



## ■ XRadar

Open source-verktøy for kode- og kvalitetsanalyse

Kjetil Jørgensen-Dahl, NOS Clearing ASA  
Rodin Lie, NOS Clearing ASA

Kristoffer Kvam, Telenor asa



## ■ Teknisk gjeld

*Although immature code may work fine and be completely acceptable to the customer, excess quantities will make a program unmasterable, [...] Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. [...] The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise.*



Ward Cunningham, OOPSLA, March 26, 1992





## ■ Hvordan ser teknisk gjeld ut?

---

- Duplisering
- Dårlig navngiving
- Testproblemer
- Høy kopling
- Lav kohesjon / problemer med ortogonalitet / ett ansvar
- Unødvendig kompleksitet
- Brudd på konvensjoner (idiomer) og standarder
- Endringer må reflekteres mange steder
- Store/lange metoder, klasser, API'er
- ...



## ■ Refrigerator code / toilet code ☺

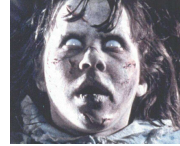
---

- **Refrigerator Code:**
  - *It's code that you're so proud of that you want to take it home and hang it on the refrigerator, right alongside of your children's drawings.*
- **Toilet Code:**
  - *It's code that's so mediocre that when somebody encounters it, they just want to flush it down the toilet.*



## ■ Hvordan motarbeide teknisk gjeld? (I)

- Google's approach(?)
  - Starte fra scratch
  - Superutviklere
  - En smidig prosess med stor frihet
  - Strengt krav til koden
  - Godt med review/parutvikling
  - Lite tidspress (leve av noe annet?)
  - Masse belønning for *a job well done*
- Svært sjelden mulig...
  - Har som regel en arv å ta med seg
  - Superutviklere er det underskudd på (=koster mye)
  - Kunden, brukeren, forretningsiden presser på
  - Microsoft Vista?



## ■ Hvordan motarbeide teknisk gjeld? (II)

- Mer realistisk – mange pågående tiltak:
  - Kompetansebygging
    - Idioms, Smells, Patterns, ...
  - Kulturbygging
    - Positiv feedback, - *Refactor mercilessly!*, - *No broken windows!*, ...
  - Metodikk/teknikk
    - Test-Driven Development, Pair Programming, prioritering, ...
  - Verktøy
    - IDEs, Refactoring, Inspection, Reporting, ...
  - Rammevilkår
    - Forankring hos ledelsen, Tid/ressurser til å investere, ...
  - ...
- Tenk entropi – *man må hele tiden tilføre energi for at systemet ikke skal bevege seg mot kaos*





## ■ Inspeksjons- og metrikkverktøy (ett tiltak)

---

- Finnes det informasjon man kan trekke ut av koden som vil fortelle noe om systemets kvalitet?
- Hypotesen er at det finnes en del indikatorer:
  - Testdekning (andel av koden dekket av tester)
  - Dokumentasjonskompletthet (Javadoc-mangler)
  - Avvik fra kodenstandard og standard idiomer
  - Kompleksitetsmetriker
  - Metriker for måling av kopleing
  - ...



## ■ Dersom hypotesen holder ...

---

... og man har et slikt verktøy:

- Kan påvise områder med potensiale for forbedring
- Benchmarking
  - mot andre prosjekt
  - mellom subsystemer/moduler
  - mot seg selv – over tid
- En rekke situasjoner
  - i egne prosjekter
  - i vurdering av open source
  - ved "audit" og "due dilligence"
  - når man overtar eller går inn i kode fra andre kilder
  - når man setter bort utviklingen til noen andre



**EDC Build Reports** Javadoc Unit Test Results Unit Test Coverage Unit Test Javadoc [xradar](#) Act Daily Act Nightly Act JavaDoc Act Coverage CosSource

**Subsystems**

- Listener
- Old Client API
- Old CRM Services
- CRM Services
- Billing Domain
- Customer Domain
- Order Domain
- Net Domain
- Product Catalog Domain
- Common Domain
- Application Mgmt Domain
- Infrastructure
- Integration
- Legacy
- Util

<http://xradar.sourceforge.net>

Static Report - System: COS, Version: EDL\_DEV, Date: 2005-03-07 - 21:13:09 Designed for use with JDepend, PMD, PMD-CPD, JavaNCSS, JUnit, JCoverage, Checkstyle, XSource, JavazHTML and Ant.

