UWV

IT STARTS WITH THE FACTS.

Presentatie resultaten Software Quality & Risk Assessment UWV RINA

15/10/21

metri
IT STARTS WITH THE FACTS

Inhoudsopgave

IT STARTS WITH THE FACTS.

uwv

metri

IT STARTS WITH THE FACTS

Introductie en Doelen

Introductie

- Voor de applicatie RINA wil UWV graag een gedetailleerd beeld krijgen van de issues in de applicatie die betrekking hebben op de Performance Efficiency, Security, Stabiliteit/continuïteit, Toegankelijkheid en Connectiviteit van de applicatie (0-meting).
- Hierbij wil UWV een Gap analyse om het verschil tussen de huidige situatie en de gewenste situatie te bepalen.
- Op basis van deze Gap analyse wenst UWV een voorstel tot prioritering inclusief een kostenraming om de verschillen op te lossen.
- Na de baseline meting wil UWV op 2 latere momenten de voortgang meten t.a.v. het dichten van de geconstateerde Gaps (1-meting en 2-meting)

- In deze presentatie worden de resultaten uit de 0-meting teruggekoppeld aan UWV.

Onderzoeksdoelen

De doelen van UWV voor dit onderzoek zijn:

- Het verkrijgen van onafhankelijk inzicht in de **technische kwaliteit** van de applicatie RINA o.b.v. alle gangbare internationale standaarden (ISO, CISQ, NIST, OWASP, etc.) en best practices, gemeten op systeem niveau (inclusief de communicatie tussen componenten, lagen en objecten).
- Het verkrijgen van inzicht in de kwaliteit van en risico's in de applicatie uitgedrukt in een score van 1 tot 4 op de 'health factors' **Robuustheid, Efficiency, Security, Wijzigbaarheid, Overdraagbaarheid en de totale kwaliteit (Total Quality Index)**.
- Het verkrijgen van inzicht in de **technische schuld** in de applicatie inclusief een indicatie wat aan technische schuld worden verwacht van een technologisch vergelijkbare applicatie die onlangs nieuw is opgeleverd.
- Het verkrijgen van **een concreet overzicht van de gevonden kritische fouten**, de locatie in de code waar deze zijn gevonden, waarom dit fouten zijn (inclusief referentie naar de betreffende standaard of best practice) en hoe deze fouten eventueel op te lossen zijn.
- Het verkrijgen van een **concreet verbeterplan** met daarin de fouten die op korte termijn tegen zo laag mogelijke kosten de hoogste kwaliteitsverbetering opleveren, inclusief een simulatie van de scores op de genoemde health factors als dit verbeterplan is uitgevoerd.

Wat meten we?

Over CAST en het ecosystem

CAST technologie levert **cruciale inzichten** met betrekking tot **software kwaliteit en risico's**, functionele en technische omvang en performance van applicatie development en beheerteams (zowel intern, outsourced of hybride) , gebaseerd op **geautomatiseerde source code analyse** ten opzichte van **alle gangbare standaarden en best practices** (ISO 25010, CISQ, CWE, MISRA, NIST, OMG, OWASP, PCI-DSS, STIG, etc.)

Resultaat: **Betere beslissingen, lagere risico's, hogere kwaliteit, meer management control en hogere team effectiviteit en efficiency.**

CAST BY THE NUMBERS

- Almost **$180M** investment in R&D
- **250+** customers worldwide
- **25+** years of software analytics experience measuring some of the most complex IT systems in the industry
- Traded on Euronext Paris
- Offices in US, Europe, India, China

Gartner COOL VENDOR

CAST Named "Cool Vendor" by Leading Analyst Firm

UBM

Editor's Choice Award: A Top-10 Company to Watch

Gartner

"CAST is **the de facto standard** for measuring quality and productivity of application services"

OMG — WE SET THE STANDARD

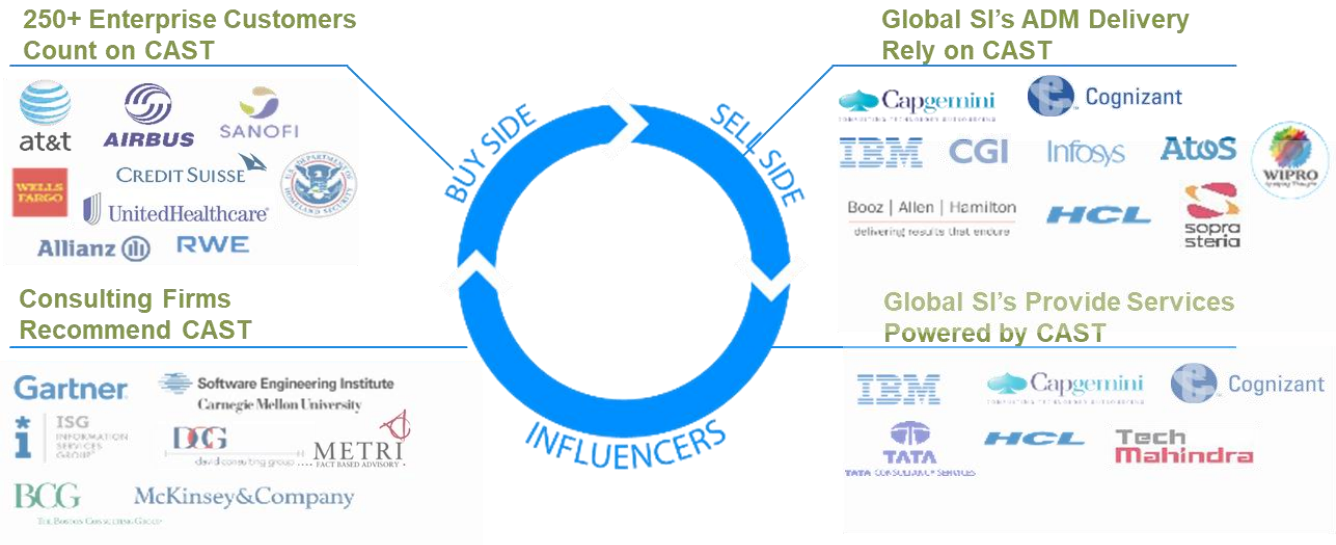
"CAST is **at the forefront** of standards adoption for **robustness, security, maintainability, and automated function points from code**"

IDC — Analyze the Future

"CAST is **the leader in the business IT space**"

OVUM

"CAST is the **leading technology** of its kind"



Management Dashboard? Voorbeeld!

CISQ

OMG

Software Engineering Institute

NIST National Institute of Standards and Technology U.S. Department of Commerce

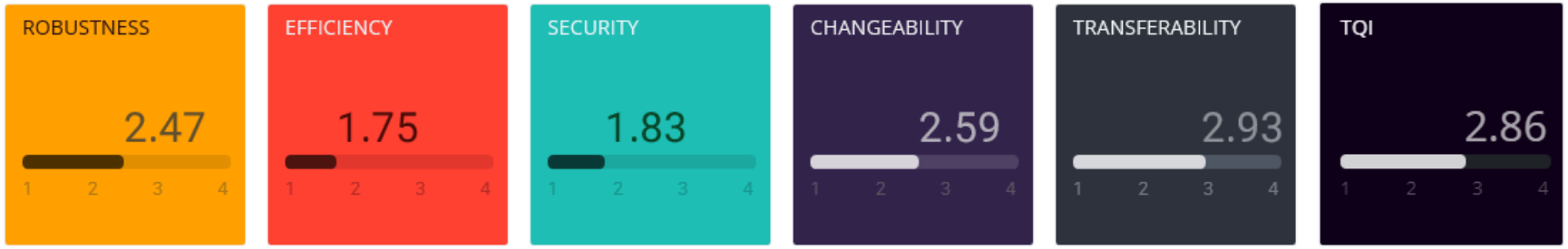
OWASP

ISO

RISICO

ONDERHOUDBAARHEID

OVERALL



Risico op kritische incidenten in productie

Risico op performance / scaling issues

Risico op security breaches

Snelheid en gemak doorvoeren van changes

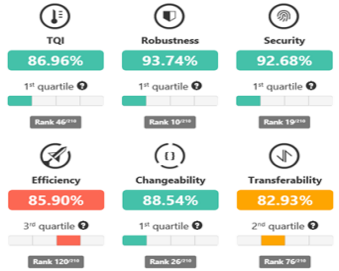
Snelheid en gemak leren

Overall Quality

De 'never-to-do's'. Moeten worden onderzocht

Kosten om alle High, 50% Medium en 25% Low severity violations

Technische omvang in LOC



BENCHMARK tegen duizenden vergelijkbare applicaties

metri
IT STARTS WITH THE FACTS

Objectief & Transparant Kwaliteitsmodel

Berekening van %compliance met de metrics en vaststellen van een score van 1 tot 4

Gewogen gemiddelde van de technical criteria

CISQ

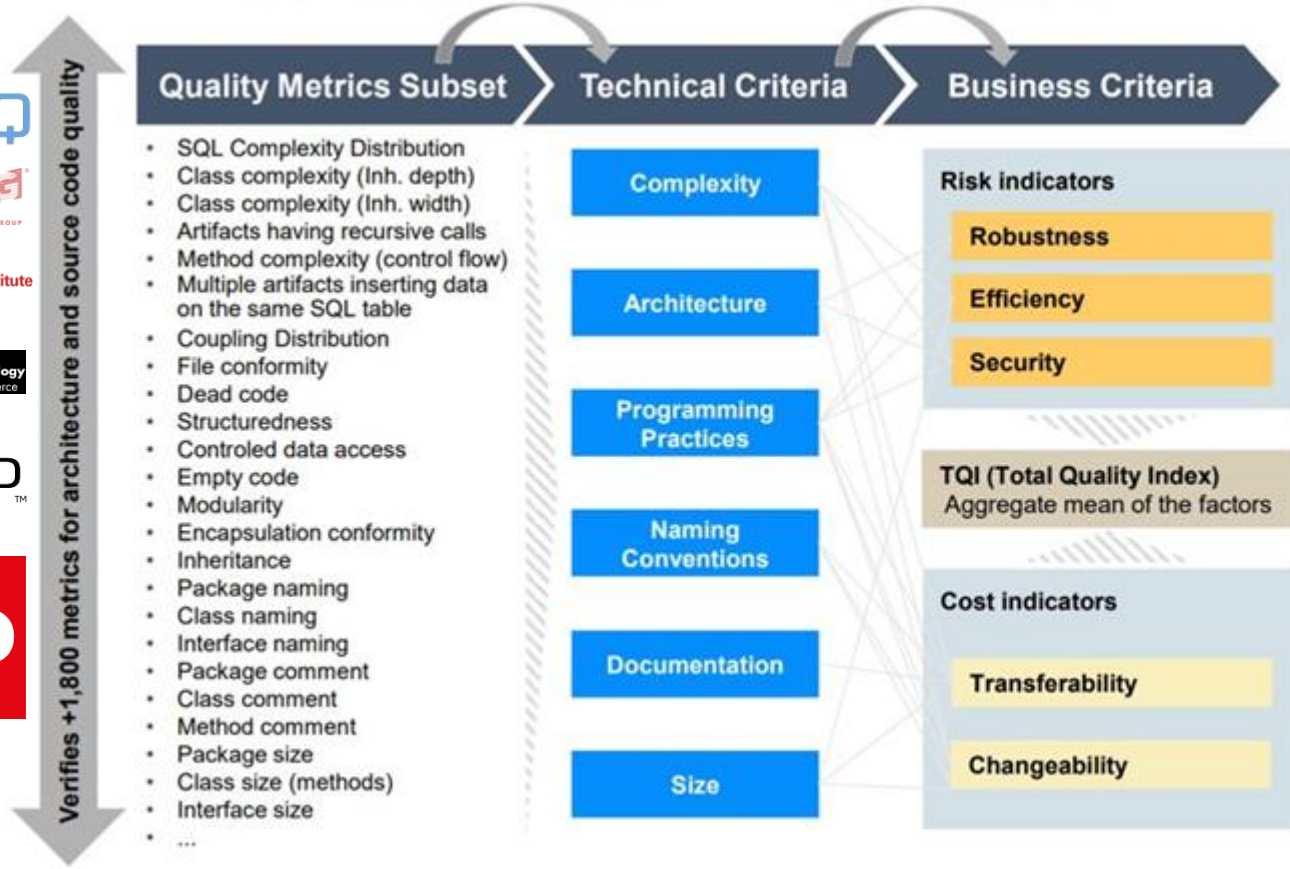
OMG
OBJECT MANAGEMENT GROUP

Software Engineering Institute

NIST National Institute of Standards and Technology U.S. Department of Commerce

OWASP™

ISO



Directe impact

Lange termijn impact

metri
IT STARTS WITH THE FACTS

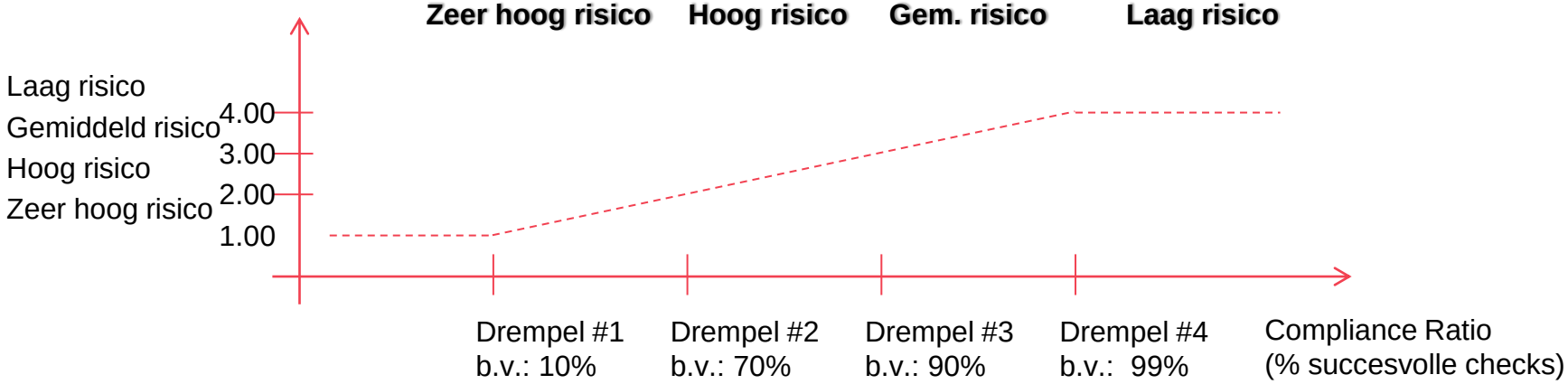
Berekening score 1 to 4

- Berekend per regel, component, module en applicatie.
- Onder de drie nemen de risico's en kosten toe.

