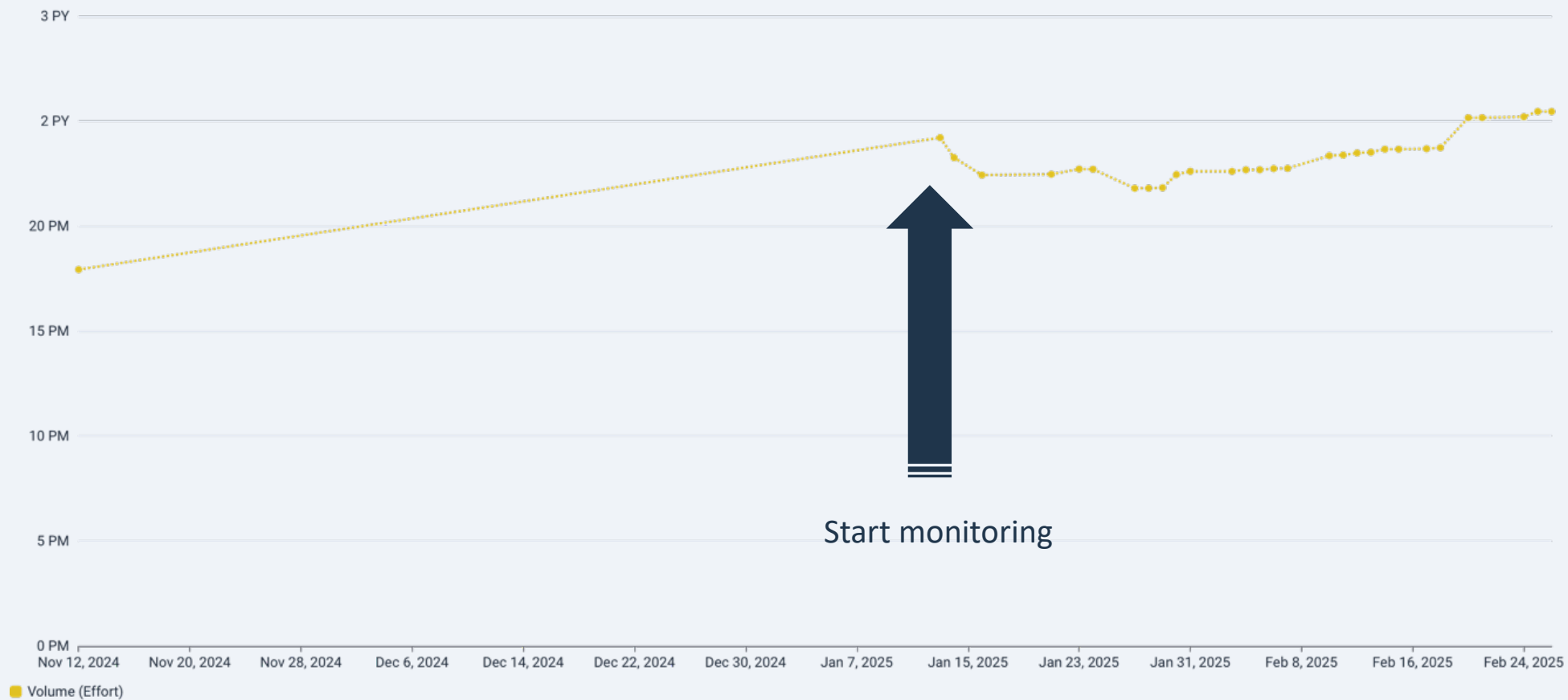
SIG heeft in opdracht van de Kiesraad een onderzoek gedaan naar het huidige ontwikkel- en voortbrengingsproces van Abacus, de applicatie wordt *in-house* ontwikkeld om het invoeren en optellen van verkiezingsuitslagen en het berekenen van de zetelverdeling in Nederland te ondersteunen. Na afronding van Abacus fase 1 kunnen de gemeenteraadsverkiezingen worden ondersteund. Na afronding fase 2 van de Abacus ontwikkeling kunnen de overige verkiezingstypen worden ondersteund.

Het SIG onderzoek heeft de volgende uitkomsten:

- Abacus is (nog) een klein systeem met een boven marktgemiddelde onderhoudbaarheid (★★★★☆), Abacus is geschreven in de moderne programmeertalen Rust en Typescript
- Abacus scoort boven marktgemiddeld op Architecture Quality (★★★★☆)
- Abacus wordt open-source ontwikkeld, het ontwikkelteam maakt gebruik van een modern proces voor softwareontwikkeling
- De momenteel gebouwde functionaliteit omvat ± 55% van de te bouwen functionaliteit in fase 1. Een realisatie van fase 1 binnen tijd (afronding eind juli 2025) en budget lijkt mogelijk, echter een deel van de '*should/could have*s'* zullen niet worden gerealiseerd
 - Om een vergelijkbare uitspraak voor de overige (in totaal 10) verkiezingstypes (in fase 2) te doen is extra onderzoek noodzakelijk naar de gelijkenissen en de verschillen tussen verkiezingstypes en de impact op de te ontwikkelen software hiervan.

