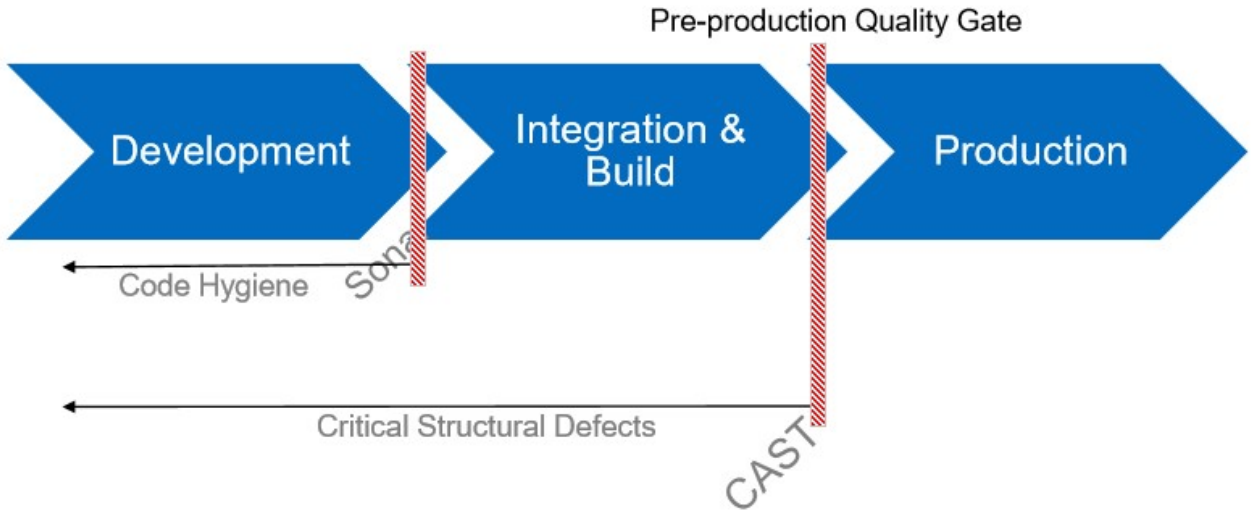
CAST SONAR

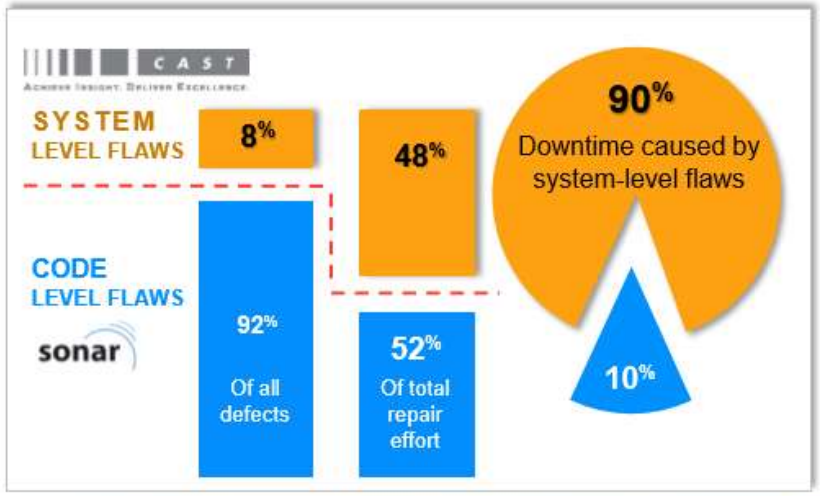
CAST and SONAR are Complimentary

- A vast majority of CAST clients are also Sonar users
- Often client's consider the two solutions to be competitors, but actually they serve very different purposes in the software development life cycle.
- Sonar is a code hygiene tool used by developers which focuses on maintainability and changeability. It is not meant to identify risks that could cause security breaches, slow performance, or critical failures in production.
- CAST acts as a pre-production quality gate, specifically focusing on finding critical structural defects to address software risk.