**[Statistics / Dynamics] overview**

**Scorecard (COS)**

This page gives a scorecard of this system's quality. The quality measure is defined for this specific system. Every metric has a value in the range from 0 to 1. 0 is the worst possible score, while 1 is the best. The formula for how the metric has been calculated is also shown. Press the subsystem graph to see scorecard details on the specific subsystem.

**Packages**

- com.telcelor.cos.adapter.calldb.ejb
- com.telcelor.cos.adapter.cosa.config.com
- com.telcelor.cos.adapter.customermaster
- com.telcelor.cos.adapter.customermaster
- com.telcelor.cos.adapter.dnnew.dbo
- com.telcelor.cos.adapter.dnnew.ejb
- com.telcelor.cos.adapter.dnnew.wrapper
- com.telcelor.cos.adapter.dnnew.wrapper.ejb
- com.telcelor.cos.adapter.fakturalutell
- com.telcelor.cos.adapter.geneva.ejb
- com.telcelor.cos.adapter.kurt.ejb
- com.telcelor.cos.adapter.online.ejb
- com.telcelor.cos.adapter.productcatalog

A grid of colored boxes representing subsystem quality scores. The grid is organized into two columns. The left column contains: Listener (orange), Old CRM Services (orange), Billing Domain (yellow), Order Domain (yellow), Net Domain (orange), Infrastructure (orange). The right column contains: Old Client API (orange), CRM Services (yellow), Customer Domain (yellow), Common Domain (yellow), Application Mgmt Domain (yellow), Integration (orange), Legacy (yellow), Util (green).

<b>Total Quality</b>	<b>[TQ= 0.35*TS + 0.35*ARCH + 0.3*CODE]</b>	<b>0.5</b>
<b>Test Suite</b>	<b>[TS= 1*TCU]</b>	<b>0.46</b>
<b>Unit Test Coverage</b>	<b>[TCU= source.statements.covered /ncs]</b>	<b>0.46</b>
<b>Architecture</b>	<b>[ARCH= 0.4*MOD + 0.6*COD]</b>	<b>0.46</b>
<b>Modularisation</b>	<b>[MOD= 1 - (count_packages(not@legal-dependencies==0)) /total_packages]</b>	<b>0.96</b>
<b>Cohesion</b>	<b>[COH= 1 - (count_packages(cycles==true) /total_packages)]</b>	<b>0.13</b>
<b>Code Quality</b>	<b>[CODE= 0.15*DDC + 0.4*DRY + 0.3*FRE + 0.15*STY]</b>	<b>0.6</b>
<b>Documentation</b>	<b>[DDC= javadocs /(functions + analysed-classes)]</b>	<b>0.9</b>
<b>DRYness</b>	<b>[DRY= 1 - (classes-with-duplications /analysed-classes)]</b>	<b>0.82</b>
<b>Freshness</b>	<b>[FRE= 1 - (classes-with-code-violations /analysed-classes)]</b>	<b>0.42</b>
<b>Stylehness</b>	<b>[STY= 1 - (classes-with-style-errors /analysed-classes)]</b>	<b>0.07</b>