*) Eisen en wensen welke niet strict noodzakelijk zijn voor de werking, maar wel gewenst

Ontwikkeling van de omvang van Abacus in de afgelopen maanden (uit Sigrid)



Verbeter unit metrieken, modulariteit architectuur en scherp richtlijnen voor ontwikkeling aan

Verbeteringen in de Abacus applicatie en het gebruikte ontwikkelproces zijn mogelijk op volgende onderdelen:

- **[Onderhoudbaarheid]** Beperking van de lengte en complexiteit van de units in de broncode
- **[Architectuur]** Een betere bewaking, m.n. de modulariteit binnen de applicatie en het volgen van de ontworpen architectuur
 - De componenten hebben relatief veel communicerende code en wijzigingen vinden vaak in meerdere componenten tegelijk plaats
- **[Proces]** Er is sprake van relatief grote tickets/stories in Git, die soms niet binnen één dag zijn uit te voeren
- **[Proces]** Project-overstijgende richtlijnen voor bouwen en onderhouden van broncode, met name in de frontend zijn vaak generiek, waardoor toepassing tijdens ontwikkeling niet altijd lukt en vooral een verdienste is van de individuele kwaliteiten binnen het team

Gegeven de huidige omvang en fase van ontwikkeling hebben de bovenstaande bevindingen momenteel beperkte gevolgen, echter op langere termijn (richting de beheerfase) kan de impact voelbaar worden:

- Verlaagde productiviteit van het ontwikkelteam, doordat het meer tijd kost om minder goed onderhoudbare code uit te breiden of te repareren
- Een verminderde overdraagbaarheid (naar nieuwe ontwikkelaars/beheer) doordat de broncode niet conform duidelijke richtlijnen (architectuur en code gerelateerd) is gebouwd en/of gevolgde richtlijnen niet expliciet zijn beschreven
- Een hogere kans op fouten in de software, doordat code complex is, en zich daardoor minder goed laat testen

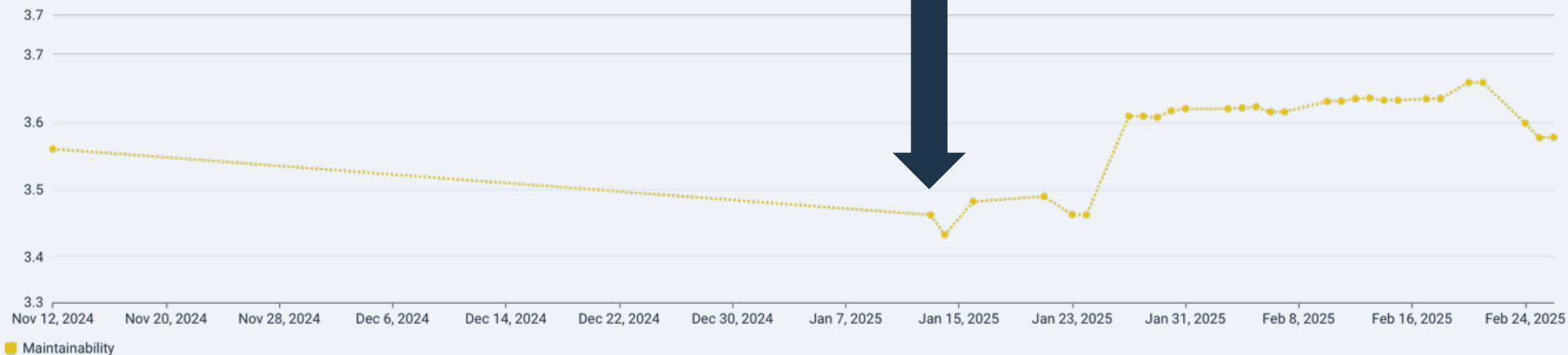
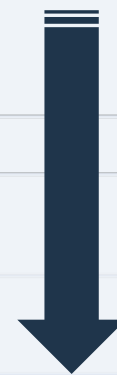
Focus op meetbare kwaliteit en het definiëren en volgen van project-overstijgende richtlijnen

SIG adviseert de volgende verbeteringen op korte en/of middellange termijn in te richten binnen het project. Deze verbeteringen hebben impact op de kwaliteit, de (ontwikkel)methodiek, architectuur en planning van de Abacus ontwikkeling.

1. **[Kwaliteit - onderhoudbaarheid]** Maak de onderhoudbaarheid onderdeel van de 'Definition of Done' met het doel om ruim 4 sterren (SIG, of equivalent) te scoren, rapporteer periodiek of laat extern toetsen, begin met extra aandacht voor de unit-metrieken.
2. **[Methodiek/Architectuur]** Actualiseer de architectuurbeschrijvingen en synchroniseer de beschreven en gebouwde architectuur, zodat er geen afwijkingen meer zijn en beide synchroon blijven. Stuur op een modulaire architectuur waardoor de componenten in isolatie kunnen worden aangepast en getest.
3. **[Methodiek]** Stel project-overstijgende richtlijnen voor de code op welke getoetst kunnen worden in elk ticket en/of elke sprint (Definition of Done /Peer reviews).
4. **[Methodiek]** Maak aanpassingen in de werkwijze, zodat er ook kleinere tickets worden gepland en uitgevoerd
5. **[Planning]** Actualiseer periodiek de schattingen en extrapolaties (omvang en mate van afronding) gedurende fase 1 zodat de einddatum steeds nauwkeuriger kan worden vastgesteld.
6. **[Planning]** Werk de scope van fase 2 verder uit om een meer zekerheid te verkrijgen over de omvang van fase 2 (is fase 2 daadwerkelijk kleiner dan fase 1 en klein genoeg om binnen gestelde tijd en budget te realiseren?).

Ontwikkeling van de onderhoudbaarheid van Abacus in de afgelopen maanden (uit Sigrid)

Start monitoring





.....
GETTING SOFTWARE RIGHT FOR A HEALTHIER DIGITAL WORLD

Bijlagen

- **Gebruikte modellen**
- Bevindingen
- Conclusies advies

We meten volgens de ISO 25010 standaard voor Software Product Kwaliteit



■ = niet gemeten door SIG

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**.