$$\text{Compliance Ratio} = \frac{\text{succesvolle checks}}{\text{succesvolle checks + onsuccesvolle checks}} \times 100$$

(% succesvolle checks)

Zeer hoog risico **Hoog risico** **Gem. risico** **Laag risico**

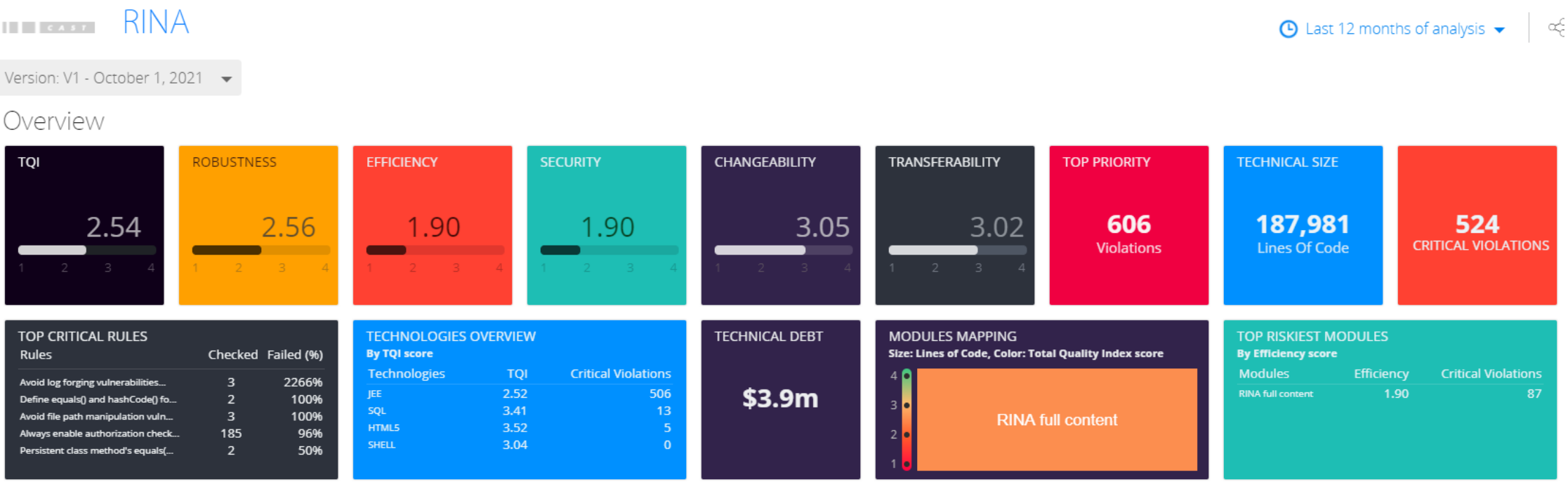


metri
IT STARTS WITH THE FACTS

Management Samenvatting

UWV RINA Health Dashboard

12



metri
IT STARTS WITH THE FACTS

Open Source risico's

CAST Consolidates one of the largest databases on software components, with billions of signatures (fingerprints) computed for each version of open source and third-party components. Based on this unique database Highlight compares your application files fingerprint and aggregates component, version, license, release date information at both portfolio and application levels.

Third-Party License Risk

1806	38	27	1734
Components	HIGH RISK	MEDIUM RISK	LOW RISK

Possible Vulnerabilities

17	53	36	4
CRITICAL	HIGH	MEDIUM	LOW

Management Samenvatting

CAST AIP

- Er zit relatief veel risico in de applicatie. Dit komt tot uiting in lage scores voor Efficiency, Security en Robustness. In totaal zijn er 542 Kritieke fouten (Critical Violations = CVE's) gevonden.
- De score voor Security is zeer laag. Aanbeveling is uit te zoeken of er direct gevaar is, met name in de 'Input Validation' sectie of dat er een andere manier is gebruikt het risico af te dekken.
- De technical Debt is hoger dan marktgemiddeld.
- De documentatiegraad van de code is laag. 2/3 van de methodes is niet becommentarieerd. Doordat de complexiteit niet hoog is, zijn de scores op Changeability en Transferability toch hoger dan 3.
- Er is een VerbeterPlan opgenomen met 158 acties in het Engineering dashboard. Uitvoering leidt tot scores van 3 of daarboven en kost zo'n 15 dagen werk van een developer (excl. test, implementatie, overhead, etc.).

Open Source Analyse (Highlight) – Software Bill of Materials (wordt meegestuurd in Excel)

- Er zijn 1806 3rd party componenten gebruikt.
  - 38 componenten hebben een licentie met hoog risico
  - 17 componenten hebben een zeer hoog risico, 53 een hoog risico m.b.t. bekende problemen in de code.

Observaties en Aanbevelingen

Observaties

- De documentatie die bij de applicatie is aangeleverd is uitgebreid en up-to-date.
- Het is opmerkelijk dat de health factors die de risico's in de code meten laag zijn, terwijl de factoren die iets zeggen over de kosten/onderhoudbaarheid op een acceptabel niveau liggen.
- De applicatie is op zich zelf niet heel erg slecht, alleen is wel duidelijk dat de applicatie niet is ontwikkeld met veel respect voor best practice codeerstandaarden.

Aanbevelingen

- Controleer z.s.m. de Security gerelateerde CVE's, met name die op het 'Input Validation' vlak.
- Controleer de (zeer) hoog risico open source compenenten (Bill of Materials wordt meegestuurd)
- De CAST licentie van UWV verloopt op 31-03-2022. Het is aan te bevelen deze te verlengen en ook Imaging toe te voegen aan de licentie. Met imaging wordt een blueprint gemaakt van de applicatie, waardoor op verschillende niveaus inzicht wordt verkregen in de exacte werking van de applicatie (de connecties/calls tussen modules/lagen/frameworks/etc.)

Verbeterplan en indicatie kosten

- Er zijn 158 acties opgenomen op het verbeterplan. Met de Actieplan Optimizer wordt gesimuleerd wat de scores worden als het plan volledig is uitgevoerd.

Application Name	Business criterion	Current grade	Simulation grade	Delta
RINA	Transferability	3,02	3,12	10,00%
RINA	Changeability	3,05	3,11	6,00%
RINA	Robustness	2,56	3,39	83,00%
RINA	Efficiency	1,90	3,20	130,00%
RINA	Security	1,90	3,35	145,00%
RINA	Total Quality Index	2,54	3,26	72,00%
RINA	Programming Practices	2,50	3,21	71,00%
RINA	Architectural Design	3,24	3,24	0,00%
RINA	Documentation	2,70	2,90	20,00%

- Het uitvoeren van deze acties kost in totaal zo’n 15 mandagen aan development tijd. Hier zit geen test, implementatie, overhead tijd, etc. bij.

Detail Slides

Critical Violations (CVE's)

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Architecture - Multi-Layers and Data Access		0	13	100%	N/A	N/A	3.68
Programming Practices - Unexpected Behavior		12	41,540	81%	N/A	N/A	1.00
Efficiency - Memory, Network and Disk Space Management		9	69,348	97%	N/A	N/A	1.00
Programming Practices - Error and Exception Handling		65	33,288	99%	N/A	N/A	1.00
Complexity - OO Inheritance and Polymorphism		8	96	92%	N/A	N/A	1.00
Secure Coding - Input Validation		92	129,251	71%	N/A	N/A	1.00
Efficiency - Expensive Calls in Loops		64	16,762	99%	N/A	N/A	1.14
Efficiency - SQL and Data Handling Performance		14	446	98%	N/A	N/A	1.00
Programming Practices - Modularity and OO Encapsulation Conformity		36	13,378	99%	N/A	N/A	2.42
Secure Coding - API Abuse		5	33,561	99%	N/A	N/A	3.37
Programming Practices - OO Inheritance and Polymorphism		16	63,982	99%	N/A	N/A	2.85
Secure Coding - Weak Security Features		192	292,597	93%	N/A	N/A	1.00
Secure Coding - Encapsulation		0	115	100%	N/A	N/A	3.85
Secure Coding - Time and State		11	30,083	93%	N/A	N/A	1.00

Laag volume CVE's met hoge impact

19

Quick facts

Grade	Compliance	Appmarq	Total Violations	Critical Violations	New Violations	Added Critical Violations	Removed Violations
2.54	90.93%	2nd Quartile	47,097	524	47,097	524	0

Removed Critical Violations	Technical Debt
0	$3.9m

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Programming Practices - Unexpected Behavior		**12**	**28,058**	**62%**	**N/A**	**N/A**	**1.00**
Define equals() and hashCode() for component		2	0	0%	23%	-23%	1.00
Persistent class method's equals() and hashCode() must access its fields through getter methods		1	1	50%	42%	8%	1.00
CWE-681: Avoid numerical data corruption during incompatible mutation		6	14,646	99%	0%	99%	4.00
Avoid directly instantiating a Class used as a managed bean		3	13,411	99%	89%	10%	3.97
Efficiency - Memory, Network and Disk Space Management		**9**	**14,654**	**94%**	**N/A**	**N/A**	**1.00**
Avoid missing release of stream connection after an effective lifetime		6	50	89%	23%	66%	1.00
Avoid thread creation for application running on application server		3	14,604	99%	100%	-1%	3.98
Complexity - OO Inheritance and Polymorphism		**8**	**96**	**92%**	**N/A**	**N/A**	**1.00**
Avoid classes overriding only equals() or only hashCode()		8	96	92%	83%	9%	1.00
Secure Coding - Time and State		**11**	**17,735**	**91%**	**N/A**	**N/A**	**1.00**
Avoid non thread safe singleton		3	9	75%	58%	17%	1.00
Avoid to use this within Constructor in multi-thread environment		7	3,017	99%	98%	1%	3.55
Avoid using predictable SecureRandom Seeds		1	14,709	99%	N/A	N/A	4.00

Security

IT STARTS WITH THE FACTS.

metri

IT STARTS WITH THE FACTS

Security

- De score op Security is lager dan mag worden verwacht.
- Er zijn 92 kritieke fouten gevonden m.b.t. Secure Coding – Input Validation (van de129.343 checks)
- Dit zou direct gevaar op kunnen leveren op een security breach en het is belangrijk dat ontwikkelaars onderzoeken of dit gevaar op een andere manier is afgedekt.
- Het kan bijvoorbeeld zijn dat er gebruik wordt gemaakt van een veilige omgeving, en dat de applicatie niet wordt blootgesteld aan de buitenwereld. Of dat er eerder in het proces al een check wordt uitgevoerd.