Example of Customer Deployment of CAST with Sonar in Dev Cycle



Quality Model Evolution – from Lines of Code to Systems



Trends and What CIOs Tell Us

- Sonar used early as developer tool for first move towards quality. Focus is on developer code hygiene.
- CAST added as quality programs matured to support four key uses:
  - System-level flaw detection
  - Robust pre-production quality gates
  - Accurate measurement including function points to enable decision-making
  - Technology coverage to address vast majority of application components

Nascent — Quality Model Evolution — *Mature*

Sonar

- File-level analysis provides entry-level insight for developer
- Rules focus on code hygiene versus detecting critical defects
- No calculation of productivity measurements (automated function points)
- Limited supported technology coverage
- No scope control (can create spotty app coverage)

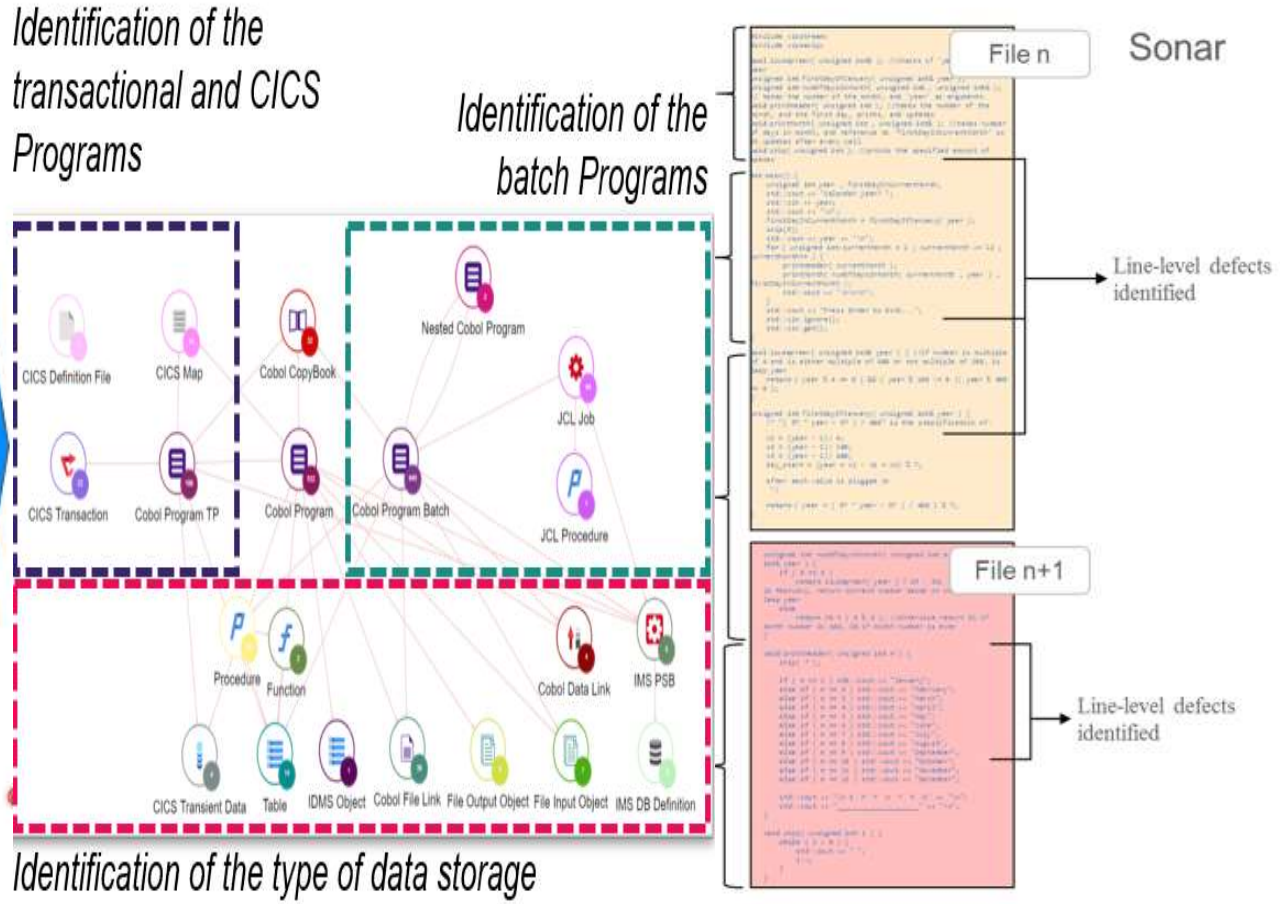
CAST AIP

- System-level view finds the most critical and hardest to detect defects
- System-level delivers accurate and repeatable measurements of technical debt, function points, and application quality
- Breadth of technology coverage delivers analytics for all of your applications from user interface to database

System-Level vs. Line-Level Code Analysis

- Fundamental difference between CAST and Sonar:
  - CAST is **System** level analysis
  - Sonar is **Component/file** level analysis
- System-level flaws represent:
  - 8% of all defects
  - 48% of total repair effort
  - 90% of downtime

  Note: See appendix for system-level flaw list



CAST Vs SONAR

CAST

Features	CAST	Sonar	Comments
System Level Analysis			AIP performs **inter-tier/ inter-technology** analyses. SONAR performs one technology at a time
Application Blueprinting			CAST has a unique capability to image an entire multi -tier application - **Imaging System**
Sizing & Estimation – Function Points			CAST provides AFP and AEP metrics using **Function Points** – IPFUG standards
Support for multiple technologies			SONAR – 25(Among them many are supported by the open source community) CAST – 50+ including older languages, frameworks, middleware, ERPs, mainframe, etc. + 30 additional technologies provided by CAST Extend
Support for industry Quality and Security Standards			CAST gives a System level perspective starting from the UI to End points(like Database, Webservices etc). It supports the **CISQ** standards along with others such as **OWASP Top 10, CWE**, NIST, STIG, OMG