**Classes**

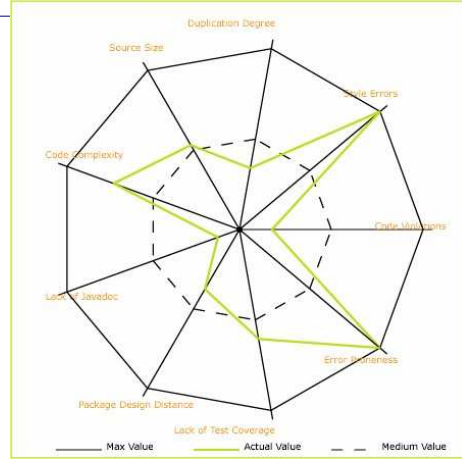
- \_J51ParamWrapperStub
- \_J512LoginWrapperStub
- \_J512WrapperStub
- \_KurWrapperStub
- \_MQISWrapperStub
- \_MQMINOWrapperStub
- \_MQNAWrapperStub
- \_MQNWrapperStub
- \_MQOrderConfirmationWrapperStub
- \_MQOrderPilotWrapperStub
- \_RINCardOrderWrapperStub
- \_RINJobQueueWrapperStub
- \_RINJobCodeProductWrapperStub
- \_RINJobCardWrapperStub

## Opprinnelig et beslutnings- og oppfølgingsverktøy for Pareto-prosjektet

- Strukturering av arkitekturen
- Identifisere problematisk kode (80/20)
- Kontrollere utvikling under og etter prosjektet
- Validere suksess!
- Etablere et skreddersydd perspektiv for viktige roller i systemets forvaltning

## ■ Bruk av XRadar

- Gir utviklerne en standardisert QA på den koden de implementerer
- Oversikt over kodebasen fra system via subsystem, pakke, klasse og metodenivå - og helt ned til enkeltlinjer.
- Gir kontroll over det som leveres inn i systemet
- Leder mot smartere strategiske investeringer
- Definerer systemets interne kvalitet



**EDC Build Reports** [Javadoc](#) [Unit Test Results](#) [Unit Test Coverage](#) [Unit Test Javadoc](#) [xradar](#) [Act Daily](#) [Act Nightly](#) [Act Javadoc](#) [Act Coverage](#) [CosSource](#)

[scorecard] [analysis] [explanations]

**Subsystems**

- Listener
- Old Client API
- Old CRM Services
- CRM Services
- Billing Domain
- Customer Domain
- Order Domain
- Net Domain
- Product Catalog Domain
- Common Domain
- Application Mgmt Domain
- Infrastructure
- Integration
- Legacy
- Util

**Packages**

- com.telener.cos.adapter.caldb.qib
- com.telener.cos.adapter.cossa.curity.conv
- com.telener.cos.adapter.customermaster
- com.telener.cos.adapter.customermaster
- com.telener.cos.adapter.customermaster
- com.telener.cos.adapter.dinew.dto
- com.telener.cos.adapter.dinew.qib
- com.telener.cos.adapter.dinew.wrapper
- com.telener.cos.adapter.dinew.wrapper
- com.telener.cos.adapter.fakturatell
- com.telener.cos.adapter.gensva.qib
- com.telener.cos.adapter.kurt.qib
- com.telener.cos.adapter.online.qib
- com.telener.cos.adapter.productatalog

**Classes**

- \_InfraNetWrapperStub
- \_IS21ClientWrapperStub
- \_IS212WrapperStub
- \_IS212WrapperStub
- \_MQESWWrapperStub
- \_MQESWWrapperStub
- \_MQRWNOWrapperStub
- \_MQNAWrapperStub
- \_MQNWWrapperStub
- \_MQDCConfirmationWrapperStub
- \_MQOrderPilotWrapperStub
- \_RINCardOrderWrapperStub
- \_RINQueueWrapperStub
- \_RINRechargeCodeProductWrapperStub
- \_RINBatchCardWrapperStub

<http://xradar.sourceforge.net>

Static Report - System: EDS, Version: EDI\_DEV, Date: 2005-03-07-2115109

Designed for use with JDepend, PMD, PMD-CPD, JavaNCSS, JUnit, JCoverage, Checkstyle, JSources, JavadocLint, and JUnit.