Security (1)

V1 - October 1, 2021 / Security

Quick facts

Grade	Compliance	Appmarq	Total Violations	Critical Violations	New Violations	Added Critical Violations	Removed Violations
1.90	94.04%	3rd Quartile	2,749	386	2,749	386	0

Removed Critical Violations
0

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Programming Practices - Unexpected Behavior		**12**	**28,058**	**62%**	**N/A**	**N/A**	**1.00**
CWE-681: Avoid numerical data corruption during incompatible mutation		6	14,646	99%	0%	99%	4.00
Avoid directly instantiating a Class used as a managed bean		3	13,411	99%	89%	10%	3.97
Define equals() and hashCode() for component		2	0	0%	23%	-23%	1.00
Persistent class method's equals() and hashCode() must access its fields through getter methods		1	1	50%	42%	8%	1.00
Secure Coding - Input Validation		**92**	**19,193**	**-240%**	**N/A**	**N/A**	**1.00**
Avoid file path manipulation vulnerabilities through API requests		3	0	0%	N/A	N/A	1.00
Avoid file path manipulation vulnerabilities		2	169	98%	99%	-1%	1.83
Avoid using XMLStreamReader without restriction of XML External Entity Reference (XXE)		1	14,709	99%	N/A	N/A	4.00
Avoid log forging vulnerabilities through API requests		68	-65	-2167%	N/A	N/A	1.00
Avoid log forging vulnerabilities		14	157	91%	98%	-7%	1.00
Avoid bypassing angular security trust		2	4,040	99%	100%	-1%	3.92
Avoid using submit markup related to "form" with id attribute		2	183	98%	93%	5%	1.92
Programming Practices - Error and Exception Handling		**65**	**32,591**	**97%**	**N/A**	**N/A**	**1.00**
Avoid directly throwing instance of Exception class		45	13,424	99%	99%	0%	3.34
Avoid empty catch blocks		16	19,117	99%	99%	0%	3.85
Never exit a finally block with a return, break, continue, or throw		4	50	92%	93%	-1%	1.00

Security (2)

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Secure Coding - Weak Security Features		**192**	**58,858**	**73%**	**N/A**	**N/A**	**1.00**
Always enable authorization checks at function level for functions called on by APIs		178	7	3%	N/A	N/A	1.00
Avoid using 'java.System.exit()'		6	25,388	99%	100%	-1%	3.97
Avoid use of a reversible one-way hash		2	3	60%	N/A	N/A	1.00
Avoid weak cryptographic algorithm		2	2	50%	N/A	N/A	1.00
Avoid using risky cryptographic hash (JEE)		2	14,708	99%	N/A	N/A	3.99
Avoid hard-coded network resource names (Typescript)		1	4,041	99%	N/A	N/A	3.97
Avoid using deprecated SSL protocols to secure connection		1	14,709	99%	N/A	N/A	4.00
Secure Coding - API Abuse		**5**	**14,796**	**99%**	**N/A**	**N/A**	**3.37**
Avoid hardcoded network resource names (JEE)		5	14,796	99%	100%	-1%	3.95
Efficiency - Memory, Network and Disk Space Management		**9**	**14,654**	**94%**	**N/A**	**N/A**	**1.00**
Avoid thread creation for application running on application server		3	14,604	99%	100%	-1%	3.98
Avoid missing release of stream connection after an effective lifetime		6	50	89%	23%	66%	1.00
Secure Coding - Time and State		**11**	**17,735**	**91%**	**N/A**	**N/A**	**1.00**
Avoid non thread safe singleton		3	9	75%	58%	17%	1.00
Avoid to use this within Constructor in multi-thread environment		7	3,017	99%	98%	1%	3.55
Avoid using predictable SecureRandom Seeds		1	14,709	99%	N/A	N/A	4.00

Avoid file path manipulation vulnerabilities through API requests

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.ee ... .controller.CasesController.getCase		Added
eu.ec.dgempl.ee ... troller.CasesController.searchCases		Added
eu.ec.dgempl.ee ... ntroller.GroupsController.getGroups		Added
eu.ec.dgempl.ee ... troller.executeCaseAssignmentAction		Added
eu.ec.dgempl.ee ... tController.deleteCommentOnDocument		Added
eu.ec.dgempl.ee ... er.UsersController.getUsersForGroup		Added
eu.ec.dgempl.ee ... mmentController.deleteCommentOnCase		Added
eu.ec.dgempl.ee ... mmentController.submitCommentOnCase		Added
eu.ec.dgempl.ee ... tController.submitCommentOnDocument		Added
eu.ec.dgempl.ee ... ntroller.CasesController.assignCase		Added

Avoid log forging vulnerabilities through API requests

Rule Documentation

Name	Avoid log forging vulnerabilities through API requests
Rationale	Writing unvalidated, unsanitized user input to log files can allow an attacker to forge log entries or inject malicious content into the logs. Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually or sometimes automated with a tool that automatically gathers log data for important events or trending information. Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim.
Tags	A3-2017, A6-2013, API10-2019, CWE-93, CWE-117, DATA-SAFETY, NIST-AU-9, NIST-SI-10, PCI-Requirement-6.5.1, PCI-Requirement-6.5.1, STIG-V-69369, STIG-V-222444
Description	This metric uses CAST data-flow engine to detect a call path where input data from API requests is written into the application logs without prior validation and sanitization.
Remediation	Use authorized sanitization methods.
Reference	*CWE-117: Improper Output Neutralization for Logs* https://cwe.mitre.org/data/definitions/117.html

Avoid log forging vulnerabilities

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.web.rest.controller.CasesController.searchCases		Added
eu.ec.dgempl.eessi.web.rest.co ... troller.AuditController.retrieveDetailsForDay		Added
eu.ec.dgempl.eessi.web.rest.controller.AuditController.retrieveTimeSlots		Added
eu.ec.dgempl.eessi.web.rest.controller.GroupsController.getGroups		Added
eu.ec.dgempl.eessi.web.rest.controller.UsersController.getUsersForGroup		Added
eu.ec.dgempl.eessi.web.rest.controller.UsersController.getUsers		Added
eu.ec.dgempl.eessi.web.rest.controller.EntityController.searchEntities		Added
eu.ec.dgempl.eessi.web.spring. ... uthenticationFilterToRinaCas.doFilterInternal		Added
eu.ec.dgempl.eessi.web.rest.controller.ResourceAdminController.getResources		Added
eu.ec.dgempl.eessi.web.rest.co ... roller.ResourceAdminController.updateResource		Added

Avoid log forging vulnerabilities

tule Documentation

Name	Avoid log forging vulnerabilities
Rationale	Writing unvalidated, unsanitized user input to log files can allow an attacker to forge log entries or inject malicious content into the logs. Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually or sometimes automated with a tool that automatically gathers log data for important events or trending information. Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim.
Tags	A3-2017 A6-2013 CWE-93 CWE-117 DATA-SAFETY NIST-AU-9 PCI-Requirement-6.5.1 PCI-Requirement-6.5.1 STIG-V-69369 STIG-V-222444
Description	This metric uses CAST data-flow engine to detect a call path where input data from the user is written into the application logs without prior validation and sanitization.
Remediation	Use authorized sanitization methods.
Reference	*CWE-117: Improper Output Neutralization for Logs* https://cwe.mitre.org/data/definitions/117.html

Avoid file path manipulation vulnerabilities

28

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.rina.commons.keystore.utils.KeyStoreImport.main		Added
eu.ec.dgempl.eessi.web.rinaclients.keystore.utils.KeyStoreImport.main		Added

metri
IT STARTS WITH THE FACTS

Avoid file path manipulation vulnerabilities

Rule Documentation

Name	Avoid file path manipulation vulnerabilities
Rationale	The software does not properly neutralize special elements that are part of paths or file names used in file system operations. This could allow an attacker to access or modify system files or other files that are critical to the application.
Tags	A5-2017, A7-2013, ASCSM-CWE-22, CWE-22, CWE-22, CWE-22, CWE-23, CWE-23, CWE-23, CWE-36, CWE-36, CWE-36, CWE-73, NIST-AC-3, NIST-SI-10, PCI-Requirement-6.5.1, PCI-Requirement-6.5.1, PCI-Requirement-6.5.8, PCI-Requirement-6.5.8
Description	Using CAST data-flow engine, this metric detects execution paths from user input methods down to file creation methods, paths which are open vulnerabilities to operating system injection flaws.
Remediation	Assume all input is malicious. Avoid using inputs. If it is not possible, use an "accept known good" input validation strategy, i.e., use stringent white-lists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack.

Avoid file path manipulation vulnerabilities through API requests

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.web.rest.controller.UploadFilesController.submitFile		Added
eu.ec.dgempl.eessi.web.rest.co ... roller.ApListenerController.onReceivedMessage		Added
eu.ec.dgempl.eessi.web.rest.co ... icationProfileController.updateRecoveryVolume		Added

Avoid file path manipulation vulnerabilities through API requests

Rule Documentation

Name	Avoid file path manipulation vulnerabilities through API requests
Rationale	The software does not properly neutralize special elements that are part of paths or file names used in file system operations. This could allow an attacker to access or modify system files or other files that are critical to the application.
Tags	A5-2017, A7-2013, ASCSM-CWE-22, CWE-22, CWE-22, CWE-22, CWE-23, CWE-23, CWE-23, CWE-36, CWE-36, CWE-36, CWE-73, NIST-AC-3, NIST-SI-10, PCI-Requirement-6.5.1, PCI-Requirement-6.5.1, PCI-Requirement-6.5.8, PCI-Requirement-6.5.8
Description	Using CAST data-flow engine, this metric detects execution paths from API requests down to file creation methods, paths which are open vulnerabilities to operating system injection flaws.
Remediation	Assume all input is malicious. Avoid using inputs. If it is not possible, use an "accept known good" input validation strategy, i.e., use stringent white-lists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack.

Avoid using submit markup related to "form" with id attribute

32

OBJECT NAME LOCATION	RISK	STATUS
C:\ProgramData\CAST\AipConsole ... \resources\templates\casLoginMessageView.html		Added
C:\ProgramData\CAST\AipConsole ... \resources\templates\fragments\loginform.html		Added

metri
IT STARTS WITH THE FACTS

Avoid using submit markup related to "form" with id attribute

Rule Documentation

Name	Avoid using submit markup related to "form" with id attribute
Rationale	An attacker can use a form to do spam emailing. Ensure the form markup does not contain id attribute.
Tags	A3-2013 A7-2017 ASCSM-CWE-79 CROSS-SITE-VULNERABILITY CWE-79 CWE-79 CWE-79 NIST-SI-10 PCI-Requirement-6.5.7 PCI-Requirement-6.5.7 STIG-V-70257 STIG-V-222602
Description	Do not allow users to use submit markup related to "form" with id attribute.
Reference	https://html5sec.org/#1 http://www.whatwg.org/specs/web-apps/current-work/multipage/association-of-controls-and-forms.html#attr-fs-formaction
Sample	<form id="myid"><input type="submit" value="Submit"></form>
Output	Associated to each violation, the following information is provided: - The number of violation occurrences - Bookmarks for violation occurrences found in the source code
Total	Number of HTML Contents

equals() and hashCode() should be defined for Hibernate/JPA component

34

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.ee ... model.jpa.entity._abstraction.Audit		Added
eu.ec.dgempl.ee ... a.model.jpa.entity._abstraction.Log		Added

metri
IT STARTS WITH THE FACTS

equals() and hashCode() should be defined for Hibernate/JPA component

Rule Documentation

Name	equals() and hashCode() should be defined for Hibernate/JPA component
Rationale	Persistent classes do not have an identifier property. You must implement equals() and hashCode(). Hibernate or JPA implementation relies on this equality routine to check instances for modifications. A custom implementation of equals() and hashCode() is not required for all component classes. However, it is recommended for any persistent class because the implementation is straightforward and safer.
Tags	ASCRM-RLB-4 CWE-581 CWE-1097 CWE-1097 CWE-1097
Description	This rule reports all hibernate and/or JPA persistent classes associated with a component (component and composite-element) declared in the mapping file or through annotation (@Embeddable and @IdClass) and that don't implement equals() and hashCode().
Remediation	Implement equals() and hashCode().
Reference	http://docs.jboss.org/hibernate/core/3.3/reference/en/html/components.html *Hibernate in Action (ISBN 1932394-15-X) p 217* *The Java Persistence API page 252 - ISBN 1-932394-88-5*

Persistent class method's equals() and hashCode() must access its fields through getter methods

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.rina.model. ... AbstractPersistableWithSidAndVersion.hashCode		Added

Persistent class method's equals() and hashCode() must access its fields through getter methods

Rule Documentation

Name	Persistent class method's equals() and hashCode() must access its fields through getter methods
Rationale	This is important, since the object instance passed as other might be a proxy object, not the actual instance that holds the persistent state. This is the case when there are lazy associations between classes. This is one point where Hibernate is not completely transparent, but it's good practice to use accessor methods instead of direct instance variable access: when tuning the performance of the application, a lazy association might be required and the issue can occur. This potential issue raises a ClassCastException and can cause the application to become unstable.
Tags	No Tag available
Description	This rule reports all Hibernate or JPA persistent class method's equals() and hashCode() that directly access its own fields without using getter methods. We consider also the case where equals and hashCode are implemented on an inherited class of the persistent class (this can happen when inherited fields are persistent).
Remediation	Use the getter instead.
Reference	*Hibernate in Action (ISBN 1932394-15-X) p 125, Java Persistence with Hibernate (ISBN 1-932394-88-5) p 400, The Java Persistence API page 400 - ISBN 1-932394-88-5*

Never exit a finally block with a return, break, continue, or throw statements

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.apclient.signing.XMLSigningUtils.getSignatureNode		Added
eu.ec.dgempl.apclient.signing.XMLSigningUtils.buildDOM		Added
eu.ec.dgempl.eessi.rina.commons.keystore.KeyStoreOperations.getCertificate		Added
eu.ec.dgempl.eessi.rina.commons.system.SocketUtil.isPortAvailable		Added

Never exit a finally block with a return, break, continue, or throw statements

Rule Documentation

Name	Never exit a finally block with a return, break, continue, or throw statements
Rationale	Care must be taken if completion of a try-catch block occurs as a result of executing a return. If a finally block also returns a value, then that return supersedes any previous return in the try-catch block. Also, if an exception was thrown in the try or catch blocks that was not caught, then execution of a return in the finally block prevents the exception from being thrown to the caller (because it is not possible for the caller to simultaneously evaluate the return and catch the exception). This is also valid for break or continue instructions.
Tags	CWE-584
Description	When the code has a jump statement inside a finally block, it will cause any thrown exception in the try/catch block to be discarded. This rule raises violation when detecting method that contains an abrupt in a finally block. An abrupt completion of a statement or block occurs when it throws an exception, executes a break or continue to an enclosing statement, or executes a return from the method thereby preventing the exception from being thrown to the caller.
Reference	*OWASP* *http://www.owasp.org/index.php/Return_Inside_Finally_Block* *CERT* *https://www.securecoding.cert.org/confluence/display/java/ERR04-J.+Do+not+exit+abruptly+from+a+finally+block*

metri
IT STARTS WITH THE FACTS

Always enable authorization checks at function level for functions called on by APIs