Architecture Rules			Native architecture rules can be augmented with **Architecture Checker** Architects can monitor compliance using the checker. Custom rules can be added.
Quality Benchmarking			CAST **AppMarq** DB has a unique application benchmarking capability, across industries, apps type, technologies. The details are available in the Health Dashboards.
Portfolio level assessment and Reporting			Sonar has limited flexibility into arranging applications as portfolios for Senior Management/Executive level
DevOps Integration			Seamless integration to the DevOps pipeline
IDE integration			SONAR provides IDE plugins.
Cross-technology Transaction Mapping/ Tagging Sensitive Data			CAST can map a transaction path from user input to database. This is crucial to ensure **application security**
Remediation plans management & prioritization			CAST provides an **Action Plan Optimizer** along with risk metrics**. Continuous Improvement** graphs can be accessed by the teams.
Compares versions			**Snapshot comparison** can be done with CAST where added/modified and deleted violations and code can be identified.

CAST ECOSYSTEM

- Analysis done by a large European bank

	Scope			Quality/Health Measurement								Size measureme			Type of Measurement					Analysis & Reporting									Supported technologies										
	C-Level Management Dashboard	Application Portfolio Mgt	Day-to-day operational tool	Transferability	Changeability	Robustness	Security	Performance	Maintainability Index	Technical Debt	Compliance to Industry standards	Automatic FP	Lines of code	Complexity	Code level	Cross Component	Cross technology	Framework analysis / transaction	Architecture & Impact analysis	Drill-down option	False Positive rate	Remediation Recommendations	Prioritization model	Application Benchmarking	Action Plan	Impact Analysis, dependency views	Role based access	History and Trending	JAVA	Java Frameworks	.NET	COBOL	C/C++	Objective C	Oracle	SQL Server	DMS	Nr of supported technologies
CAST AIP	●	●	◔	●	●	●	◕	●	●	●	●	●	●	●	●	●	●	●	◕	●	◔	●	●	●	●	●	●	●	●	●	●	●	●	●	●	◑	40
HP Fortify	○	◑	●	○	○	○	●	○	○	○	●	○	○	○	●	◔	○	◑	○	○	◔	◕	◕	○	○	○	○	○	●	●	●	○	●	○	●	●	○	20
MIA Semantor	○	○	◑	○	○	◑	◑	◑	○	○	○	○	○	○	●	○	○	◕	○	○	●	○	○	○	○	◑	○	○	◔	◔	◔	●	○	○	○	○	●	
SonarQube	○	◔	●	◑	◑	◑	○	◑	○	◔	○	○	◑	◑	●	●	○	○	○	●	◑	◔	◔	○	◔	○	○	◑	●	○	●	●	●	○	○	○	○	21

