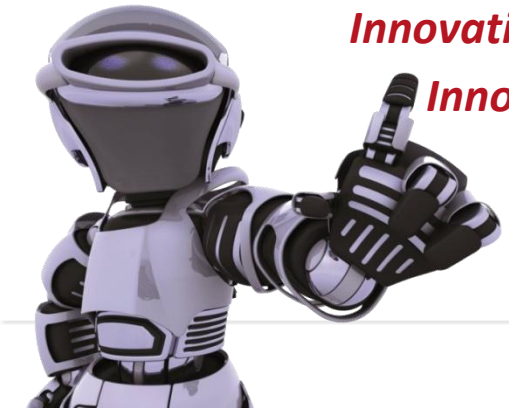
INFOSTROY

Information Systems Technologies Research Outsourcing for You

Метрический анализ исходного кода.
Автоматизированная генерация рекомендаций по совершенствованию кода и архитектуры ПО

2012 CEE-SEC(R)
Software Engineering Conference in Russia

Липанов Александр
Доцент кафедры Информатики Харьковского Национального Университета Радиоэлектроники,
Директор ООО Инфострой
alex@infostroy.com.ua



Innovations in Time.
Innovations for You.

Microsoft Partner
Silver Web Development
Silver Software Development

Modelio Partner

intel Software Partner

ISO 9001 BUREAU VERITAS Certification

Метрики исходного кода ПО

Метрики исходного кода это численные показатели вычисляемые на основе данных получаемых при проведении анализа кода.

Виды метрик:

Статистические метрики:

Количество (уровень приложения): Packages, Namespaces, Types, Global Types, Classes, Global Classes, Interfaces, Global Interfaces, Structures, Global Structures, Methods, Properties, Fields, Global Fields, Lines Of Code, Comments
Comments Density, Level, Percent Public Data, Depth Of Inheritance Tree, Response For a Class
Количество (уровень методов): Parameters, Overloads, Functions

Объектно-ориентированные:

Coupling, Afferent Coupling, Efferent Coupling, Instability, Relational Cohesion, Distance from the Main Sequence, Abstractness, Association Between Classes, Cohesion of LCOM , Cohesion of LCOM HS , Modularity , OOP Level For Types, OOP Level For Methods, OOP Level For Fields

Задачи, которые возникают при анализе кода с использованием метрик

- Понимание значения метрик исходного кода с точки зрения их значения для исходного кода
- Применение вычисленных значений метрик
- Оценка качества исходного кода при наличии значений метрик
- Оценка правильности направления, по которому идет разработка исходного кода
- Как метрики связаны с такими свойствами исходного кода, как надежность, повторное использование, расширяемость и т.д.?
- Получение рекомендаций по улучшению кода на основе метрик исходного кода
- Какие существуют зависимости между метриками и как модифицировать исходный код, чтобы достичь наилучших показателей с точки зрения всей группы метрик?

Этапы анализа кода и формирование рекомендаций по его улучшению

- Формирование системы метрик исходного кода
- Определение подходов формирования рекомендаций к улучшению исходного кода с использованием результатов вычисления метрик
- Анализ взаимосвязей метрик и исходного кода

Области применения метрик

	Номер уровня (j)	1	2	3	4	5	6	7
Номер метрики (i)	Метрика	Пространства имен или пакеты	Типы	Классы	Методы	Интерфейсы	Структуры	Перечисления
1	Сцепление	+		+		+	+	
2	Центростремитель-ное сцепление	+		+		+	+	
3	Внешнее сцепление	+		+		+	+	
4	Нестабильность	+						
5	Относительное единство	+						
6	Расстояние от главной последовательности	+						
7	Абстрактность	+	+	+		+	+	+
8	Связность LCOM		+	+	+			
9	Связность LCOMHS		+	+	+			
10	Ассоциация между классами		+	+		+	+	+

Сцепление

Сцепление (coupling) или зависимость – это мера того, насколько программный модуль зависит от каждого из остальных модулей