OBJECT NAME LOCATION	RISK	STATUS
C:\ProgramData\CAST\AipConsole ... CasesController.java/Cases/PORT/GET/Cases/{}/		Added
C:\ProgramData\CAST\AipConsole ... java/message/PORT/POST/message/status-update/		Added
C:\ProgramData\CAST\AipConsole ... ontroller.java/Forms/PORT/GET/Forms/Metadata/		Added
C:\ProgramData\CAST\AipConsole ... a/message/PORT/POST/message/signature-update/		Added
C:\ProgramData\CAST\AipConsole ... PORT/POST/Cases/{}/Documents/{}/Subdocuments/		Added
C:\ProgramData\CAST\AipConsole ... .java//PORT/PUT/Cases/{}/Actions/{}/Document/		Added
C:\ProgramData\CAST\AipConsole ... RT/PUT/Cases/{}/Documents/{}/Subdocuments/{}/		Added
C:\ProgramData\CAST\AipConsole ... /PORT/GET/Cases/{}/Documents/{}/Subdocuments/		Added
C:\ProgramData\CAST\AipConsole ... Controller.java/AuditLogs/PORT/GET/AuditLogs/		Added
C:\ProgramData\CAST\AipConsole ... er\CasesController.java/Cases/PORT/GET/Cases/		Added

Always enable authorization checks at function level for functions called on by APIs

Rule Documentation

Name	Always enable authorization checks at function level for functions called on by APIs
Rationale	Exploitation requires the attacker to send legitimate API calls to the API endpoint that they should not have access to. These endpoints might be exposed to anonymous users or regular, non-privileged users. It's easier to discover these flaws in APIs since APIs are more structured, and the way to access certain functions is more predictable (e.g., replacing the HTTP method from GET to PUT, or changing the "users" string in the URL to "admins"). When defining APIs inside controllers and the methods called on by APIs to perform certain tasks it is important to restrict the user(authenticated or otherwise) based on her role.
Tags	A5-2017 A7-2013 API5-2019 CWE-269 CWE-285 CWE-424 CWE-424 CWE-424 CWE-862 NIST-AC-3
Description	The rule will raise a violation when it finds functions that are called by APIs to perform actions, if they do not implement any form of authorization check.
Remediation	Review your API endpoints against function level authorization flaws, while keeping in mind the business logic of the application and groups hierarchy. Put restrictions on actions like Delete, Put after reviewing the context thoroughly.
Reference	https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa5-broken-function-level-authorization.md

Avoid weak cryptographic algorithm

42

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.rina.common ... eystore.utils.MsPvkUtil.encryptPrivateKeyBlob		Added
eu.ec.dgempl.eessi.rina.common ... eystore.utils.MsPvkUtil.decryptPrivateKeyBlob		Added

metri
IT STARTS WITH THE FACTS

Avoid weak cryptographic algorithm

Rule Documentation

Name	Avoid weak cryptographic algorithm
Rationale	The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected. Well-known techniques may exist to break the algorithm.
Tags	A3-2017 A6-2013 ASCSM-CWE-327 CWE-327 M5-2016 NIST-SC-12 PCI-Requirement-4.1 PCI-Requirement-4.1 PCI-Requirement-6.5.4 PCI-Requirement-6.5.4 STIG-V-70245 STIG-V-222596
Description	Using CAST data-flow engine, this metric detects the use of a broken or risky cryptographic algorithm.
Remediation	Use a recommended algorithm. Example: Advanced Encryption Standard (AES).
Reference	*CWE-327: Use of a Broken or Risky Cryptographic Algorithm* https://cwe.mitre.org/data/definitions/327.html *Open Web Application Security Project (OWASP)* https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure

Avoid use of a reversible one-way hash

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.rina.common ... keystore.utils.MsPvkUtil.getEncryptedInternal		Added
eu.ec.dgempl.eessi.rina.commons.keystore.utils.MsPvkUtil.loadEncrypted		Added

Avoid use of a reversible one-way hash

Rule Documentation

Name	Avoid use of a reversible one-way hash
Rationale	This weakness is especially dangerous when the hash is used in security algorithms that require the one-way property to hold. For example, if an authentication system takes an incoming password and generates a hash, then compares the hash to another hash that it has stored in its authentication database, then the ability to create a collision could allow an attacker to provide an alternate password that produces the same target hash, bypassing authentication.
Tags	A3-2017 A6-2013 CWE-328 PCI-Requirement-6.5.4 PCI-Requirement-6.5.4 STIG-V-70245 STIG-V-222596
Description	Using CAST data-flow engine, this metric detects the use of a risky hashing algorithm.
Remediation	Use a recommended hash method. Example: SHA-2.
Reference	*CWE-328: Reversible One-Way Hash* https://cwe.mitre.org/data/definitions/328.html *Open Web Application Security Project (OWASP)* https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure

Avoid hard-coded network resource names (Typescript)

46

OBJECT NAME LOCATION	RISK	STATUS
C:\ProgramData\CAST\AipConsole ... ingsComponent.checkForWhoAmIAndApplicationIdC		Added

metri
IT STARTS WITH THE FACTS

Avoid hard-coded network resource names (Typescript)

Rule Documentation

Name	Avoid hard-coded network resource names (Typescript)
Rationale	Built-in remote addresses cause problems when the target is moved. Avoid hard-coded network resources (e.g., IP addresses, URLs, etc.)
Tags	ASCRM-RLB-18 CWE-1051 CWE-1051 CWE-1051 PCI-Requirement-1.3.8 PCI-Requirement-1.3.8
Description	This quality rule reports all artifacts that contain hard-coded - IP addresses (IPv4 and IPv6) - URLs
Remediation	Utilize indirect access to network resources using underlaying operating system calls and relative paths. In case hardcoding is necessary, isolating hard-coded data to installation scripts or configuration files can limit its potential negative impact.
Reference	*ASCRM 1.0, Automated Source Code Reliability Measure, Object Management Group.*
Sample	var a = '127.0.0.1' ; //Or let a = 'fe80:0000:0000:0000:0204:61ff:fe9d:f156/0';
Remediation Sample	//the url is loaded from somewhere else var url = loadServerAddress()

Avoid missing release of stream connection after an effective lifetime

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Avoid missing release of stream connection after an effective lifetime		6	50	89%	23%	66%	1.00

48

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.ap ... latorWithSax.addSedWithHeaderAsBody		Added
eu.ec.dgempl.ap ... are.AntimalwareDispatcher.scanFiles		Added
eu.ec.dgempl.ee tools.es2sql.io.SqlFileWriter.open		Added
eu.ec.dgempl.ee ... toreOperations.addCertificate2Store		Added
eu.ec.dgempl.ee ... .keystore.utils.KeyStoreImport.main		Added
eu.ec.dgempl.ee ... .keystore.utils.KeyStoreImport.main		Added

metri
IT STARTS WITH THE FACTS

Avoid missing release of stream connection after an effective lifetime

Rule Documentation

Name

Avoid missing release of stream connection after an effective lifetime

Rationale

A frequent issue when dealing with stream is resource leak. This mainly comes from an incorrect code that miss to close the stream in any cases. Incorrect resource management is a common source of failures in production applications, with the usual pitfalls being database connections and file descriptors remaining opened after an exception has occurred somewhere else in the code. This leads to application servers being frequently restarted when resource exhaustion occurs, because operating systems and server applications generally have an upper-bound limit for resources.

Tags

ASCRM-CWE-772 ASCSM-CWE-772 CWE-772 CWE-772 CWE-772

Description

Reports methods that open a stream in the body and that:
- doesn't close the outermost stream in a finally block. Note that the number of calls to open a stream and the methods in the finally must be the same.
- or doesn't annotate this stream with @Cleanup annotation (lombok.Cleanup)
- or doesn't use the try with resource to declare the stream that must be closed

The following objects are taken into account:
- output streams
- input streams
- readers
- writers
- channel

Avoid non thread safe singleton

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.apclient.service.db.DAO		Added
eu.ec.dgempl.apclient.service.queue.impl.DequeRegistryDAOImpl		Added
eu.ec.dgempl.nieclient.NieClientImpl		Added

Avoid non thread safe singleton

Rule Documentation

Name	Avoid non thread safe singleton
Rationale	The code contains a function or method that operates in a multi-threaded environment but owns an unsafe non-final static storable or member data element. This issue can prevent the software from running reliably. If the relevant code is reachable by an attacker, then this reliability problem might introduce a vulnerability. The software uses the singleton pattern when creating a resource within a multithreaded environment that may not be thread-safe.
Tags	ASCRM-RLB-11 CWE-543 CWE-543 CWE-543 CWE-1058 CWE-1058 CWE-1058 M7-2016
Description	The implementation of the singleton pattern may not be thread-safe: If several calls were made into this method from a multi-threaded program, you could end up creating multiple instances of the class, and also mess up seriously since the system expects to have only one instance. The problem takes place where you check if the instance is null. This rule reports singleton that initialize the static field that refer to the single instance in a non synchronized method. A singleton is defined as: - a class with a static member with the same type or parent type (extended or implemented) as the class - a static method that refers the instance and return an object of same type or a parent type (extended or implemented) - a class that has only private constructors
Remediation	To remediate to this issue (in case of multi-threaded environment), there is two solutions: - Declare the field that hold the unique instance as static final and initialize it in the declaration - Synchronize the method that initialize the field
Reference	http://tekpool.wordpress.com/2006/10/27/singleton-pattern-part-2-thread-safe-implemenation/ https://javarevisited.blogspot.com/2014/05/double-checked-locking-on-singleton-in-java.html

IT STARTS WITH THE FACTS.

metri

IT STARTS WITH THE FACTS

Efficiency

- De score op Efficiency is lager dan mag worden verwacht: 1,9.
- Er zijn 87 kritieke fouten gevonden die bijdragen aan deze score. Het totaal aantal violations is 1369.
- De meeste kritieke fouten zijn m.b.t. de regel ‘Avoid Instantiations inside loops’: 51 van de 87. Dit betreft een violation van Common Weakness Enumeration CWE-1050, zoals door Mitre gedefinieerd. Toch scoort de applicatie een 3,24 voor deze regel, omdat de compliancy 99% is (>13000 keer succesvol).
- De score wordt voor een groot deel veroorzaakt door in totaal 33 kritische fouten op 4 regels:
  - Avoid direct or indirect remote calls inside a loop: 13 CVE’s
  - Avoid missing release of stream connection after an effective lifetime: 6 CVE’s
  - Lazy fetching should be used for Hibernate collection: 1 CVE
  - Avoid tables without primary key / unique key constraint / unique index: 13 CVE’s
- Deze 33 kritieke fouten zijn ook op het Verbeterplan geplaatst.

Efficiency

Quick facts

Grade	Compliance	Appmarq	Total Violations	Critical Violations	New Violations	Added Critical Violations	Removed Violations
1.90	94.76%	1st Quartile	1,369	87	1,369	87	0

Removed Critical Violations
0

54

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Efficiency - Expensive Calls in Loops		**64**	**14,047**	**98%**	**N/A**	**N/A**	**1.14**
Avoid direct or indirect remote calls inside a loop		13	686	98%	98%	0%	1.14
Avoid instantiations inside loops		51	13,361	99%	98%	1%	3.24
Efficiency - Memory, Network and Disk Space Management		**9**	**14,654**	**94%**	**N/A**	**N/A**	**1.00**
Avoid thread creation for application running on application server		3	14,604	99%	100%	-1%	3.98
Avoid missing release of stream connection after an effective lifetime		6	50	89%	23%	66%	1.00
Efficiency - SQL and Data Handling Performance		**14**	**161**	**93%**	**N/A**	**N/A**	**1.00**
Use lazy fetching for collection		1	66	98%	69%	29%	1.51
Avoid tables without primary key / unique key constraint / unique index		13	95	87%	N/A	N/A	1.00

Avoid direct or indirect remote calls inside a loop

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Efficiency - Expensive Calls in Loops		64	14,047	98%	N/A	N/A	1.14
Avoid direct or indirect remote calls inside a loop		13	686	98%	98%	0%	1.14

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.eessi.rina.buc.common.service.SenderServiceImpl.sendMessages		Added
eu.ec.dgempl.eessi.rina.buc.common.service.SenderServiceImpl.sendMessages		Added
eu.ec.dgempl.eessi.apclient.ms ... r.service.ApClientSenderServiceImpl.configure		Added
eu.ec.dgempl.eessi.rina.buc.co ... mon.service.SenderServiceImpl.resendDocuments		Added
eu.ec.dgempl.eessi.rina.buc.co ... mon.service.SenderServiceImpl.resendDocuments		Added
eu.ec.dgempl.apclient.service. ... e.impl.TechnicalQueueImpl.processEbmsMessages		Added
eu.ec.dgempl.apclient.ApClientManagerImpl.reLoadFilesInQueue		Added
eu.ec.dgempl.apclient.service. ... MessageSignatureDeadMsgQueueProcessor.process		Added
eu.ec.dgempl.apclient.service. ... or.MessageStatusDeadMsgQueueProcessor.process		Added
eu.ec.dgempl.eessi.rina.repo.N ... tificationRepoImpl.getConsolidatedSummaryList		Added

Avoid direct or indirect remote calls inside a loop

Rule Documentation

Name	Avoid direct or indirect remote calls inside a loop
Rationale	Having a loop body or loop condition that contains a control element that directly or indirectly consumes platform resources (e.g. messaging, sessions, locks, or file descriptors) can make the software perform more slowly. Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow. If an attacker can influence the number of iterations in the loop, then this performance problem might allow a denial of service by consuming more platform resources than intended
Tags	ASCPEM-PRF-8 CWE-1050 CWE-1050 CWE-1050
Description	This rules reports violation when detecting one of the following remote call inside a loop at a depth level less than : * SQL Statement through loop (example: SQL cursor on SQL Server, nested cursors on Oracle) * Stored procedure called many times from the client in a loop. * EJB3 Session remote method is a parameter that can be changed at will.
Remediation	The loop execution can be delegated to the server side of the application so that not network latency will occur.