[Statics / Dynamics] overview

[scorecard] [analysis] [explanations]

(overview) (architecture) (package design) (code) (duplication) (src) (source control) (system specs)

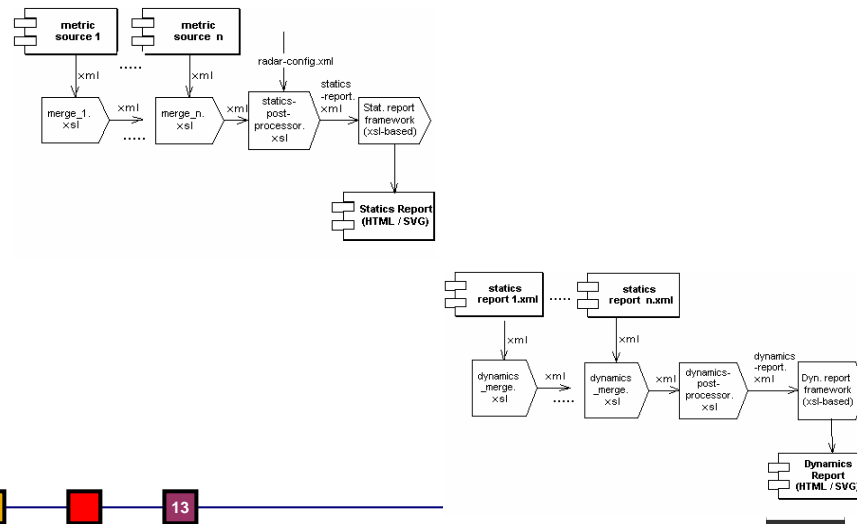
**Scorecard (EDS)**

This page gives a scorecard of this system's quality. The quality measure is defined for this specific system. Every metric has a value in the range from 0 to 1. 0 is the worst possible score, while 1 is the best. The formula for how the metric has been calculated is also shown. Press the subsystem graph to see scorecard details on the specific subsystem.

Listener	Old Client API
Old CRM Services	CRM Services
Billing Domain	Customer Domain
Order Domain	Net Domain
Product Catalog Domain	Common Domain
Application Mgmt Domain	Infrastructure
Integration	Legacy
Util	

Total Quality [EQ= 0.35*TS + 0.35*ARCH + 0.3*CODE]	0.5
Test Suite [TS= 1*TCU]	0.46
Unit Test Coverage [TCU= source statements covered /ncss]	0.46
Architecture [ARCH= 0.4*MOD + 0.6*COM]	0.46
Modularization [MOD= 1 - (count_packages(not(legal-dependencies==0)) / total_packages)]	0.96
Cohesion [COM= 1 - (count_packages(cycles=true) / total_packages)]	0.13
Code Quality [EQES= 0.15*DOE + 0.4*DRY + 0.3*FRE + 0.15*STY]	0.6
Documentation [DOE= javadocs / (functions + analysed-classes)]	0.9
DRYness [DRY= 1 - (classes-with-duplications / analysed-classes)]	0.82
Freshness [FRE= 1 - (classes-with-code-violations / analysed-classes)]	0.42
Stylishness [STY= 1 - (classes-with-style-errors / analysed-classes)]	0.07

## XRadar Architecture



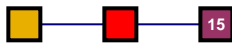
## Dagens kilder og verktøy

- Koden (kilde og kompilert), via...
  - PMD
  - PMD-CPD
  - Checkstyle
  - JDepend
  - JavaNCSS
  - JUnit
  - JCoverage/Cobertura
  - ...
- Konfigurasjon
  - Subsystemdefinisjon
  - Releaser
- Konfigurasjonsstyrings-verktøy/versjonsstyrings-system
  - Aktivitet (feilretting)