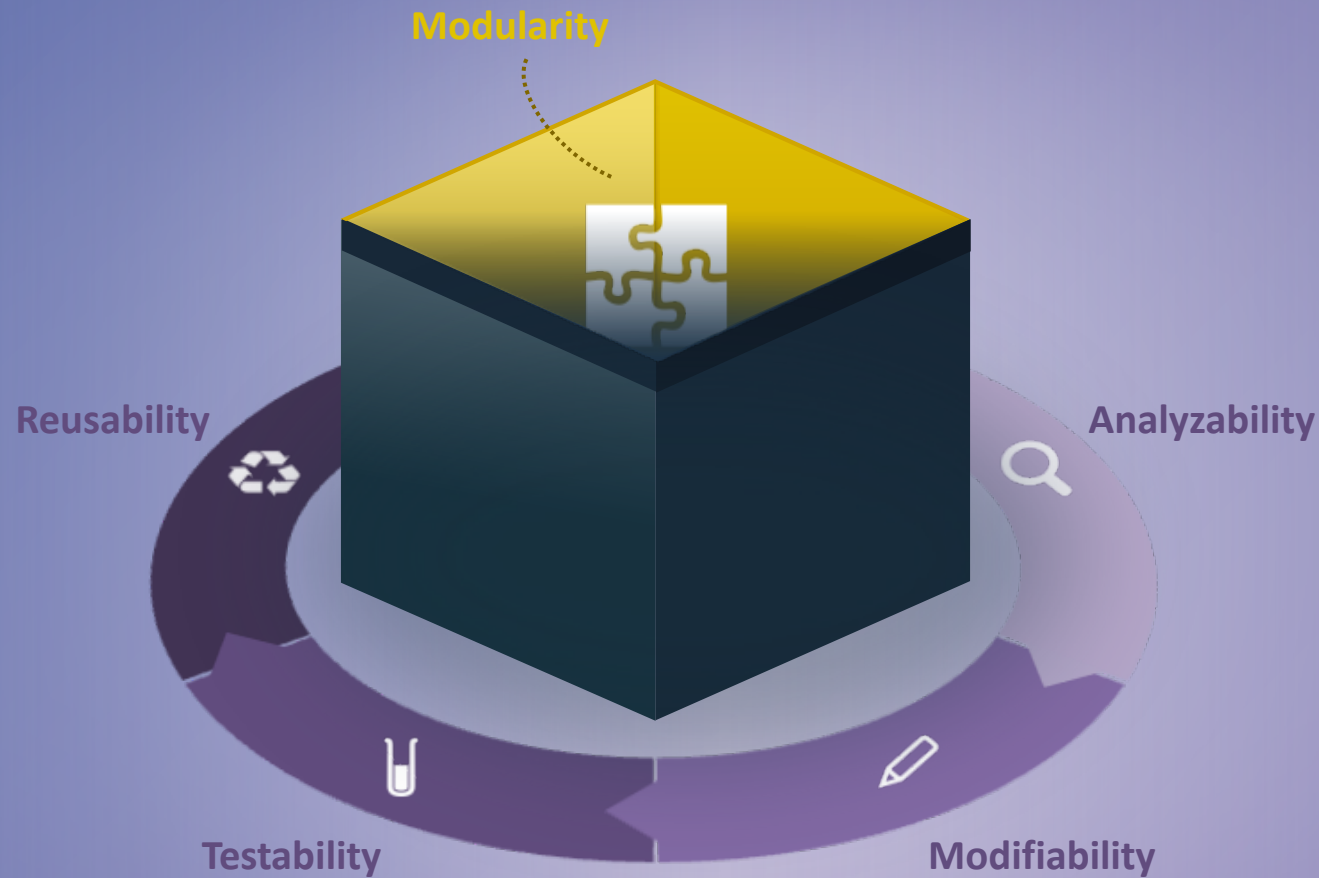
In aanvulling op deze modellen biedt SIG ook **vele andere diensten** die helpen om inzicht te krijgen in de risico's waaraan uw software (en bedrijf) worden blootgesteld.

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



Maintainability ISO 25010

Analyzability
Modifiability
Testability
Modularity
Reusability



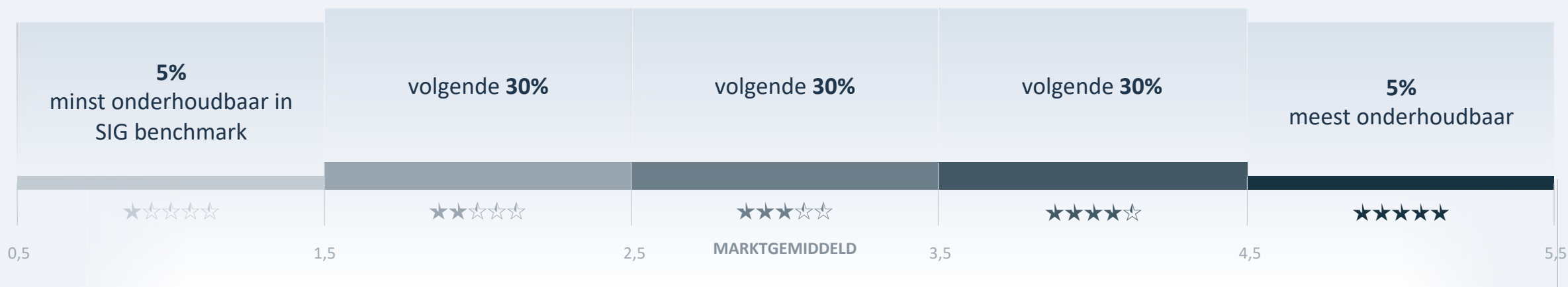
The ISO 25010 standard refers to **maintainability** as:
“The degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.”

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.

The SIG/TÜViT model maps the sub-characteristics to the system properties

 		 Volume	 Duplication	 Unit size	 Unit complexity	 Unit interfacing	 Module coupling	 Component entanglement	 Component independence
	Analyzability	■	■	■					
	Modifiability		■		■		■		
	Testability	■			■				■
	Modularity						■	■	■
	Reusability			■		■			

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

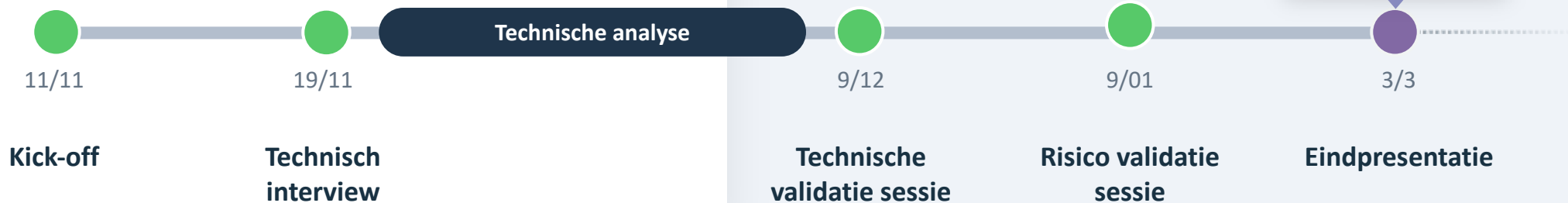


Onderhoudbaarheid wordt vastgesteld door geautomatiseerde broncode-analyse.