Avoid missing release of stream connection after an effective lifetime

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Avoid missing release of stream connection after an effective lifetime		6	50	89%	23%	66%	1.00

OBJECT NAME LOCATION	RISK	STATUS
eu.ec.dgempl.ap ... latorWithSax.addSedWithHeaderAsBody		Added
eu.ec.dgempl.ap ... are.AntimalwareDispatcher.scanFiles		Added
eu.ec.dgempl.ee tools.es2sql.io.SqlFileWriter.open		Added
eu.ec.dgempl.ee ... toreOperations.addCertificate2Store		Added
eu.ec.dgempl.ee ... .keystore.utils.KeyStoreImport.main		Added
eu.ec.dgempl.ee ... .keystore.utils.KeyStoreImport.main		Added

Avoid missing release of stream connection after an effective lifetime

Rule Documentation

Name	Avoid missing release of stream connection after an effective lifetime
Rationale	A frequent issue when dealing with stream is resource leak. This mainly comes from an incorrect code that miss to close the stream in any cases. Incorrect resource management is a common source of failures in production applications, with the usual pitfalls being database connections and file descriptors remaining opened after an exception has occurred somewhere else in the code. This leads to application servers being frequently restarted when resource exhaustion occurs, because operating systems and server applications generally have an upper-bound limit for resources.
Tags	ASCRM-CWE-772 ASCSM-CWE-772 CWE-772 CWE-772 CWE-772
Description	Reports methods that open a stream in the body and that: - doesn't close the outermost stream in a finally block. Note that the number of calls to open a stream and the methods in the finally must be the same. - or doesn't annotate this stream with @Cleanup annotation (lombok.Cleanup) - or doesn't use the try with resource to declare the stream that must be closed The following objects are taken into account: - output streams - input streams - readers - writers - channel

Lazy fetching should be used for Hibernate collection

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Use lazy fetching for collection		1	66	98%	69%	29%	1.51

OBJECT NAME LOCATION	RISK	STATUS
[C:\ProgramData\CAST\AipConsol ... ssage.java].PendingMessage.pendingAttachments		Added

Lazy fetching should be used for Hibernate collection

Rationale	In a lazy association, the associated object or collection is fetched when it's first accessed. This results in a new request to the database (unless the associated object is cached). This option is almost always used for collection mappings (it should be the default, and we recommend that you consider it as a default for all your collection mappings). Generally, the performance benefits are such that you will want to use lazy instantiation wherever possible (compared with the massive task of reading in all of the entities concerned) When eager fetching is used, the associated object or collection is fetched together with the owning object, using an SQL outer join, and no further database request is required. But the issue is that it will always be done like this and if the number of rows is high, the performance will be affected.
Tags	No Tag available
Description	This rule reports all collection association (set, list, bag, idbag, map) that don't use lazy fetching strategy. This strategy is set through the lazy property set to "true".
Remediation	You need to make sure to use the right FetchType for your use case to avoid common Hibernate performance issues. For most use cases, the FetchType.LAZY is a good choice. EAGER fetching tells Hibernate to get the related entities with the initial query. This can be very efficient because all entities are fetched with only one query. But in most cases it just creates a huge overhead because you select entities you don't need in your use case. You can prevent this with FetchType.LAZY. This tells Hibernate to delay the initialization of the relationship until you access it in your business code. The drawback of this approach is that Hibernate needs to execute an additional query to initialize each relationship.

Avoid tables without primary key / unique key constraint / unique index

61

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Avoid tables without primary key / unique key constraint / unique index		13	95	87%	N/A	N/A	1.00

OBJECT NAME LOCATION	RISK	STATUS
DEFAULT.CASE_SUBJECT_ORG		Added
DEFAULT.NOTIFICATION_USER		Added
DEFAULT.ASSIGNMENT_GROUP		Added
DEFAULT.ASSIGNMENT_USER		Added
DEFAULT.DOC_BVERSION_ATTACHMENT		Added
DEFAULT.DOC_BVERSION_SUBDOC_BVERSION		Added
DEFAULT.SUBDOC_BVERSION_ATTACHMENT		Added
DEFAULT.SEARCH_DEF_GROUP		Added
DEFAULT.SEARCH_DEF_ORG		Added
DEFAULT.SEARCH_DEF_PROC_DEF		Added

metri
IT STARTS WITH THE FACTS

Avoid tables without primary key / unique key constraint / unique index

Rule Documentation

Name	Avoid tables without primary key / unique key constraint / unique index
Rationale	Unique keys should be created for two reasons: data integrity and performance. That's why every huge table should have a Primary Key or Unique Key constraint or Unique Index.
Tags	No Tag available
Description	This metric displays the list of tables that do not have Primary Key / Unique Key constraint / Unique Index. Names of Tables should not start with (TMP|LOG)_% and should not end with %_(TMP|LOG). The prefix and suffix values are parameters, "Table not prefixed by" and "Table not suffixed with", that can be changed at will.
Remediation	Check table and if so, add primary key or unique key constraint or at least unique index.
Reference	https://dotnettutorials.net/lesson/performance-improvement-using-unique-keys/ http://www.geeksengine.com/database/design/primary-key-constraint.php https://www.ibm.com/support/knowledgecenter/SSCJDQ/com.ibm.swg.im.dashdb.admin.dbobj.doc/doc/c0061097.html

IT STARTS WITH THE FACTS.

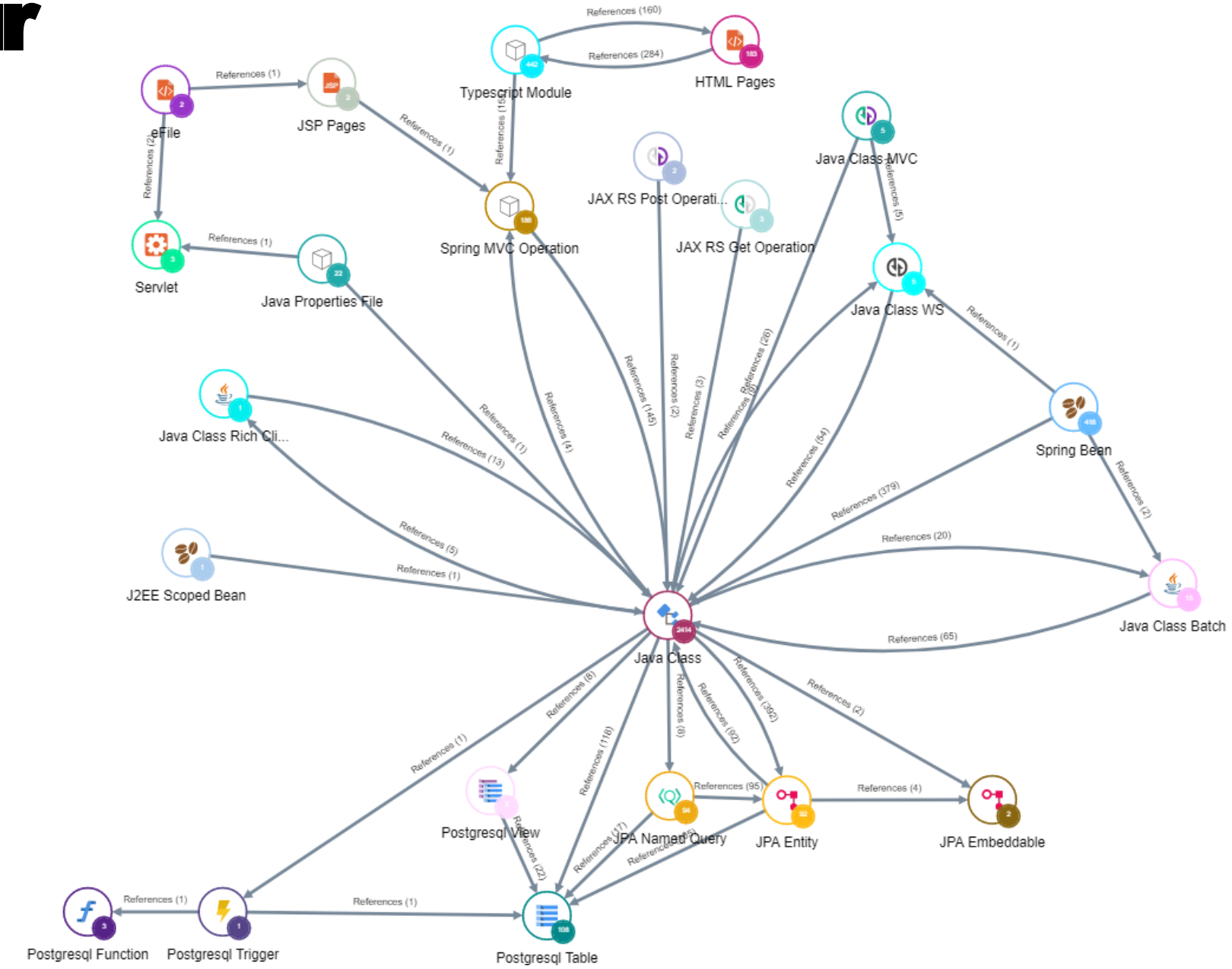
metri

IT STARTS WITH THE FACTS

Architectuur Overzicht

64

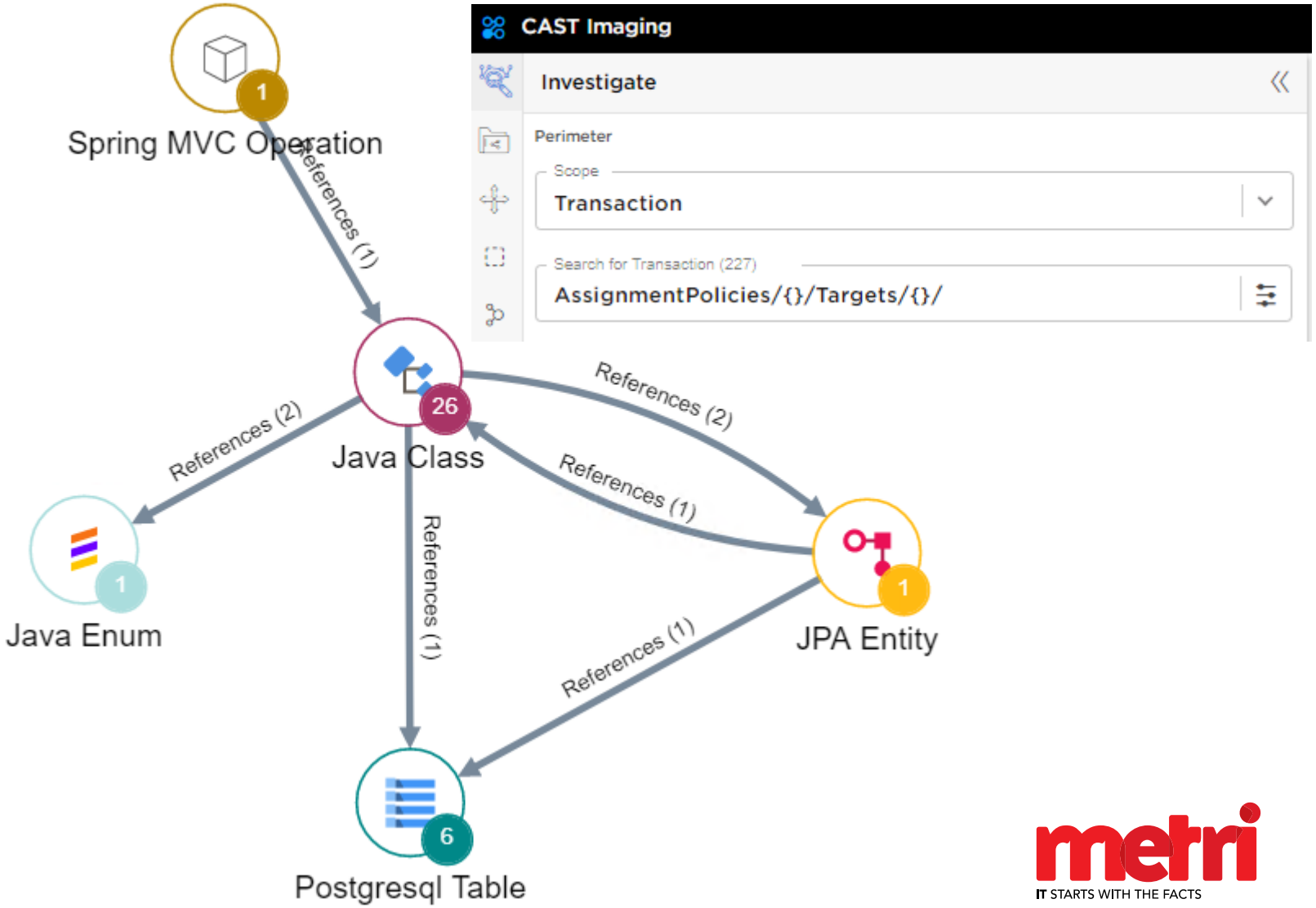
Met CAST Imaging krijgen we een gedetailleerd inzicht in de architectuur van de applicatie