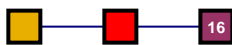
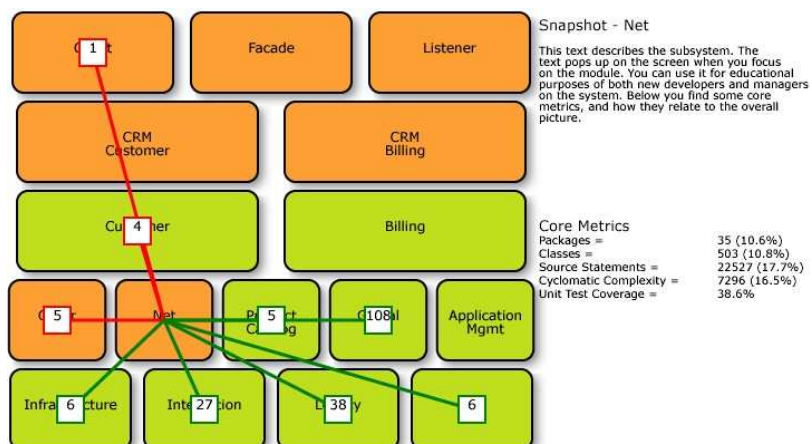


## ■ Eksempler på praktisk bruk

- Kontroll av ulovlige avhengigheter
- Spidergraf på pakke-nivå
- "Svartelister"
- Avvik fra kodenstandard
- Lokalisere kodeproblemer (anti-patterns)

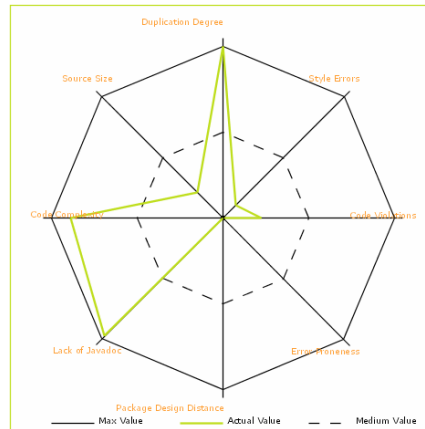


## ■ "Spaghetti"-motvirkning – på subsystemnivå





## Spidergraf på pakke-nivå



- Tilgjengelig fra web og plug-in



17

## "Svartelister"

### Method Analysis - XRadar Test

[\[overview\]](#) [\[class analysis\]](#) [\[complex methods analysis\]](#) [\[changed methods analysis\]](#)

In the analysis below, the methods are ranked by number of changes to method multiplied by the method's cyclomatic complexity. In many systems, this number will be an indicator on methods that should be refactored. The reason for this is that methods with high complexity will have a tendency to be hard to maintain.

Rank	Changes	Source Statements	Cyclomatic Complexity	Package	Class	Method
1	2	32	12	org.xradar.test.f	F1	F1()
2	1	32	12	org.xradar.test.a	A1	testMethod29()
3	1	32	12	org.xradar.test.c	C1	testMethod28()
4	1	32	12	org.xradar.test.c	C1	testMethod29()
5	1	25	6	org.xradar.test.d	D1	methodPartyCopiedToD2()



18



## ■ "Svartelister" II

### Methods Changed Since Last Version Analysis - XRadar Test

[\[overview\]](#) [\[class analysis\]](#) [\[complex methods analysis\]](#) [\[changed methods analysis\]](#)

In the analysis below, the methods are ranked the lack of test coverage of the methods multiplied by their complexity. This analysis shows the methods that have been changed since the last build. Use this report to identify complex methods that are that are not being tested when added or changed.