- 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.

Fase 1: Initiatie en analyse

Fase 2: Advies

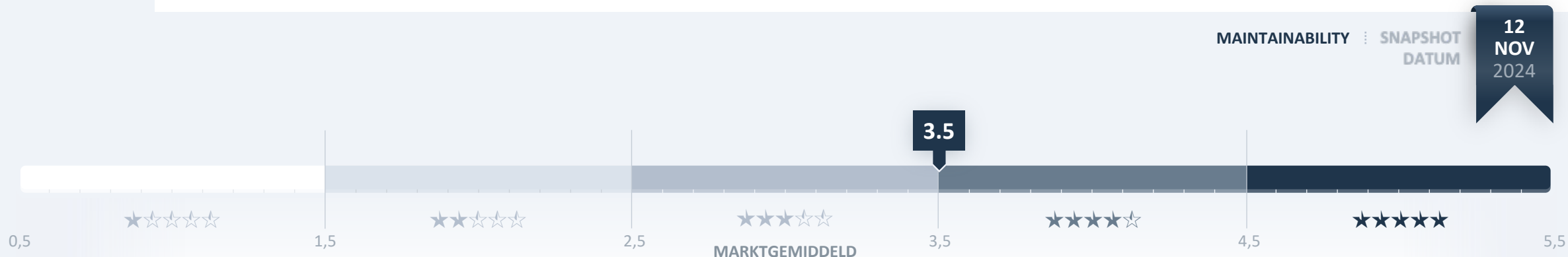


.... LEGENDA: ● Completed session ● Current session ● Future session

Bijlagen

- Gebruikte modellen
- **Bevindingen**
- Conclusies en advies

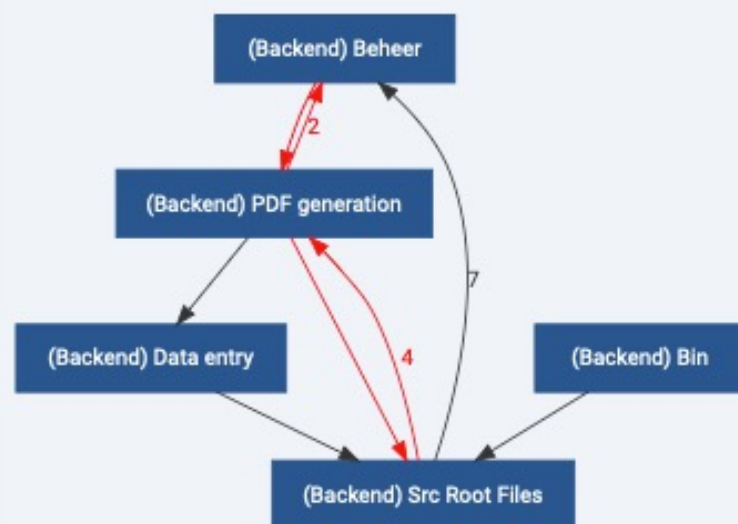
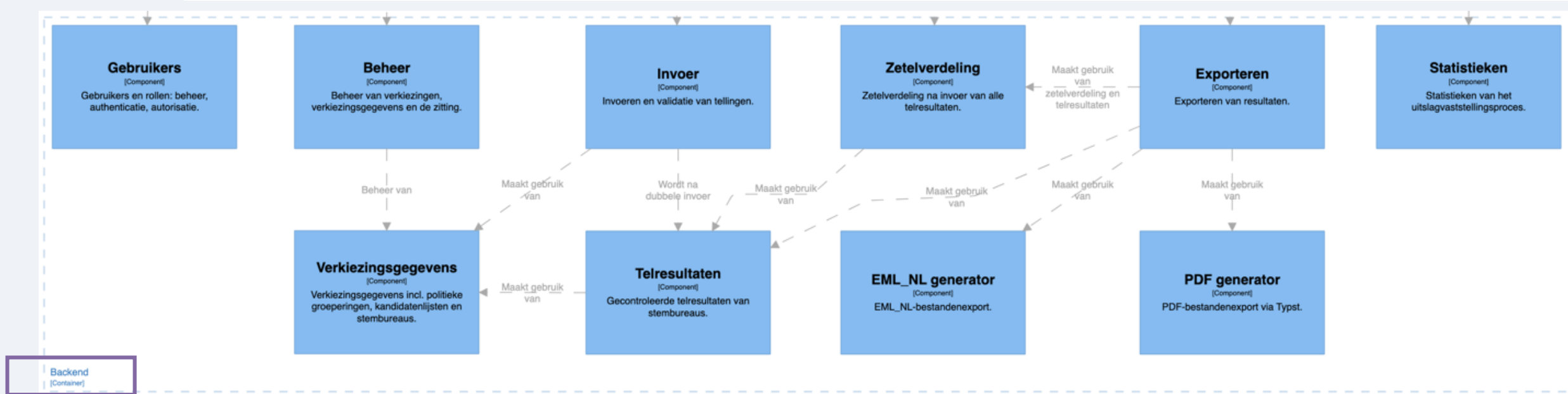
De onderhoudbaarheid van Abacus is (net) boven marktgemiddeld ★★★★★☆ (3.5)