Architectuur – Begrijpen van de transacties

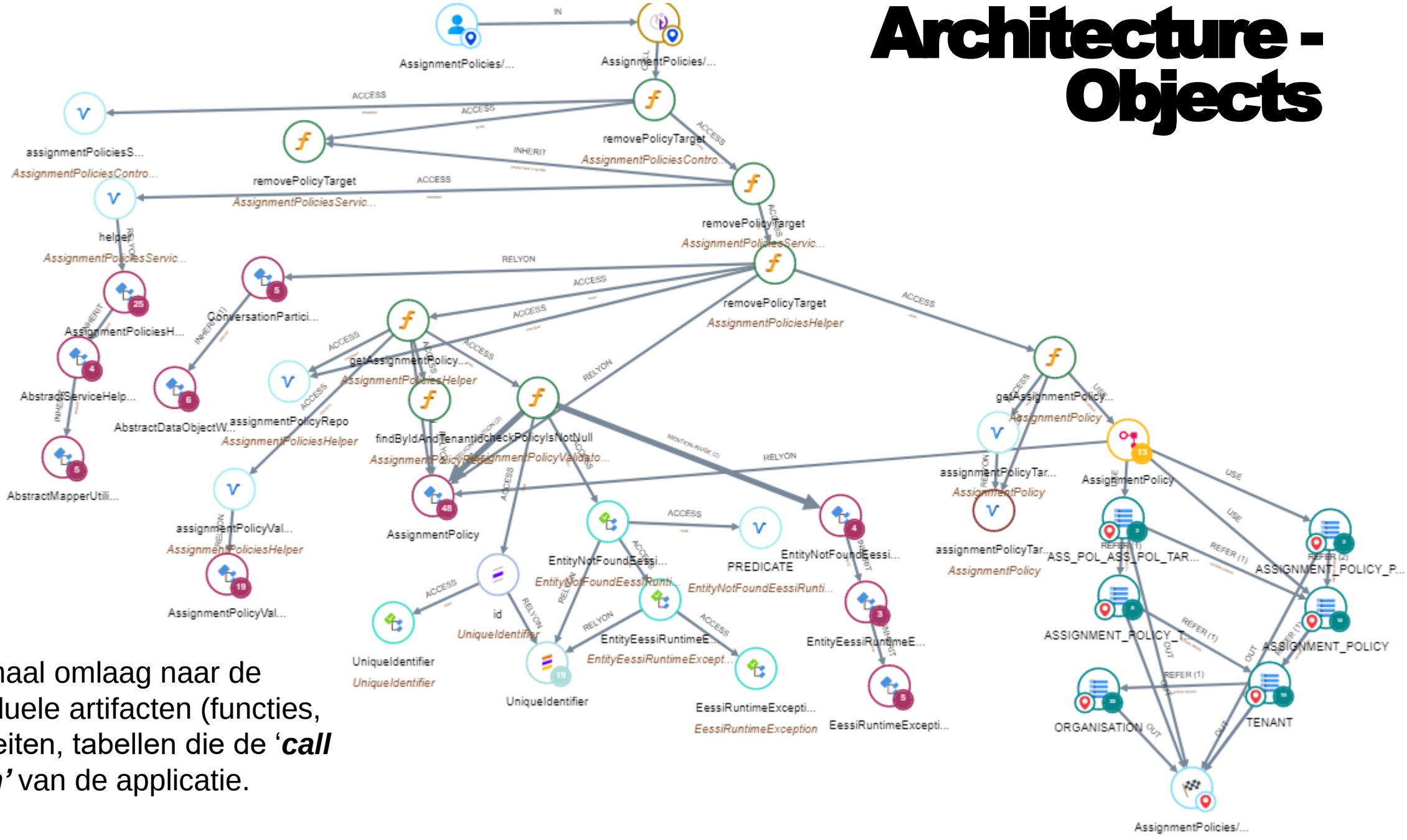
65

… en drill steeds een laag dieper om te begrijpen hoe de applicatie werkt.



metri
IT STARTS WITH THE FACTS

Architecture - Objects



Helemaal omlaag naar de individuele artifacten (functies, entitiiteiten, tabellen die de '***call graph'*** van de applicatie.

Technical Debt / CVE's

IT STARTS WITH THE FACTS.

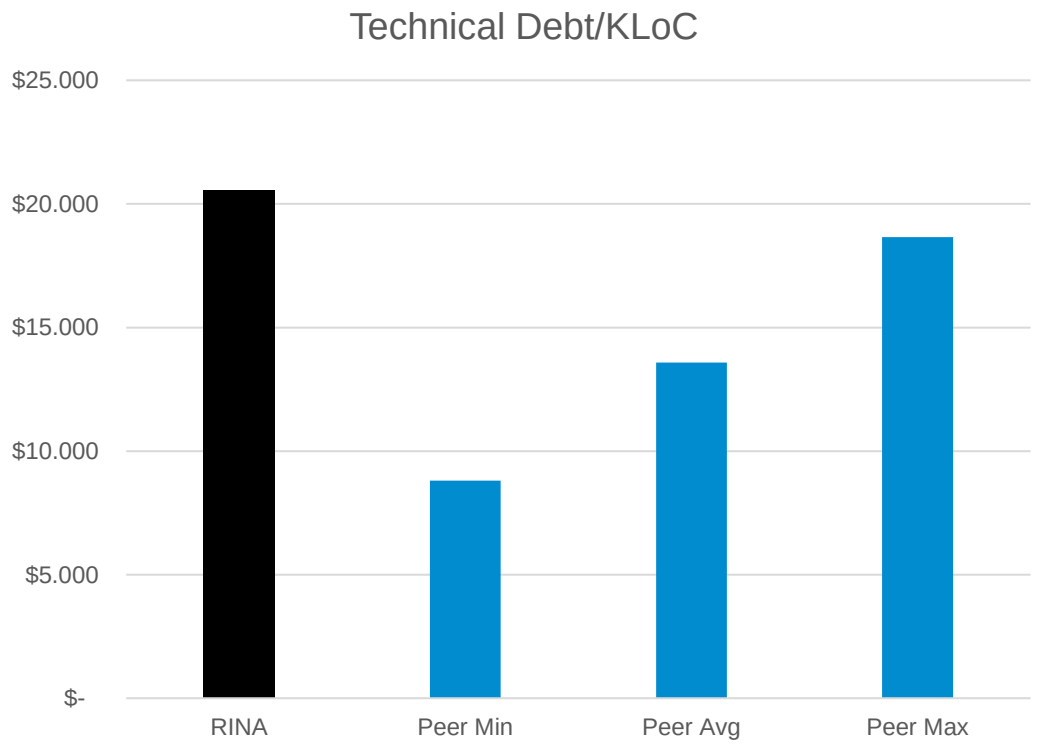
metri

IT STARTS WITH THE FACTS

Technische schuld: $3,9 miljoen

De technische schuld wordt berekend volgens de OMG standaard:

- **50% van de high severity violations wordt opgelost**
- **25% van de medium severity violations wordt opgelost**
- **10% van de low severity violations wordt opgelost**
- **Iedere violation kost 1 uur oplostijd**
- **Het uurtarief is $75**



III. Four Steps for Calculating Technical Debt

Step 1. The density of violations per thousand lines of code (KLOC) is derived from source code analysis using the CAST Application Intelligence Platform.

Step 2. Violations are categorized into low, medium and high severity. The Technical Debt calculation assumes that only 50% of high-severity violations, 25% of medium-severity violations, and 10% of low-severity violations require fixing to prevent business disruption.

Step 3. We conservatively assume that each violation, no matter its level of severity, takes 1 hour to fix at a fully-burdened cost of $75 per hour. Although these numbers could be a lot higher, especially when the fix is applied during operation, we assume these values to produce a conservative estimate.

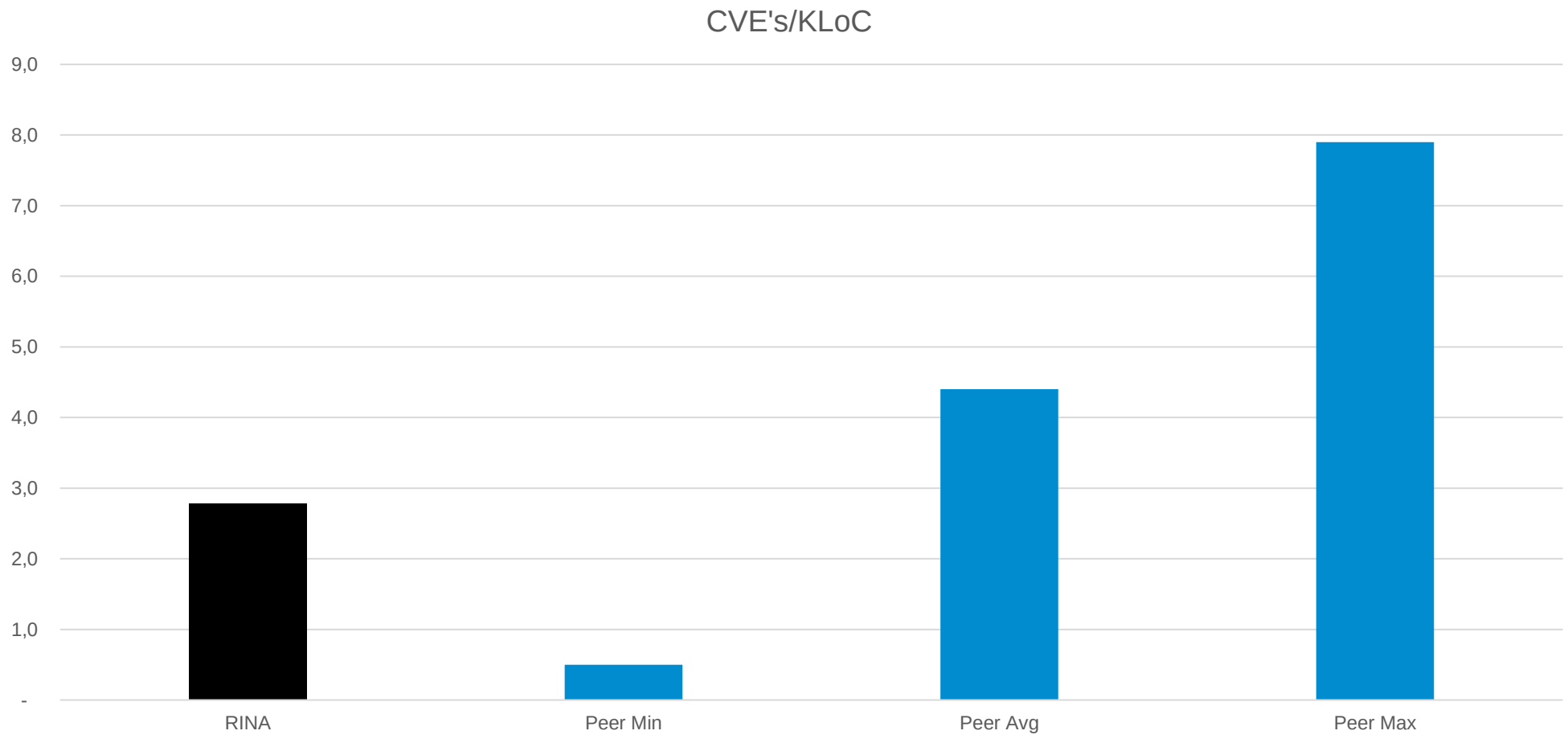
Step 4. The formula for Technical Debt:

- L = Number of Low-Severity Violations per KLOC
- M = Number of Medium-Severity Violations per KLOC
- H = Number of High-Severity Violations per KLOC
- S = Average Application Size (KLOC)
- C = Cost to Fix a Violation ($ per Hour)
- T = Time to Fix a Violation (Number of Hours)

Technical Debt per Application = [(10% * L) + (20% * M) + (50% * H)] * C * T * S

Using Appmarq data to arrive at **the values for L, M, H, and S, the amount of Technical Debt in a typical business application of 374 KLOC is over $1 Million.**

CVE Benchmark



metri
IT STARTS WITH THE FACTS

Documentatie

IT STARTS WITH THE FACTS.

metri

IT STARTS WITH THE FACTS

Documentatie

- Documentatie is uitgebreid
- Up-to-date: over het algmeneen RINA versie 6.2.3
- Paar uitzonderingen (versie 6.2.1)