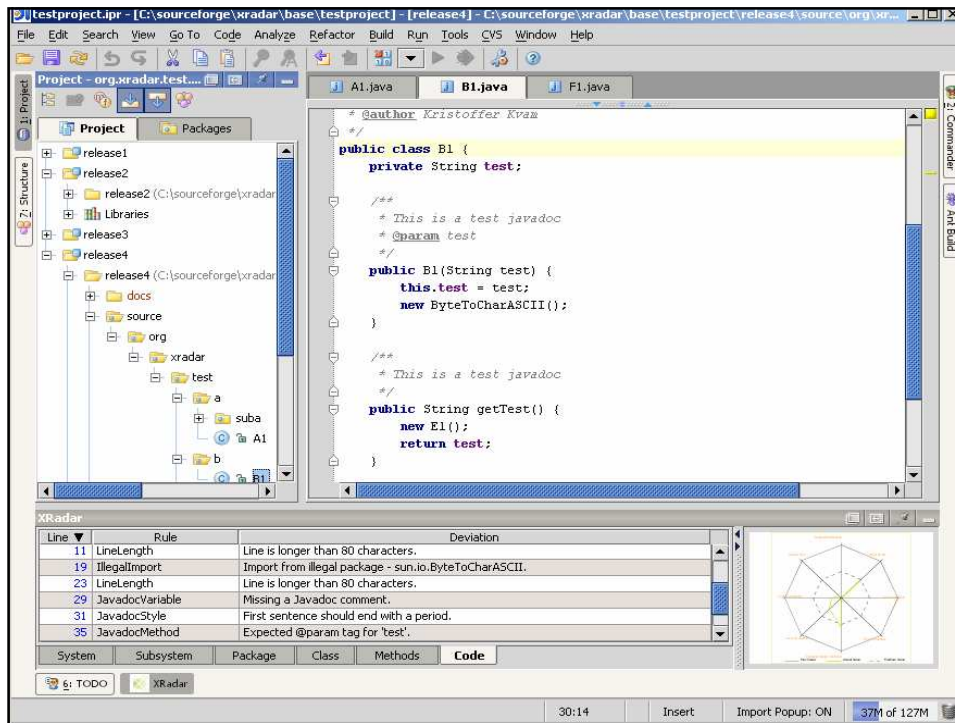
Rank	Changes	Source Statements	Cyclomatic Complexity	Package	Class	Method
1	1	32	12	org.xradar.test.a	A1	testMethod29()
2	1	9	3	org.xradar.test.a	A1	methodWithParameters(boolean,boolean)
3	2	3	1	org.xradar.test.b	B1	getTest()
4	1	1	1	org.xradar.test.c	C1	testMethod1()
5	1	1	1	org.xradar.test.c	C1	testMethod2()
6	1	1	1	org.xradar.test.c	C1	testMethod3()



## ■ Avvik fra kodenstandard

- Mangler og feil
- Kilder: pmd, checkstyle, findbugs(?)
- Enkelt med plug-in





## ■ Lokalisere andre kodeproblemer - "Smells"/Anti-patterns



- Duplisering (copy&paste)
- Store klasser (blobs)
- Spaghetti på klassenivå
  - Høy kompleksitet
  - Sykliske avhengigheter
  - etc
- Redundant kode ("lava flow")



22



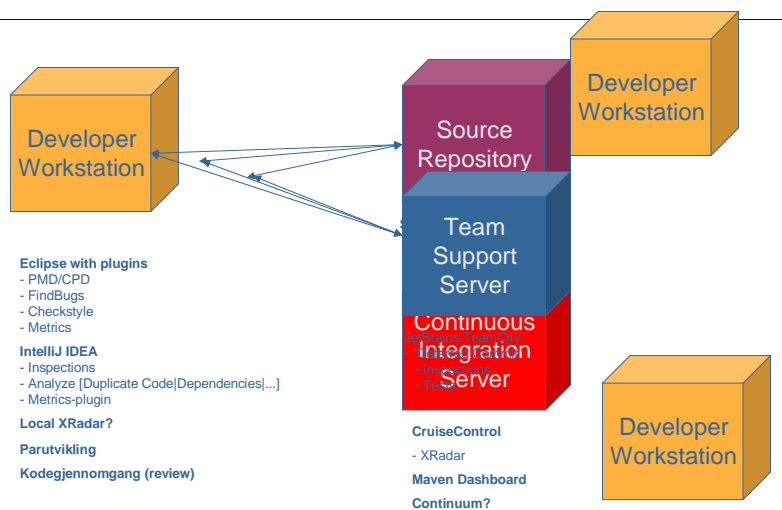
## ■ Noen utfordringer?