- Abacus is een klein system met een volume van ±18 persoonsmaanden
- Op *duplication* en *module coupling* scoort Abacus boven marktgemiddeld
- Op de meeste unit metrieken scoort Abacus marktgemiddeld, *unit complexity* heeft aandacht nodig:
 - Met name de frontend bevat code met veel keuzepaden, dit wijst op te verminderde aandacht voor het single responsibility principe
- Unit size scoort onder marktgemiddeld:
 - 31% van de code is onderdeel van lange units (>30 LoC)
 - Het opsplitsen van lange units kan ook de unit complexity score verbeteren
- De componenten zijn verknoopt en hebben een hoge afhankelijkheid; de architectuur in de broncode wijkt af van de ontworpen architectuur (zie volgende slide)

Volume	★★★★★	5.3
Duplication	★★★★☆	4.5
Unit size	★★★☆☆	2.3
Unit complexity	★★★★☆	2.8
Unit interfacing	★★★★☆	3.3
Module coupling	★★★★☆	4.3
Component independence	★★★★☆	2.9
Component entanglement	★★★★☆	2.6

Mapping van de code componenten op de architectuur is (nog) niet duidelijk*



- De mapping van de ontworpen architectuur op de gemeten architectuur is niet eenduidig
- Het is wenselijk om een duidelijke mapping te hebben van de geïmplementeerde architectuur op de ontworpen architectuur
- Update de architectuur diagrammen wanneer er afgeweken moet worden

* Er is enkel een software architectuur voor de backend aangetroffen

Abacus scoort boven het marktgemiddelde op Architecture Quality (★★★★☆)

- De componenten zijn intern samenhangend en losjes gekoppeld. Dit duidt op een redelijk modulaire opbouw. Verder is er geen sterke data koppeling gevonden tussen componenten, wat wijst op een scheiding van verantwoordelijkheden.
- ⚠ De ontworpen architectuur komt niet helemaal overeen met de gemeten architectuur.
- ⚠ Een deel van de componenten heeft veel communicerende code. Dit is geen reden tot zorg in de huidige implementatie. Bij groei van het systeem is het belangrijk om hier goed oog voor te hebben.
- Alle componenten worden actief onderhouden door verschillende auteurs. Dit duidt op een ontwikkelcultuur waarin kennis en verantwoordelijkheden worden gedeeld.
- ⚠ Wijzigingen vinden vaak in meerdere componenten tegelijk plaats. Dit kan wijzen op sterke onderlinge afhankelijkheden tussen componenten of op *crosscutting* concerns die meerdere delen van het systeem beïnvloeden. De verklaring lijkt mede te liggen in de omvang van de tickets & 'dat er tijdens het oplossen van tickets regelmatig aanvullende verbeteringen worden meegenomen.'

Code breakdown	☆☆☆☆☆ (N/A)
Component coupling	★★★★☆ (4.3)
Component cohesion	★★★★☆ (3.7)
Code reuse	★★★★★ (5.5)
Communication centralization	★★★★☆ (3.6)
Data coupling	★★★★★ (4.5)
Bounded evolution	★★★☆☆ (3.2)
Knowledge distribution	★★★★☆ (4.2)
Component freshness	★★★★★ (5.2)

Het ontwikkelproces is robuust, controleer de componenten en de koppeling

- Het ontwikkelproces van de Kiesraad is weloverwogen, er is nog ruimte voor verbetering:
 - Begin eerder met deployment, ook als dit alleen kan (door nabootsing) in een productie-like omgeving
 - Het Github code security tooling wordt gebruikt, ontwikkel ook secure coding kennis in het team
 - Het team volgt een filosofie waarin veel wordt samengewerkt, het is verstandig om dit te blijven borgen. Deze filosofie zou uitgebreid kunnen worden door:
 - Te controleren op koppeling(en) (tussen componenten) bij peer reviews
 - Tickets te introduceren die in minder dan een dag kunnen worden afgerond, dit kan voorkomen dat een ontwikkelaar 'geblokkeerd is' op momenten dat er weinig mensen aanwezig zijn
- De in software gemeten architectuur komt niet overeen met de ontworpen architectuur, het is verstandig om een verbeterplan op te stellen. Het introduceren van een controle op koppeling bij peer reviews kan hier een eerste stap in zijn. Update de ontworpen architectuur wanneer de geïmplementeerde architectuur evolueert.
- Package management en de gemeten Open Source Health is goed, hier zijn momenteel *geen* verbeterpunten. Hou dit vast voor zover mogelijk en realistisch:
 - De transitieve afhankelijkheden laten activiteitsrisico's zien, SIG raadt aan om hier bewust van te zijn maar nog geen directe actie op te ondernemen

De OSH score is boven marktgemiddeld (inclusief transitieve dependencies)

Kiesraad Portfolio

>

Abacus

System Overview

Maintainability

Architecture

Open Source Health

Findings

Code Explorer

Nov 20, 2024 - Dec 19, 2024

Kiesraad Portfolio > Abacus > Open Source Health

Open Source Health

★★★★☆ (3.9) = 0.00

Vulnerability★★★★★ (5.0) = 0.00

License use★★★★☆ (4.2) = 0.00

Freshness★☆☆☆☆ (1.4) = 0.00

Management★★★★★ (5.5) = 0.00

Activity★☆☆☆☆ (1.4) = 0.00

Model version: 2024

OSH benchmark is a new feature that enables you to compare your open source processes against the market. Click [here](#) to learn more about this new feature.