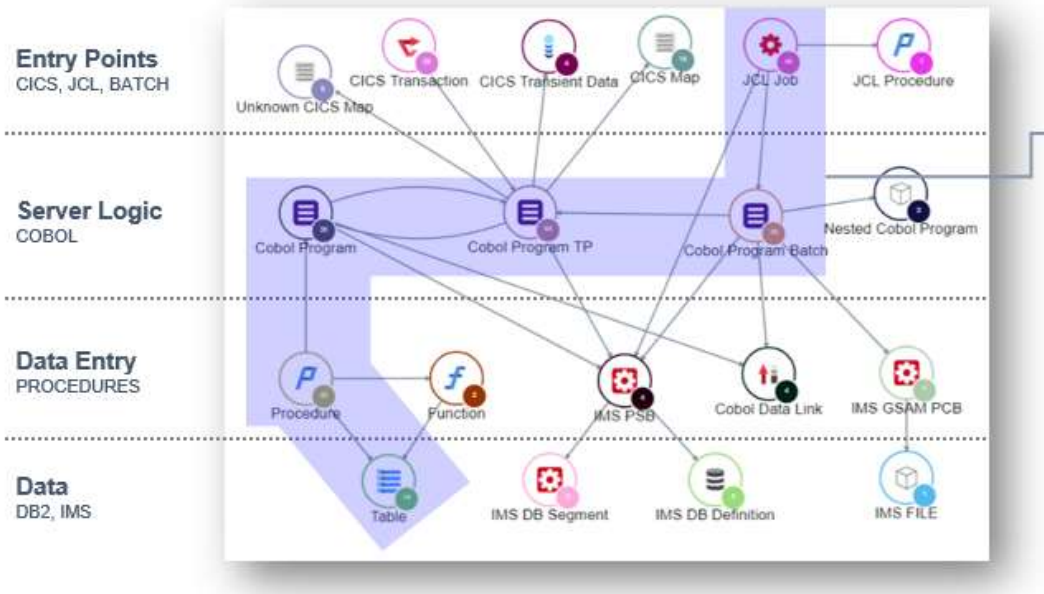
Legend ● = Exemplary ◕ = Good ◑ = Adequate ◔ = Inadequate ○ = Poor

CAST Solution

A complex multi-technology system requires multi-level views for understanding and analysis.