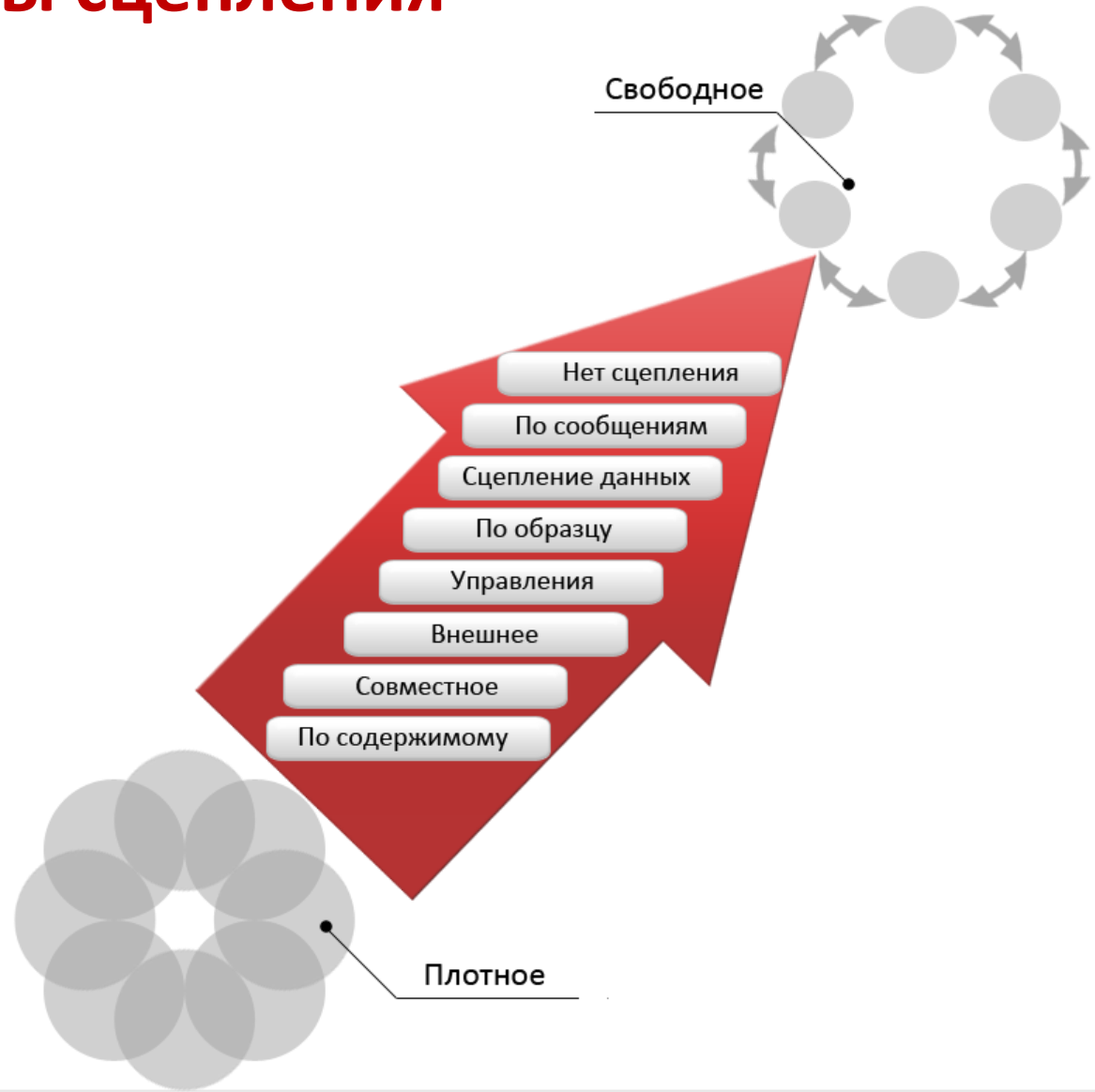
$$Coupling = r_t \left(\sum_{i=0}^{n} r_i \right)^{-1}$$

где r_i – количество ссылок на класс, где ссылка – это поле, локальная переменная, возвращаемый тип или параметр метода,

r_t – количество ссылок на класс, участвующий в подсчете метрики,

n – количество классов.

Типы сцепления



Связность

Связность (cohesion) – это мера того, насколько сильно связаны между собой методы модуля приложения.

$$LCOM = 1 - \frac{1}{M * F}\sum_{i=0}^{n} MF$$

$$LCOM\ HS = \frac{1}{M-1}\left(M - \frac{1}{F}\sum_{i=0}^{n} MF\right)$$

где M – количество методов в классе (статических и экземплярных конструкторов, геттеров и сеттеров свойств, методов добавления и удаления событий),

F – количество экземплярных полей класса,

MF – количество методов класса, имеющих доступ к определенному экземплярному полю,

$\sum_{i=0}^{n} MF$ – сумма MF по всем экземплярным полям класса.