Libraries

428

Latest scan

Dec 12, 2024, 3:26 PM

Status

Added0

Upgraded0

Downgraded0

Libraries with vulnerability risk

Critical risk0 = 0

High risk0 = 0

Medium risk1 = 0

Trend Lines

Expand

Library name

Search by library name

Export Software Bill of Materials

Type	Name	Dependency type	Version	Library freshness	Status	License	Risk ↑							
Cargo	backend	Transitive	0.1.0	-	Unchanged	MIT								
Cargo	crunchy	Transitive	0.2.2	-	Unchanged	MIT								
NPM	js-tokens	Transitive	4.0.0	66 M	Unchanged	MIT								

// SOFTWARE IMPROVEMENT GROUP CONFIDENTIAL

22

Broncode voor zetelverdeling niet in de upload aanwezig

Artikel		Invulling door het Abacus ontwikkelteam
de programmatuur bevat de functionaliteiten die overeenkomstig de specificatie, bedoeld in artikel P 1, tweede lid, van het Kiesbesluit nodig zijn voor de berekening van de uitslag van de verkiezingen en de zetelverdeling;		Op basis van de beschreven use-cases zal de programmatuur de benodigde functionaliteiten bevatten zoals in het artikel beschreven
de programmatuur, waaronder de broncode, is gestructureerd opgebouwd, zodanig dat modulaire aanpassingen mogelijk zijn;		Modulaire aanpassingen worden nu nog beperkt uitgevoerd, dit kan het gevolg zijn van de huidige (start)fase van Abacus, maar ook het gevolg zijn van het gekozen ontwerp (architectuur)
de kritische functies voor de berekening van de uitslag van de verkiezingen en de zetelverdeling zijn in de programmatuur herkenbaar en van elkaar gescheiden;		Volgens het (architectuur)model is dit onderdeel van een aparte component genaamd 'Zetelverdeling', deze is niet in de ontvangen broncode aanwezig. → Ondertussen is deze code in ontwikkeling en aangetroffen in de codebase
de programmatuur is, zonder dat hiervoor aanpassingen nodig zijn, te gebruiken voor verschillende soorten verkiezingen;	!	Momenteel is dit niet mogelijk, in fase 1 worden enkel gemeenteraadsverkiezingen ondersteund, de uitbreiding met andere verkiezingstypen is in scope voor fase 2

Abacus wordt open-source ontwikkeld, foutief gebruik wordt door het ontwerp voorkomen

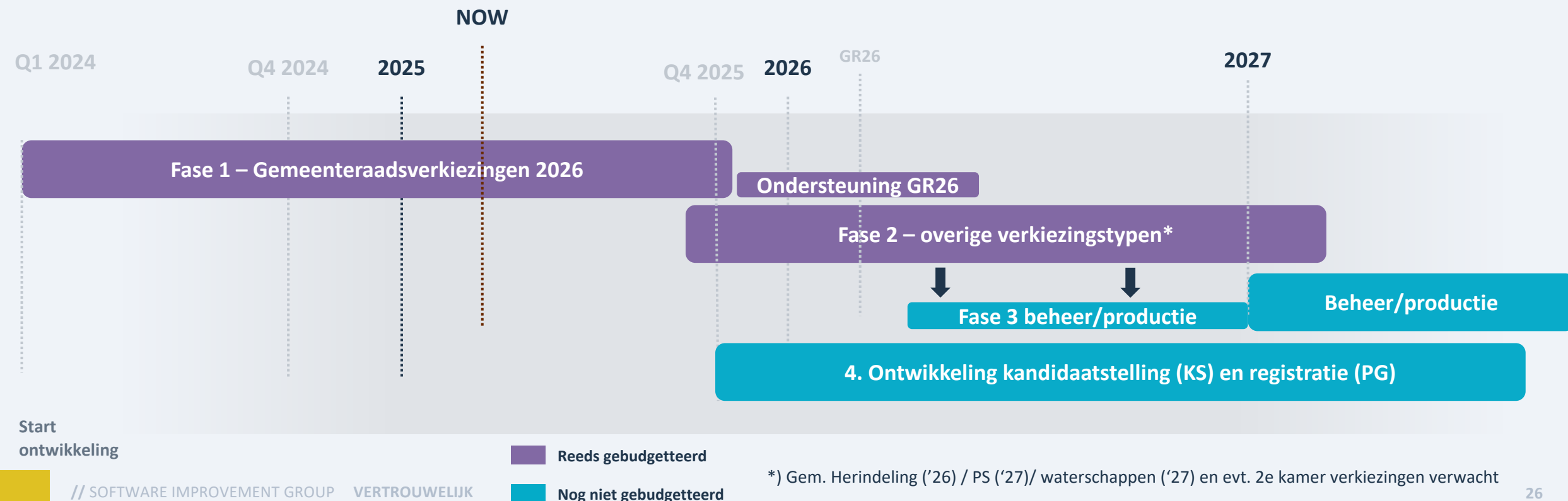
Artikel		Invulling
toevallig of opzettelijk foutief gebruik van de programmatuur wordt, voor zover redelijkerwijs technisch mogelijk is, door het ontwerp voorkomen;		Er is sprake van ondersteuning van het 4-ogen principe, foutsituaties worden herkend en er wordt logging gedaan van de acties van gebruikers → Meer logging is voorzien door het team
de programmatuur ondersteunt voor de vermelding van de aanduidingen van de politieke groeperingen en de namen van de kandidaten in ieder geval de diakritische tekens van de tekenset die op grond van artikel 3, eerste lid, van het Besluit basisregistratie personen voor de basisregistratie personen is vastgesteld;	?	Het is onduidelijk of dit mogelijk is in de ontwikkelde software, of dat dit een eis is, die in scope is voor dit gedeelte van de verkiezingssoftware → Team geeft aan testcases voor Diakrieten in te regelen
de programmatuur wordt als open source ontwikkeld en maakt gebruik van open standaarden. Indien dit aantoonbaar niet mogelijk is wordt technologie toegepast waarvan de doeltreffendheid in de praktijk is aangetoond en die direct toepasbaar is. Voor verkiezingsgegevens zoals kandidatenlijsten en zetelverdeling wordt de EML_NL standaard toegepast;		De broncode van de verschillende versies van Abacus zijn Open Source beschikbaar en kunnen mede ontwikkeld worden door derden. Voor gegevens zoals beschreven in dit artikel wordt EML_NL gebruikt

IP ligt bij Kiesraad, authenticiteit programmatuur/verkiezingsgegevens niet vast te stellen