Folder	Documentnaam	Type document	RINA Versie	# Blz
Deployment_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - Installation and Configuration - UBUNTU	Installatie/Configuratie handboek	6.2.3	30
Deployment_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - Installation and Configuration - WINDOWS	Installatie/Configuratie handboek	6.2.3	36
Deployment_Documentation	EESSI 2020 HF1 - RINA 6.2.3 (BMIWS) - Installation and Configuration - UBUNTU	Installatie/Configuratie handboek	6.2.3	30
Deployment_Documentation	EESSI 2020 HF1 - RINA 6.2.3 (BMIWS) - Installation and Configuration - WINDOWS	Installatie/Configuratie handboek	6.2.3	30
EESSI_2020_documentations	EESSI - CDM - Functional Specs - SBDH pattern of SedSchemaVersion - CR 4055	Change request		6
EESSI_2020_documentations	EESSI - CDM - Functional Specs - Update enum lists (Other Unknown) - CR 5895	Change request		8
EESSI_2020_documentations	EESSI - Functional Specifications - CDM Synchronisation	Change request		57
EESSI_2020_documentations	EESSI - CDM - S_BUC_01 Mainproc Specification - CR 5944	Business case		16
EESSI_2020_documentations	EESSI - Functional Specifications - IR Synchronisation - CR 3967	Change request		125
EESSI_End2End_test_cases	AW_BUC_01a_TestCase_000001.xlsx t/m UB_BUC_04_TestCase_001235.xlsx	1362 Testcases		1362
EESSI_structural_enhancements_activities	EESSI - TA 2020 - CDM Improvements - BUC version Lifespan - Summary & Conclusions	Common Data Model - ppt		
EESSI_structural_enhancements_activities	EESSI - TA2020 - CDM improvements - Simplification and Optimisation	impossible to open		
Operations_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - Deployment Guidelines	Deployment handboek	6.2.3	26
Operations_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - DMT 2.1.0 - Operations - Data Migration Guidelines	Data Migratie handboek	6.2.3	10
Operations_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - Operations Manual	Operations handboek	6.2.3	59
Operations_Documentation	EESSI 2020 HF1 - RINA 6.2.3 - User Manual	Gebruikershandboek	6.2.3	132
Operations_Documentation	EESSI 2020 HF1- RINA 6.2.3 - Administration Manual	Administration handboek	6.2.3	106
Operations_Documentation	Migrate-Holodeck-Database-RINA2019toRINA2020	Zipfile met technische migratiefiles		
RINA 2020 Architecture Documentation	EESSI - RINA - Functional Specs - Administration Portal	Functional Specs - Administration Portal		183
RINA 2020 Architecture Documentation	EESSI - RINA - Functional Specs - Case Management	Functional Specs - Case Management		225
RINA 2020 Architecture Documentation	EESSI - RINA - Functional Specs - Forms design	Functional Specs - Forms design		29
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Architecture Overview	Architecture Overview		86
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Archiving Specifications - rev01	Archiving Specifications	6.2.1	25
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - BUC Technical Specifications	BUC Technical Specifications		75
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Business Messaging Interface (BMI)	Business Messaging Interface (BMI)		109
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Business Use Case Engine Architecture	Business Use Case Engine Architecture		26
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Case Processing Interface (CPI) - Reference	Case Processing Interface (CPI) - Referenc	6.2.1	253
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Case Processing Interface (CPI) - rev01	Case Processing Interface (CPI) - rev01		74
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Identity and Access Management (IAM)	Identity and Access Management (IAM)		46
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - Localisation Specifications - rev01	Localisation Specifications		21
RINA 2020 Architecture Documentation	EESSI 2020 - RINA - National Information Exchange Interface (NIE)	National Information Exchange Interface (NIE)		50
RINA 2020 Architecture Documentation	EESSI 2020 - RINA 6.2.1 - SQL DB Schema - Pysical Data Model	SQL DB Schema - Pysical Data Model	6.2.1	205
RINA 2020 Architecture Documentation	EESSI 2020 - RINA 6.2.1 - SQL DB Schema - Figure	Gif file - Database schema	6.2.1	
RINA_Test_Reports	EESSI - RINA 2020 test campaign Final Test report_V1.0	Test Report final		12
RINA_Test_Scenarios_and_Test_cases	EESSI - RINA 6.2.1 - CPI and BMI - Test Framework	User manual		11
RINA_Test_Scenarios_and_Test_cases	RINA 2020 test cases	zip file with test cases		
RINA_Test_Scenarios_and_Test_cases	rinacpitest-master	zip file met applicatiecode		3463

Documentatie in de code

Module treemap

RULES	WEIGHT	# FAILED	# SUCCEEDED	% COMPLIANCE	INDUSTRY	GAP	SCORE
Documentation - Volume of Comments		**14,093**	**18,274**	**50%**	**67%**	**-17%**	**1.69**
Avoid undocumented Methods		9,643	4,964	33%	64%	-31%	1.00
Avoid undocumented Classes		202	2,208	91%	73%	18%	2.32
Avoid undocumented Interfaces		33	316	90%	60%	30%	2.11
Avoid undocumented Functions (Javascript)		5	2	28%	38%	-10%	1.00
Avoid undocumented Triggers, Functions and Procedures		4	0	0%	69%	-69%	1.00
Avoid programs with low comment / code ratio (Shell)		25	8	24%	71%	-47%	1.24
Avoid programs with low comment / code ratio (HTML5/Javascript)		0	8	100%	100%	0%	4.00
Avoid Methods with a very low comment/code ratio		3,909	8,619	68%	65%	3%	1.47
Avoid Classes with a very low comment/code ratio		263	2,147	89%	72%	17%	1.98
Avoid Functions having a very low Comment/Code ratio (Javascript)		5	2	28%	37%	-9%	1.00
Avoid triggers, functions and procedures with a very low comment/code ratio		4	0	0%	66%	-66%	1.00

Software Composition Analysis

Open Source risico's

CAST Consolidates one of the largest databases on software components, with billions of signatures (fingerprints) computed for each version of open source and third-party components. Based on this unique database Highlight compares your application files fingerprint and aggregates component, version, license, release date information at both portfolio and application levels.

Third-Party License Risk

1806	38	27	1734
Components	HIGH RISK	MEDIUM RISK	LOW RISK

Possible Vulnerabilities

17	53	36	4
CRITICAL	HIGH	MEDIUM	LOW

RINA - Cloud Boosters

Cloud Requirement	Technology
Using PostgreSQL database	Java

RINA - Cloud blockers

Cloud Requirement	Technology	Impact	Criticality	Roadblocks
Hardcoded URLs using HTTP protocol	Java	C	Low	60
Hardcoded URLs using LDAP protocol	Java	C	Low	1
Perform Directory Manipulation	JavaScript	CFA	Medium	3
Perform File Manipulation	Java	CFA	Medium	198
Use of an unsecured data string	Java	CA	High	16
Use of unsecured network protocols (HTTP)	Java	C	Low	20
Use of unsecured network protocols (HTTP)	Typescript	C	Low	10
Use of unsecured network protocols (HTTP)	JavaScript	C	Low	11
Use of unsecured network protocols (HTTP, FTP)	Java	C	Low	20
Use of unsecured network protocols (HTTP, FTP)	Typescript	C	Low	10
Use of unsecured network protocols (HTTP, FTP)	JavaScript	C	Low	11

RINA – Cloud blockers

⚠ Cloud Requirement	Technology	Impact	Criticality	⧗ Roadblocks
Using Access Control List	Java	CFA	Critical	1
Using file system	Java	CFA	Medium	3
Using hardcoded network IP address (IPV4, IPV6)	Java	C	Low	1
Using stateful session (Spring)	Java	CFA	High	1

About CAST

IT STARTS WITH THE FACTS.

metri

IT STARTS WITH THE FACTS

About CAST and it's Ecosystem

CAST technology delivers **key insight** about **software health, risk, functional & technical sizes**, and **ADM performance of internal or outsourced teams**; based on automated analysis of application source code.

→ Better decisions, lower risk, increased management control and higher ADM effectiveness and efficiency.

CAST BY THE NUMBERS

- Almost **$180M** investment in R&D
- **250+** customers worldwide
- **25+** years of software analytics experience measuring some of the most complex IT systems in the industry
- Traded on Euronext Paris
- Offices in US, Europe, India, China

Gartner COOL VENDOR — CAST Named "Cool Vendor" by Leading Analyst Firm

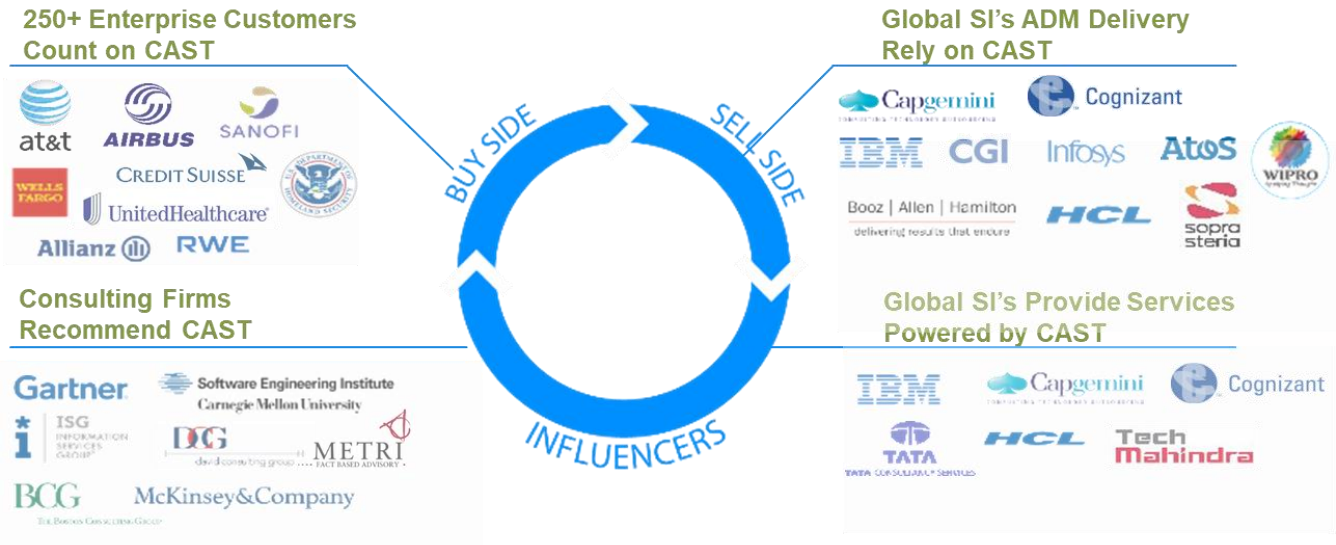
UBM — Editor's Choice Award: A Top-10 Company to Watch

Gartner — "CAST is **the de facto standard** for measuring quality and productivity of application services"

OMG — "CAST is **at the forefront** of standards adoption for **robustness, security, maintainability, and automated function points from code**"

IDC — "CAST is **the leader in the business IT space**"

OVUM — "CAST is the **leading technology** of its kind"



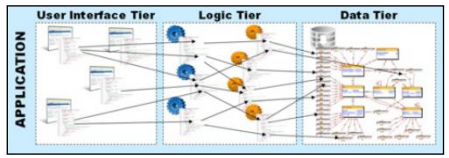
How it works

The CAST Application Intelligence Center

Measurement Platform

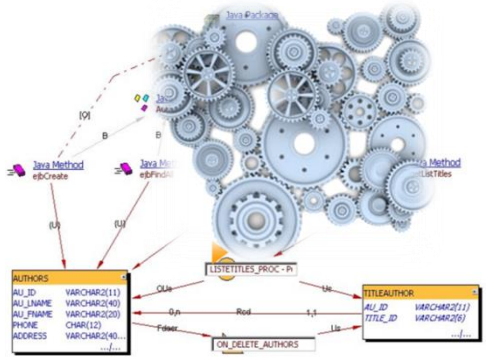
CAST Data

Engineering Platform



AIP Analytical Engine

- Quality Measurement & Functional Sizing
- Compliance checks
- Code, Architecture and Data Structure Analysis



Actionable visibility across the IT organization

Reports on application health, team performance, benchmarking

CFOs, CIOs and VPs

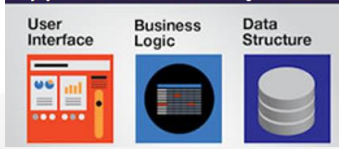
Software characteristics, cost and risk drivers, ARB, architectural governance, root cause analysis…

PMs, QA, Architects

… and feedback & advice on software quality and engineering

Development Teams

App Archi Discovery Doc



Application Source Code

- CICS, IMS, COBOL, DB2 z/OS
- J2EE, .NET and all Major RDBMS
- Web Apps, BI, EAI, C/C++, VB, PB
- Siebel, SAP, PSFT, OBS, Amdocs

metri
IT STARTS WITH THE FACTS

About CAST AIP

Software Intelligence Pioneer & Leader

$155 million R&D investments

25 years and counting

Global presence
US–EU–INDIA–CHINA

CISQ Consortium for IT Software Quality

Software Engineering Institute

Hundreds of Enterprise Customers

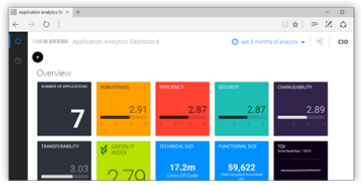
BOEING · Allianz · BNY MELLON · at&t · KAISER PERMANENTE · BMW

ISVs and Global SI's Customers

Infosys · IBM · Capgemini · Atos · CGI · Broadridge · accenture · wipro · DXC.technology · Cognizant · Tech Mahindra · sopra steria · HCL

CAST APPLICATION INTELLIGENCE PLATFORM

Health Dashboard

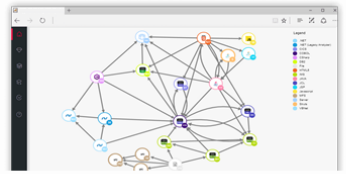


- Software health and security measures and trending
- Tech debt and software sizing

Engineering Dashboard

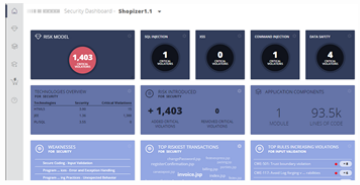
- Software flaw identification
- Engineering best practices
- Remediation planning

Imaging System



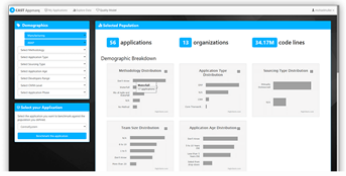
- Blueprinting
- Application discovery

Security Dashboard



- Security and risk identification
- Secure engineering best practices
- Remediation planning