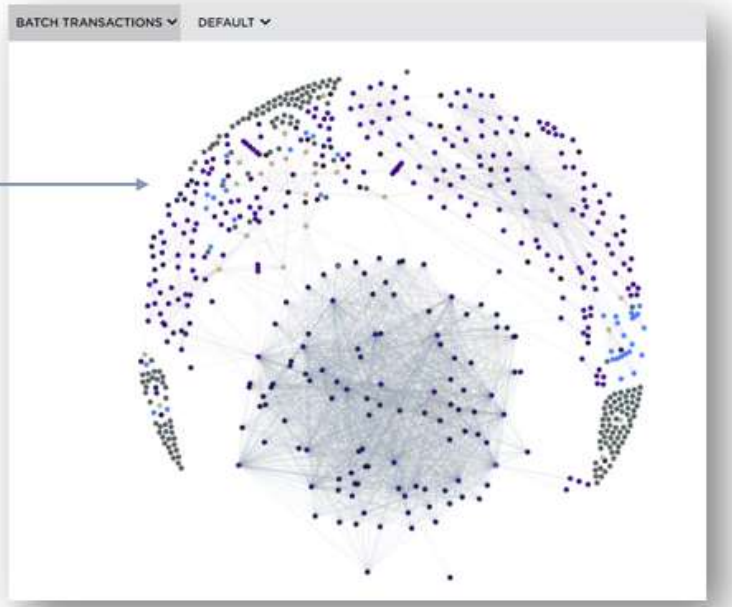
Architectural Abstraction

A starting point for determining enhancement or modernization approach to multi-technology systems



Transactional Deep Dive

Object level visibility of a transaction helps engineers validate their understanding and new engineers navigate complex logic and dependencies.



System Level - Holistic Approach

Comprehensive Software Intelligence Solution

Portfolio level Actionable metrics on following health measures

- ✓ Seeking visibility into risk of software quality, security vulnerabilities, and technical debt

Identify Areas of Impact

- ✓ Provides a snapshot of the application portfolio and highlights risk, complexity, as well as cost information through code analysis

Caters to a wide audience

- ✓ IT Executives – Seeking visibility into risk of software quality, security vulnerabilities, and technical debt Architects/Developers – Engineering Dashboard
- ✓ Architects – To ensure compliance to overall architecture and code reviews
- ✓ Development teams – For defect identification and resolution

Deep Dive interception capabilities

Software Intelligence by performing Contextual Analysis at System-Level- A data-based digital image

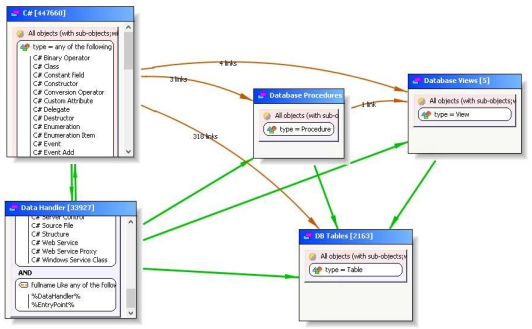
- ✓ Unlike code analyzers that focus on unit level analysis, the CAST Platform analyses the overall application across UI, middleware and database layers to find structural flaws
- ✓ CAST evaluates system architecture and identifies inefficient and performance taxing transactions across different technologies which is almost impossible to discover using traditional unit level analysis
- ✓ System-level view finds the most critical and hardest to detect defects

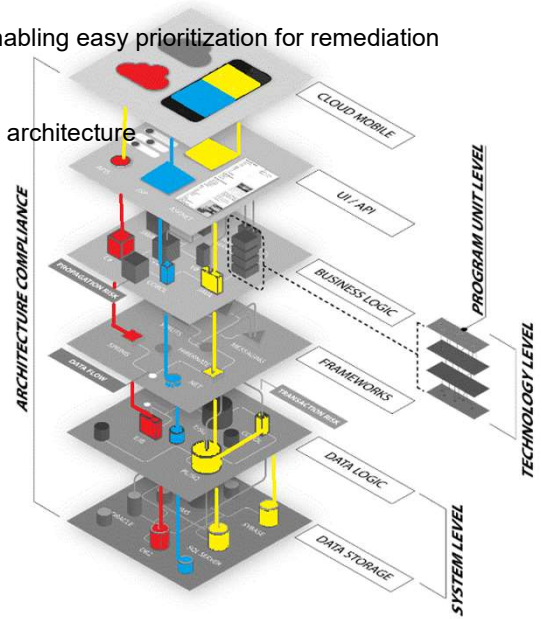
Propagated Risk Analysis

- ✓ Identifies top riskiest objects and transactions, enabling easy prioritization for remediation