Artikel		Invulling
de standaard programmatuur waarvan gebruik wordt gemaakt is vrij verkrijgbaar;		De software is te downloaden uit GitHub (https://github.com/kiesraad/abacus)
het intellectueel eigendom van de maatwerkprogrammatuur berust bij een centraal stembureau;		Het intellectueel eigendom (IP) van de ontwikkelde software ligt bij de Kiesraad
de programmatuur is geschreven in een programmeertaal, waarvoor een door een actieve gemeenschap onderhouden open source <i>compiler</i> , onderscheidenlijk <i>interpreter</i> beschikbaar is;		Er wordt gebruik gemaakt van Typescript (React) in de frontend en Rust in de backend. Deze moderne technologieën voldoen aan deze eis
de programmatuur wordt ontwikkeld voor verschillende besturingssystemen, waaronder in ieder geval een open source besturingssysteem;		Abacus wordt actief getest op Windows (>10), Linux en MacOS
het is mogelijk de authenticiteit van de programmatuur vast te stellen;	?	Niet aangetroffen
bij het inlezen van verkiezingsgegevens in de programmatuur wordt de authenticiteit van de gegevens vastgesteld, bij voorkeur door middel van een gekwalificeerde elektronische handtekening.	?	Niet aangetroffen

Abacus wordt ontwikkeld in drie fasen, naast de ontwikkeling van de modules KS en PG

- 1 Fase 1: ontwikkeling Abacus voor Gemeenteraadsverkiezingen (gericht op de 2026 verkiezingen)
- 2 Fase 2: doorontwikkeling Abacus voor de overige verkiezingstypen (9x), inclusief beheer bestaande software
- 3 Fase 3: Abacus in productie (beheer), incl. standaard doorontwikkeling
- 4 Ontwikkeling kandidaatstelling en registratie (overige modules), middels een aanbesteding



Totaal budget voor Abacus fase 1 en 2 bedraagt 6Mio EURO beschikbaar in 2024 t/m 2026

Uitgangspunten voor de ontwikkeling van Abacus:

- Initiële ontwikkelperiode is 3 jaar (2024 t/m 2026)
 - Budget voor deze periode is 6Mio (1,8Mio voor inhuur per jaar, 200k overige kosten per jaar)

Actuals en prognose

- 2024 actuals/prognose: 1,5M (onder budget), 'ramping up' het ontwikkelteam, ontwikkelteam is volop productief vanaf juli 2024
- 2025 prognose: 1,9M
- 2026 prognose: 1,8M
 - O.b.v. het actuele budget is enige uitloop in 2027 mogelijk

Momenteel is ±55% van Must, Should en Could haves gebouwd

- Van de onderwerpen in scope voor fase 1 (GR26) is momenteel ±55% gebouwd, met de name de ‘*must haves*’. Het grootste onderwerp ‘Uitslagbepaling GSB’ (goed voor 16% van de Fase 1 scope) is ±66% gereed
- De totale scope is toegenomen met ±20 %, het onderwerp ‘Uitslagbepaling’ is opgesplitst, de omvang is toegenomen

Onderwerpen	Zwaarte' onderwerp	Omvang per onderwerp	Afgerond als % van geheel	% Afgerond per onderwerp
Gebruikers en rollen	2	4%	3%	90%
Inrichting van de applicatie	9	16%	10%	59%
Invoeren van uitslagen GSB	5	9%	7%	75%
Uitslagbepaling GSB	9	16%	11%	66%
Uitslagbepaling CSB	8	15%	11%	74%
Ondersteunende functies	3	5%	4%	77%
Versie 1.0: zeer gewenst (should have)	4	7%	2%	28%
Versie 1.0: gewenst (could have)	15	27%	7%	27%
Totaal	55	100%	55%	55%
Opgetelde complexiteit			% Afgerond van het totaal	

- Op basis van de ontwikkeltijd (±7 maanden) met het complete team (in periode Aug24 – Feb25) voor 55% van de functionaliteit zijn wordt het lastig om de overige 45% af te ronden in 4-5 maanden (afronding eind juli 2025), hierdoor zal een aantal ‘could haves’ en ‘should haves’ niet zijn afgerond.
- De totale omvang van Abacus fase 1 zal ongeveer verdubbelen naar ± 4 Persoonsjaren

De opgeleverde Abacus software wordt regelmatig op meerdere niveau's getest



- **Norm:** Volwassen software is aantoonbaar en herhaalbaar getest op vijf niveaus (zie figuur).
- **Bevindingen:**
 - De opgeleverde functionaliteit worden door het team regelmatig op meerdere niveau's geïnstalleerd en (gedeeltelijk) getest
 - Niet-functionele testen kunnen worden uitgebreid om betere dekking te verkrijgen
 - Testen met gemeenten in 25Q1

*) de Abacus software wordt uiteindelijk in isolatie (air-gapped) uitgerold

Er zijn geen beperkingen in de software om de overige verkiezingstypen te ondersteunen

- *Q: Hoe flexibel is de architectuur van het systeem en in hoeverre zijn best practices toegepast zodanig dat veranderende businesswensen kunnen worden ondersteund?* → Mogelijke veranderingen (volgens het ontwikkelteam):
 - Ondersteuning van een districtenstelsel
- Bevindingen:
 - Abacus is een systeem waarin het (leidende) papieren verkiezingsproces wordt ondersteund, een proces waarbij gebruik wordt gemaakt van districten is nog niet beschreven
 - De beschreven architectuur is onafhankelijk van het type verkiezing dat wordt uitgevoerd. De beschreven architectuur en de vastgestelde softwarearchitectuur zijn (nog) niet in overeenstemming
 - Er is geen beschrijving van classes/objecten/entiteiten aangetroffen (domainmodel), de verschillende verkiezingstypen en stembureaus zijn beschreven
 - Verkiezingstype (*ElectionCategory*) is een onderscheidende parameter in de codebase, deze wordt nog beperkt gebruikt
 - Er zijn verschillende typen stembureaus actief in de verschillende verkiezingen, elke type stembureau heeft een andere rol
 - Een stembureau heeft een type, welke in de software kan worden gebruikt om verschillende functionaliteit te triggeren
 - **Functioneel nog te veel onduidelijk om een inschatting te geven van de haalbaarheid van een districtenstelsel binnen Abacus**

Bijlagen

- Gebruikte modellen
- Bevindingen
- **Conclusies en advies**

Onderhoudbaarheid boven marktgemiddeld, verbetering mogelijk in units en de architectuur

Abacus is momenteel nog een klein systeem geschreven in Rust en Typescript, de onderhoudbaarheid van Abacus is (net) boven marktgemiddeld ★★★★★☆ (3.5). De technische herbouwwaarde bedraagt ±18 PM.