- Tilbakemeldingstid
  - Tar tid på store prosjekt som COS (1 time) (og minutter på små prosjekt)
- "Information overflow"
  - Trenger mer skreddersydde "views"
- Effektiv presentasjon i plugins
  - Mye info – liten plass
- Regime for innsjekket kode
  - Bør kjøres *før* innsjekking
- Har mange ideer ... men kunne gjerne hatt enda mer ressurser...



23

## ■ Hvordan komme i gang?



24

SOURCEFORGE **net** RADAR

XRadar Home | XRadar SourceForge

**Introduction**

- Motivation
- Reports
- What's Different?
- Vision
- Credits
- References

**Installation and Execution**

- ▶ Ant Configuration
- ▶ Maven Configuration
- ▶ XRadar-Config.xml
- ▶ Continuous Build

**Documentation**

- ▶ The XRadar Architecture
- ▶ Developer Documentation
- ▶ Plugins

**Analysis Library**

- ▶ Example Analysis

**Project Documentation**

- ▶ About XRadar Test
- ▶ Project
- ▶ Project Info
- ▶ Project Reports
- ▶ Development Process

## News

**Announcing the XRadar Partner Program**

The XRadar team is growing and new commercial and academic partners are now working on their parts of the system. See [here](#) for more.

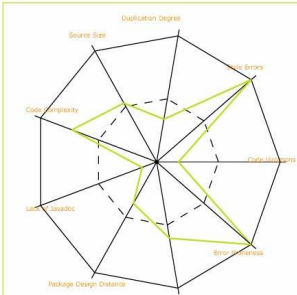
**XRadar version 0.98 is released:**

- The maven plugin is now ready for download under the [sf project page](#). Your will be amazed at how simple it is to run the XRadar on your project. Installation instructions can be found under [Maven Configuration](#).
- Some bugfixes have been done on the reports.
- Simplified configuration.

Go to the [Test Report](#) for a teaser. To get the full potential, you need a SVG viewer in your browser. XRadar downloads can be found under file releases under the [sourceforge project page](#).

---

## XRadar



The XRadar is an open extensible code report tool currently supporting all Java based systems. The batch-processing framework produces HTML/SVG reports of the systems current state and the development over time - all presented in sexy tables and graphs. It gets results from several brilliant open source projects and a couple of in house grown projects and presents the results as massive unified html/svg reports. The architecture is based on java, xml and xsl. Presently it only supports Java, but there are plans to produce plug ins for other leading languages.


The XRadar was build to solve the needs of a large reengineering initiative. Hence, reports and results presented are based on real requirements - likely to match needs in your organisation as well. An important design requirement was that all stakeholders should get their preferred view of the system : managers, architects and developers. We believe they all deserve an ownership and understanding of the systems quality and development. Hence you can navigate from abstract system quality representations, through modules, packages down through classes to the source and javadoc - everything is integrated.

As default, the XRadar gives measurements on standard software metrics


RADAR

## ■ XRadar – oppsummering

- Rapport- og analyseverktøy for Java-løsninger
- Styringsverktøy
  - Utviklere, arkitekter og IT-ledere
  - Målbildeoppnåelse og systemkvalitet
- Detekterer automatisk problemområder og trender i systemer
- Internasjonal oppmerksomhet, open source (på Sourceforge siden 2004) under BSD-lisens



26



## ■ XRadar

---

### Presentasjoner

- XP2004
- javaBin-møte (juni 2004)
- For Simula og Norsk Regnesentral (2005)
- En rekke konsulentselskap (2004,2005)
- OOPSLA (2005)
- JavaZone (2005)
- Fag på IFI... (2005/2007)
- +++

### Bra interesse

- Flere forskningsprogram er relatert til XRadar
- Over hundre nedlastninger hver måned



---

Spørsmål?



■ Kristoffer Kvam ([kristoffer.kvam@telenor.com](mailto:kristoffer.kvam@telenor.com))  
Kjetil Jørgensen-Dahl ([kjd@nos.no](mailto:kjd@nos.no))

<http://xradar.sourceforge.net/>