Gain Control of your Architecture

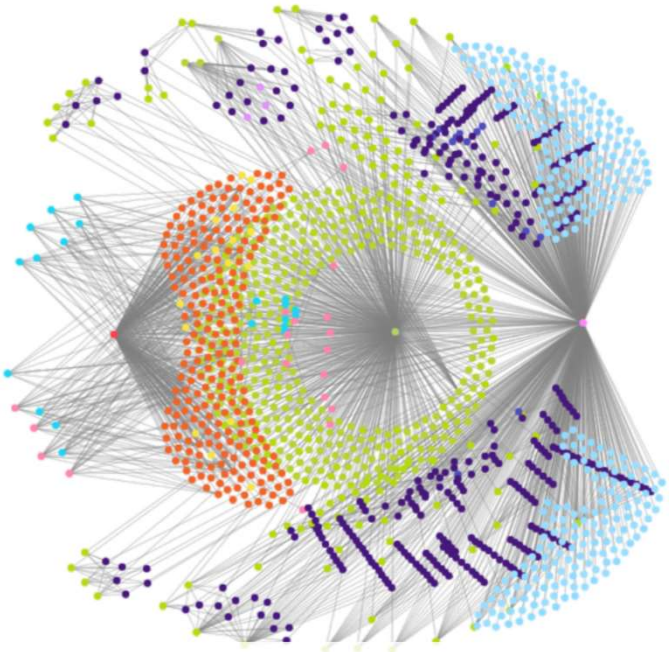
- ✓ Measure development adherence to the targeted architecture
- ✓ Automatically discovery of non-compliant links





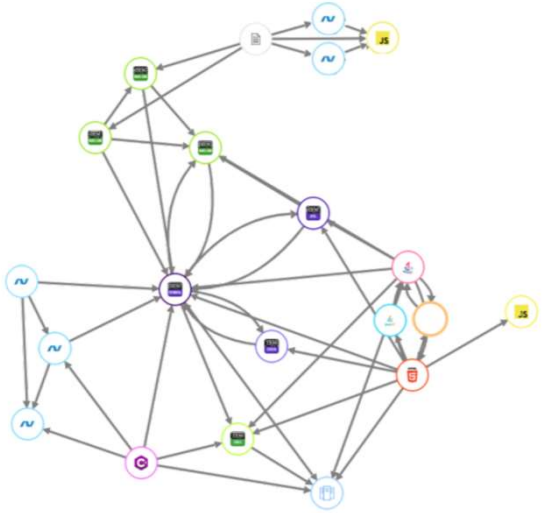
Uniqueness in Mapping Dependencies

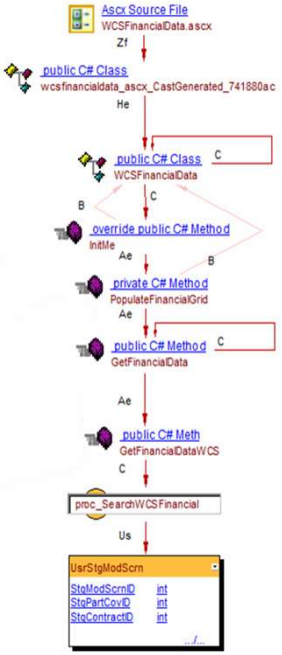
CAST understands your Architecture, inter-connections and dependencies



- ✓ CAST static analysis identifies application defects, design flaws and potentially malicious code by **blueprinting system as a WHOLE** in comparison to other dynamic analysis which monitors the app behavior, system memory & CPU response time
- ✓ View end-to-end **cross-technology dependencies** between components
- ✓ **Breadth of technology** coverage delivers analytics for all of your applications from user interface to database

Intelligently organize complex applications into logical application layers





- ✓ Contextual code analysis examines how **all components interact** with each other and across technology layers
- ✓ CAST capability identify **End-to-End Transactions** and architectural risks which may impact stability, performance & security
- ✓ CAST **Analytics solution** – Provides rich visualization, data mining capabilities, actionable insights

The Bottom-line

CAST

Salient Features: Software analysis and measurement platform that addresses entire Application Quality