SIG adviseert 4★ als het minimale streefniveau voor de onderhoudbaarheid van nieuwe systemen

- De unit metrieken zijn *slechts* markt-gemiddeld, verbeteringen zijn mogelijk en kunnen bijvoorbeeld worden meegenomen in de DoD/peer reviews na afronding van ticket of sprint
 - De frontend bevat relatief veel code met veel keuzepaden (hoge complexiteit)
 - Unit size scoort onder marktgemiddeld, het opsplitsen van lange units kan tevens de unit complexity score verbeteren
- Er is sprake van een verknoping van de componenten en de componenten zijn niet altijd losjes gekoppeld
- Er is relatief CSS broncode in het systeem, doordat bewust geen CSS frameworks worden gebruikt. Toets regelmatig of het CSS-broncode volume beperkt blijft

Abacus scoort boven het marktgemiddelde op Architecture Quality (4.3★)

Abacus scoort boven het marktgemiddelde op Architecture Quality (★★★★☆), volgende verbeteringen zijn mogelijk

- Een deel van de componenten heeft veel communicerende code
- Wijzigingen vinden vaak in meerdere componenten tegelijk plaats
- De componenten zijn (marktgemiddeld) verknoopt en er zijn cyclische dependencies tussen componenten
- De componenten (en de interacties tussen componenten) wijken af van de ontworpen architectuur

Aan de wettelijke eisen voor verkiezingssoftware wordt grotendeels voldaan, een aantal eisen is (nog) niet te toetsen

- Verbetering zijn nog mogelijk op de het vlak van modulariteit van de applicatie

Op basis van de nu reeds afgeronde functionaliteit ($\pm 44\%$ van Gemeenteraad-scope) en de daarvoor gebruikte inspanning door het team lijkt de afronding van de complete scope van fase 1 haalbaar binnen tijd (afronding voor november 2025) en budget.

Hierdoor blijft er tijd en budget beschikbaar voor fase 2, echter;

- Het is nog niet vast te stellen of de scope van fase 2 daadwerkelijk kleiner is dan de scope van fase 1.
- Hierdoor kan niet met voldoende zekerheid worden gezegd dat het overgebleven budget en de tijd voldoende zijn om de volledige scope van fase 2 binnen budget en de gestelde mijlpalen af te ronden.

Het bouwproces van Abacus is 'weloverwogen, beperkte verbeteringen zijn mogelijk

Het Abacus ontwikkelproces gebruikt een groot aantal moderne ontwikkel best-practices en scoort met name goed op de onderdelen 'Programming, Documentation en Planning'. De volgende onderdelen scoren minder goed:

- Project-overstijgende richtlijnen voor het bouwen en onderhouden van de broncode, met name in de frontend zijn vaak generiek
- Er is sprake van relatief grote tickets/stories in Git en veel interactie binnen het team, waardoor het maken van de changes regelmatig langer dan een dag nodig heeft
- De beschreven architectuur is niet eenduidig terug te vinden in de ontwikkelde codebase; er ontbreken nog componenten, componenten hebben een andere naam en/of kunnen niet gemapped worden. Er bestaan verbindingen tussen componenten die niet in het ontwerp staan

Volgende verbeteringen mogelijk:

- Begin zo snel als mogelijk met deployment, ook als dit alleen kan in een (nagebootste) productie-like omgeving
- Maak toetsing van de architectuur en de koppeling tussen componenten onderdeel van de peer reviews, pas de ontworpen architectuur aan, indien noodzakelijk
- Introduceer ook kleinere tickets, die in minder dan 1 dag geïmplementeerd kunnen worden

Focus op meetbare kwaliteit en het volgen van project-overstijgende richtlijnen

SIG adviseert de volgende verbeteringen op korte en/of middellange termijn in te richten binnen het project. Deze verbeteringen hebben impact op de kwaliteit, de (ontwikkel)methodiek, architectuur en planning.

1. **[Kwaliteit - onderhoudbaarheid]** Maak de onderhoudbaarheid onderdeel van de DoD met het doel om ruim 4 sterren (SIG, of equivalent) te scoren, rapporteer periodiek of laat extern toetsen, begin met extra aandacht voor de unit-metrieken
2. **[Methodiek/Architectuur]** Actualiseer de architectuurbeschrijvingen en synchroniseer de beschreven en gebouwde architectuur, zodat er geen afwijkingen meer zijn en blijven. Stuur op een modulaire architectuur waardoor de componenten beter in isolatie kunnen worden aangepast en getest
3. **[Methodiek]** Stel projectoverstijgende richtlijnen voor de code op welk getoetst kunnen worden in elk ticket of sprint (DoD/Peer reviews)
4. **[Methodiek]** Maak aanpassingen in de werkwijze, zodat er ook kleinere tickets worden gepland en uitgevoerd
5. **[Planning]** Actualiseer regelmatig de schattingen en extrapolaties (omvang en mate van afronding) gedurende de rest van fase 1 zodat de einddatum steeds nauwkeuriger kan worden vastgesteld
6. **[Planning]** Werk de scope van fase 2 verder uit om een hogere zekerheid te verkrijgen over de omvang van fase 2 (daadwerkelijk kleiner dan fase 1 en klein genoeg om binnen gestelde tijd en budget te realiseren)



.....

GETTING SOFTWARE RIGHT FOR A HEALTHIER DIGITAL WORLD