Типы связности

Нестабильность

Нестабильность (instability) – это мера того, насколько зависят классы в пределах модуля.

$$I = \frac{C_e}{C_a + C_e},$$

где C_e – центростремительное сцепление,

C_a – центробежное сцепление.

Ассоциация между классами

Метрика «ассоциация между классами» - **это количество членов других типов или классов, которые непосредственно использует данный конкретный класс в своих методах**. Всегда необходимо стремиться, чтобы метод использовал в основном, элементы собственного класса.

Данная метрика вычисляется отдельно для классов, интерфейсов, структур и перечислений. Для целей экспертной системы с целью нами предложена формула:

$$ABC_X = \sum_{i=0}^{N} i \mid (F \notin X)$$

где ABC_X-значение метрики для некоторого класса;

i-количество обращений к некоторому методу, структуре, интерфейсу или перечислению F, не принадлежащим данному классу X;

N – количество всех классов в системе.

Формирование рекомендаций по улучшению объектно-ориетированной архитектуры кода

При формировании рекомендаций необходимо учитывать следующие факторы:

- **Область применения метрики**, т.е. для каких составляющих исходного кода может быть применена метрика, например, для пространства имен/сборки, типа, класса, метода и т.д. Иными словами уровень анализа исходного кода (таблицы 1, 2);
- **Интервалы значения метрики** – лучший интервал, приемлемый, не приемлемый;
- **Перечень элементов** (пространства имен, классы, методы и т.д.) структуры исходного кода, которые подлежат модификации с целью удовлетворения требованиям метрики;
- **Формулировка базовой рекомендации**, связанной с метрикой
- Правило формирования расширенной информации для рекомендации связанной с метрикой

Этапы работы экспертной системы

- Определение, в какой из интервалов [0, 0.35), [0.35, 0.7) и [0.7, 1] попадает значение метрики, поскольку для интервалов [0, 0.35) и [0.35, 0.7) рекомендации могут отличаться
- Определение, к какому уровню кода относится метрика
- Выбор рекомендации, относящейся к данной метрике на данном уровне метрик и данном интервале. Рекомендация выбирается из заранее сформированного множества рекомендаций
- Формирование дополнительной информации к рекомендации на основе исходного кода в зависимости от уровня метрики и интервала

Формализованное представление работы экспертной системы

$$R = \begin{cases} R^1 m_{ij} \cup Dm_{ij}, & m_{ij} \in [0, 0.35) \\ R^2 m_{ij} \cup Dm_{ij}, & m_{ij} \in [0.35, 0.7) \\ R_S, & m_{ij} \in [0.7, 1] \end{cases}$$

где m_{ij}- значение i-ой метрики на j-м уровне,
$R^1 m_{ij}$- рекомендация, относящаяся к метрике m_i для j-го уровня, если $m_{ij} \in [0, 0.35)$;
$R^2 m_{ij}$- рекомендация, относящаяся к метрике m_i для j-го уровня, если $m_{ij} \in [0.35, 0.7)$;
R_s –сообщение о том, что улучшений кода не требуется в случае, если $m_{ij} \in [0.7, 1]$;
Dm_{ij}- дополнительная информация к рекомендации для определенной метрики некоторого уровня. Эта информация формируется с использованием специальных алгоритмов и правил, определенных для каждой метрики.

Пример рекомендации

Рекомендация для пакета abc.web.components

Классы из данного пакеты используют небольшое количество внешних классов. В этом случае возможно возникновение повторяющегося кода или плохая организация дерева кода. Необходимо реорганизовать код для повышения количества использования внешних классов данным пакетом. Возможно стоит перестроить код таким образом, чтобы количество внешних ссылок приближалось к количеству классов данного пакета. Список классов данного пакета, которые не используют классов других пакетов:

abc.web.components.FloorViewComponent
abc.web.components.AlertedFieldsComponent
abc.web.components.AlertedFieldsComponent.FieldListElement
abc.web.components.BaseStaticFormComponent
abc.web.components.BedViewComponent
abc.web.components.DailyProgramViewComponent
abc.web.components.DocumentsComponent

Процесс разработки ПО с постоянным контролем состояния кода

codEnforcer.com. Облачная система анализа и совершенствования кода

Функции системы

Source code checking (C++, Java, C#, PHP)

- Source code checking basing on schedule
- Metrics calculation for analysis and building improvements scenarios
- Code convention checking
- Code checking basing on user's rules
- Code validation
- Source code statistics collection
- Recommendations for source code improvements