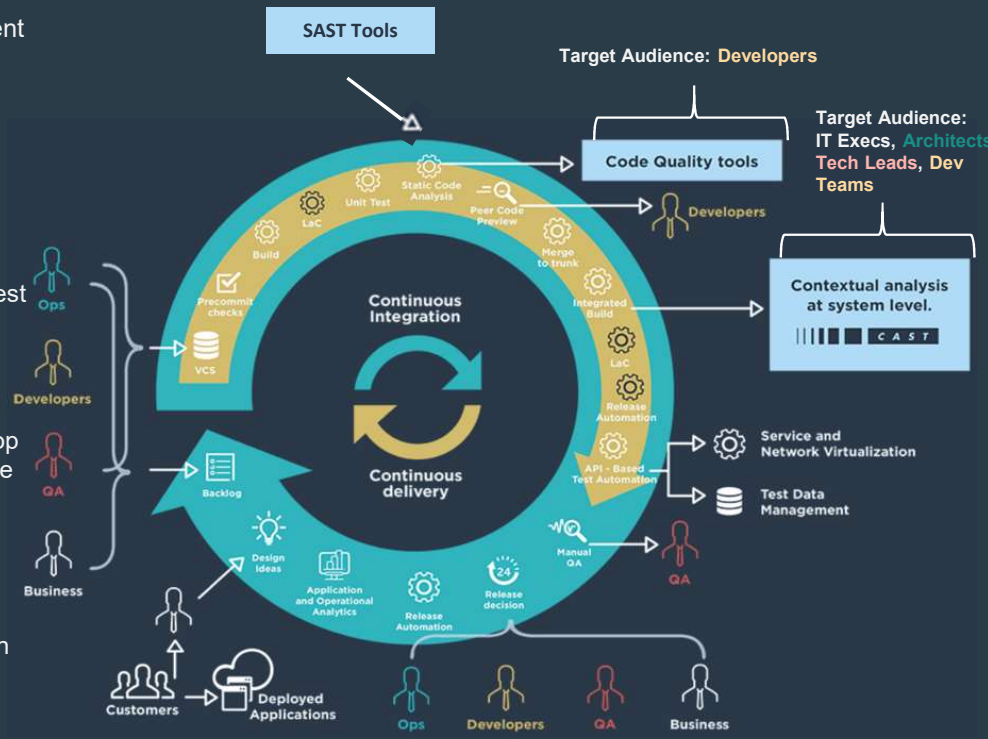
- ✓ Understands component dependencies across technologies and layers
- ✓ Holistic view of system that pinpoints critical defect
- ✓ Reverse engineers systems to create visibility into system
- ✓ Ensures architectural, structural and programming best practices

Depth of Coverage

- ✓ Analyzes across tiers of complex applications from top to bottom – UI layer to data layer – at the source code level
- ✓ Multi-Technology Interactions -Not just the multiple language approach but understands inter and intra language code components.
- ✓ Multi-Tier Interactions: Analyses interactions between code components from different tiers of application stack

Breadth of Coverage:

- ✓ Breadth of technology coverage delivers analytics for all of your applications from user interface to database
- ✓ Covering nearly 50+ languages and 12+ database technologies.



Code Quality/ SAST Tools

Salient Features: Most of code analysis tools are developer tool that addresses Code Quality at the language/program level & Analysis occurs locally at developer workstation

- ✓ Unit/program level analysis
- ✓ These tools are very focused on code hygiene with only a few controls between components of the same technology

Depth of Coverage

- ✓ Unit/program level analysis
- ✓ Ensures basic programming best practices
- ✓ Checks for good code hygiene, readability and cleanliness
- ✓ Code level view that flags poor coding hygiene and non-conformance to coding standards

Breadth of Coverage:

- ✓ Most of the tools have limited coverage of technology (Technology coverage matters-specially at Organizational level)

Case Study – Expansion opportunity of applications for a large BFSI Client in Europe

Client

A large multinational SI with large footprint in Europe

- 500+ ADM professionals working for a BFSI Client account in distributed ODC (Near- and Off-shore) with expansion opportunities
- License purchase agreement in Q2 2016 with a scope of 100 FTE (Upgraded to ELA in Q3 2017)
- Business goal is to run major quality initiative to sustain confidence with Client, monitor Transition-in risk and secure margins for newly inherited scopes