Appmarq



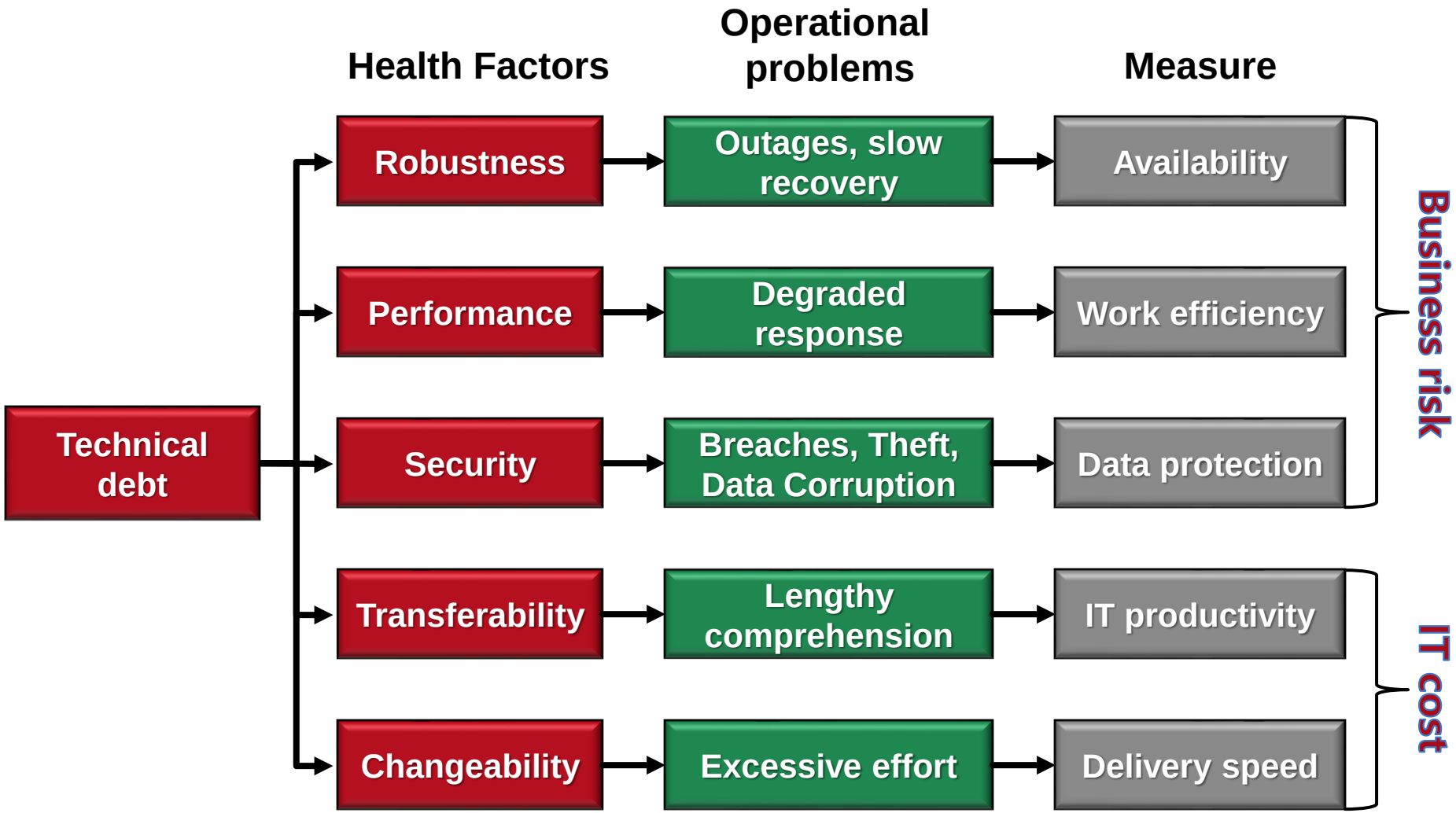
- Industry, tech & geo benchmarks
- +2500 apps, +3B LOC
- Custom benchmarking

Enlighten

- Interactive exploration
- Impact Analysis
- Automated Documentation

metri
IT STARTS WITH THE FACTS

Health factors and risk/cost



metri
IT STARTS WITH THE FACTS

Health Factors explained

	Description	Business value
Transferability	Measurement of the effort needed to transfer knowledge and ownership of the application to a new team either external or internal or to integrate new team member in the existing team.	▪ Avoid to be tied to a internal resource / team or outsourcer ▪ Improve team productivity ▪ Ease transfer between contractors, internal teams and outsourcer
Changeability	Effort measurement to implement a fix or a new feature within the application.	▪ Improved maintenance ease and delays ▪ Improved predictability of application releases ▪ Improve time to market
Robustness	Measure the robustness of the application and the risk to introduce instability during code maintenance or development.	▪ Reduce defects and bugs in production ▪ Lower the application downtimes ▪ Improve User Experience
Efficiency	Measurement of the risk of bad performance of the application based on its design and architecture	▪ Improve response time of the application ▪ Lower resources needs of the application ▪ Improve scalability
Security	Measurement of the risk whether an application can have possible security breach and how its data are protected	▪ Improve security of both the application and the critical business data used
Maintainability (TQI)	Appreciates the cost and ease to globally maintain an application in the future.	• Lower general maintenance costs of the applications

IT cost

Business risk

metri
IT STARTS WITH THE FACTS

Transferability

Based on Norm **ISO/IEC 25010** (initially out of ISO/IEC 9126) : Ease of analysis

- Measure the ease how :
  - A new team can take responsibility of an application
  - A development team can integrate a new member

This measure rely on:

- Documentation (code/comments ratios, automated documentations, commented code…)
- Complexity (Mc-Cabe complexity, SQL complexity, control structures…
- Links management between application components
- Dead Code
- Respect of commons development norms and standards (code structures, naming conventions…)
- Number of components and sizing

Changeability

Based on Norm **ISO/IEC 25010** (initially out of ISO/IEC 9126) : ease of change

- Appreciate the effort needed to perform evolutions in software code

This measure rely on:

- Respect of architecture rules (inter layer calls, links between components, code independence from the underlying platforms and OS)
- Complexity (Mc-Cabe complexity, SQL complexity, control structures, imbrication and deepness …)
- Copy and Pasted code
- Code factorization and reuse
- Dead Code
- Respect of commons development norms and standards (code structures, naming conventions…)
- Documentation (code/comment ratios)

Robustness

Based on Norm **ISO/IEC 25010** (initially out of ISO/IEC 9126) : Stability and ease of test

- Measure the testing effort needed to validate an application
- Measure risk of production failure

This measure rely on:

- Respect of architecture rules (inter layer calls, components linkage, code independence from the underlying platforms and OS)
- Complexity (Mc-Cabe complexity, SQL complexity, control structures, code imbrication and deepness)
- Dead Code
- Code Structures and language paradigms use
- Error management and Exception handling
- Secure coding, time and state
- Application sizing and number of components

Efficiency

Measurement of the static performance of the application.

- Evaluate the risk to meet performance issues coming from application code

This measure rely on:

- Persistent data handling strategy and performance
- Persistent data design
- Dynamic calls and instantiation use
- Expensive call and pattern (resource wise, especially in loops)
- Memory, network and disk space management

Security

Evaluate the risk of security breach in the application as well as how secure data is managed inside and outside the application.

This measure rely on:

- Respect of architecture rules (inter layer calls, components linkage, code independence from the underlying platforms and OS)
- Persistent data handling strategy and performance
- Unexpected behavior risk
- Error management and Exception handling
- Secure coding (time and state, input validation, parameters control thru call chain)

Over Metri

IT STARTS WITH THE FACTS.

metri

IT STARTS WITH THE FACTS

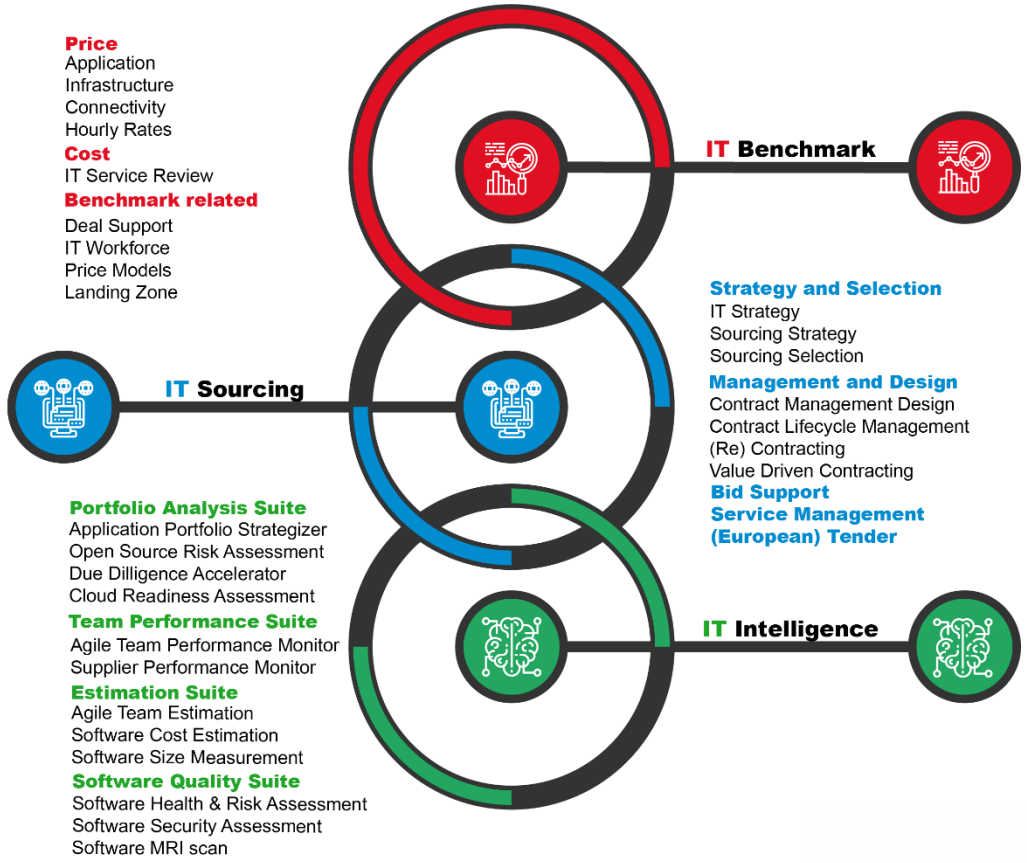
Over Metri

Metri, opgericht in 2003, is een onafhankelijk adviesbureau op het gebied van IT-benchmarking, IT-sourcing en IT Intelligence die op feiten gebaseerd, gedegen advies biedt, ter ondersteuning van innovatie en het creëren van bedrijfswaarde.

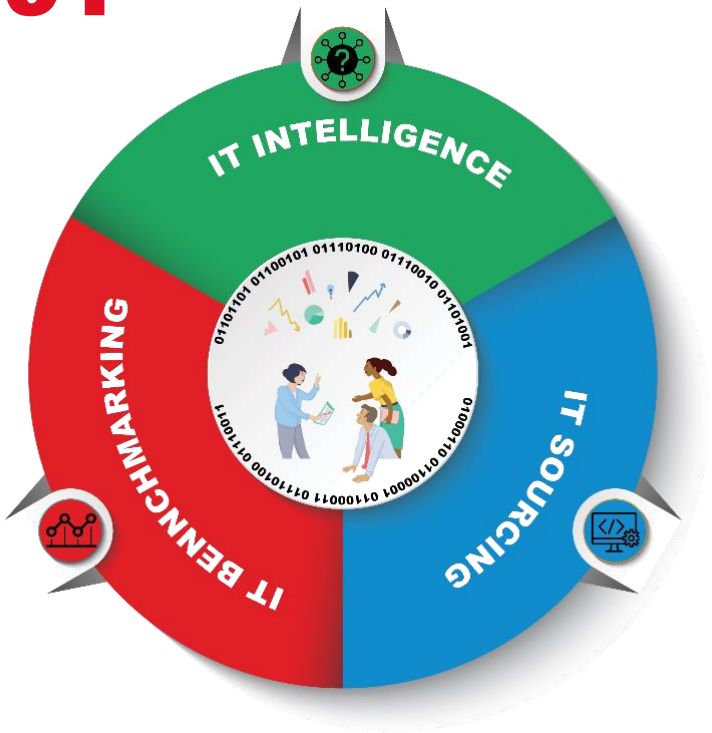
Benchmarking en feiten zijn ons DNA. Klanten waarderen onze expertise en ondersteuning bij sourcing-, governance- en innovatiebeslissingen door scenario-planning en business case-analyse.

Dankzij onze op feiten gebaseerde, bottom-up benadering kunnen onze klanten hun IT-doelstellingen realiseren. Metri is sinds 2021 onderdeel van het wereldwijde IDC concern.

Metri. Een effectieve combinatie van facts & mensen.



Metri Services



IT Intelligence

Portfolio Analysis Suite

- Application Portfolio Strategizer
- Open Source Risk Assessment
- Due Diligence Accelerator
- Cloud Readiness Assessment

Team Performance Suite

- Agile team Performance Monitor
- Supplier Performance Monitor

Estimation Suite

- Agile team Estimation
- Software Cost Estimation
- Software Size Measurement
- IT Business Case Support

Software Quality Suite

- Software Health & Risk Assessment
- Software Security Assessment
- Software MRI Scan

IT Sourcing

(European) Tender

Strategy

- IT Strategy
- Sourcing Strategy
- Sourcing Quick Scan

Sourcing selection

- Landing Zone
- Contracting
- Value Driven Contracting

Transition & Transformation

Bid Support

Contract Lifecycle Management

Contract Management organization

Service Management

IT Benchmark

Price Benchmark

- Application Services
- Infrastructure Services
- Connectivity Services
- Hourly rates

Cost Benchmark

- IT Services Review
- IT Workforce benchmark
- Landing zone

Deal support

IT STARTS WITH THE FACTS

Disclaimer