Context

Major Objectives behind CAST Software Intelligence adoption-

- Reduce transition time and improve knowledge transfer
- Improve overall code quality
- Benefits from early detection of probable production defects
- Engage with fact based communication with Client
- Help to measure, manage and reduce the technical debt

Solution

- Deployed CAST Software Intelligence on selected risk exposed applications; with periodic meetings with Service Managers to sustain adoption and gather feedback
- Proceed to a fully automated analysis from source code retrieval (file system, Clear Case, Git) to report generation
- CAST Action Plan Optimizer leveraged by CoE - Remediation plan proposed, centered on business priorities & budget

Scale & Coverage

Deployment started in Q3 2016

- CAST AIP socialized across multiple LOB within client ecosystem
- CoE Team in France and Morocco

Status as of Q1 2018

- AIP deployed for 35 apps during and after Transition
- 20+ MLOC analyzed with 10+ technologies and frameworks
- Several snapshots analyzed in some cases, also *Automated Function Point* count was enabled for 2 applications

Outcome

- The value of CAST Software Intelligence insight was acknowledged by application teams, despite usage of SonarQube at unit level.
- Actionable insights enabled fact-based discussions with client; hence, gaining more transparency and confidence with Client. For example, justifying the development of a new framework from scratch *vis a vis* rewriting the existing code; and hence saving 100 K€ by avoiding the rewriting of an existing custom framework, common to different applications.
- Highly satisfied account team and CoE contributing to solutioning RFPs - expansion opportunities with Client.

"CAST rescans are now made mandatory at each major release (quarterly).

CAST is also mandatory to drive the quality for all transitions within various entities at client."

- SI Service Manager

Case Study – CAST in outsourcing strategy towards software health improvement & industrialized global delivery

Client

Big 10 Sourcing Player in NA

- 70,000+ professionals
- USD 10B+ yearly revenues
- 100,000+ clients across the globe
- ELA signed with CAST, and deployment started in April 2016 with twofold scope
  - IP Solutions (Suite of products) and
  - End-Customer applications

Context

Build a Software Intelligence CoE focused on –

- Productivity improvement on large outsourcing engagements via detailed analytics – cost, quality & innovation capacity
- Continuous improvement and innovation (DevOps and Automation) for IP Solutions
- Margin improvements by automated quality gates
- Leveraging CAST for output-based Managed Services engagements

Solution

CAST helped client build, staff & operationalize CAST Global Software Intelligence CoE based out of India

- Initial focus on IP – improve product robustness & health at system level
- Other tools were part of the Agile framework incl. SonarQube for unit testing at developer level
- Later, CAST was embedded into outsourcing strategy and CAST Highlight leveraged during pre-sales (*Due Diligence* of application portfolio)

Scale & Coverage

- The program started with pilot of 4 critical apps
- Over next 16 months the app portfolio grew to 193
- Analytics and Engineering dashboards user base grew up to 2,000+ users

Snapshot of Scale (as of September 2017) –

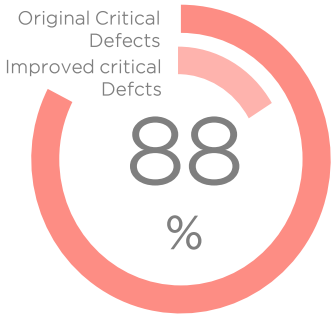
- 193 apps out of which 60 had frequent re-scans
- 122 IP Solutions and 37 End-Customer applications under ADM engagement

Outcome

Significant improvement observed during 12 months for two proxy business critical applications

- Critical defects and rework reduced by 80+%
- Knowledge Transfer time reduced by 25% from 4 to 3 months
- Defect density reduced by 50%
- Improved review effectiveness – Code reviews went from style and formatting to functional scope, requirements and design reviews

Critical Defects Impact



Defect Rework Time Impact