Automated intellectual source code modification and refactoring

- Providing possible ways how to optimize, improve classes, hierarchy and classes/ methods structure
- Proposing scenarios for code modification
- Automatically inserting recommendations into source code scenario
- Manual indication of scenario for automatic source code modification

Working with source code

- Web based tools for source code review
- Web based tools for source code editing
- Assigning tasks for developers
- Integration with SVN and TFS

Team work

- Creating projects groups
- Users and projects management
- Team productivity statistic
- Tasks completion report

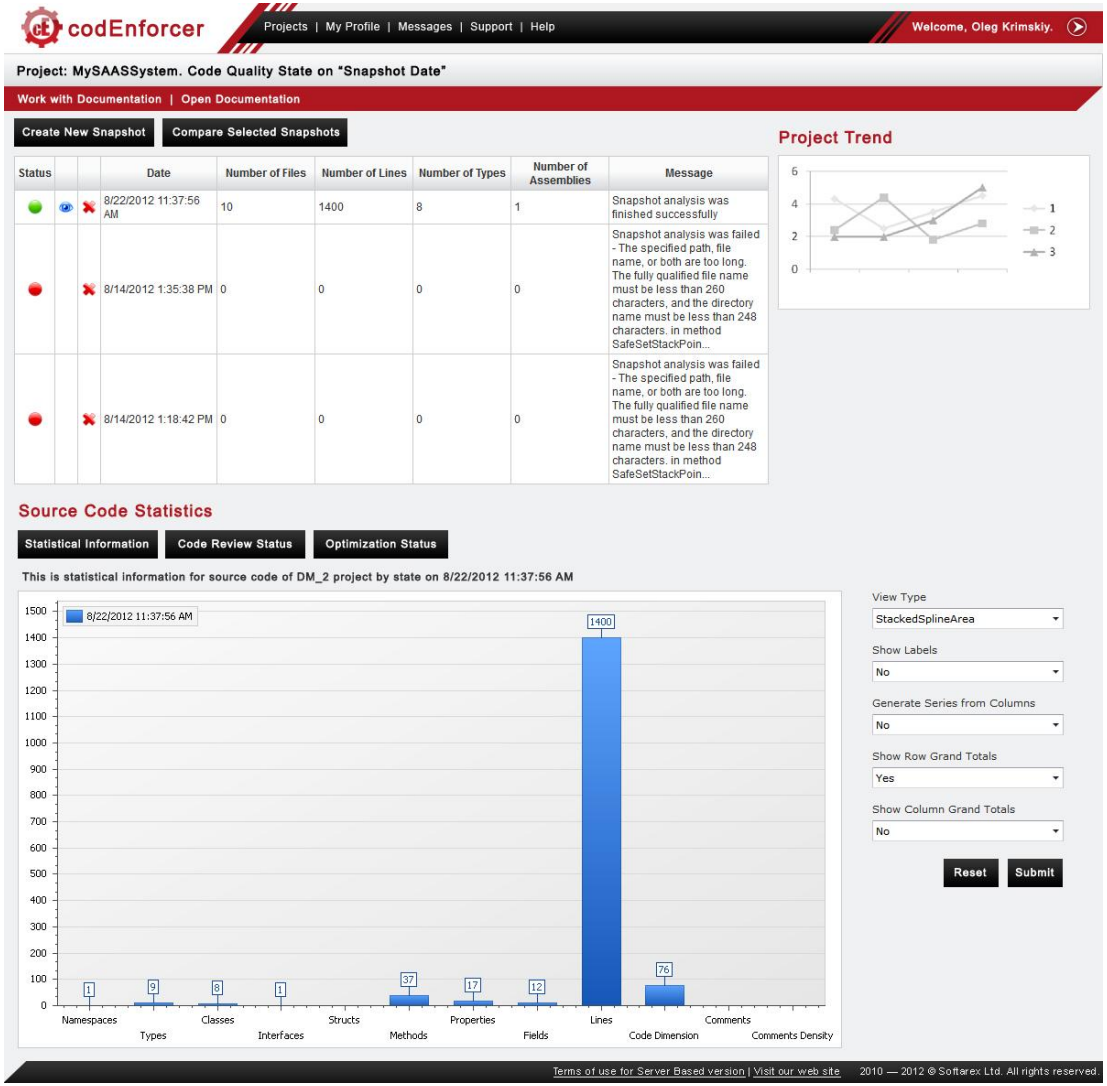
Measurements and Reports

- Source code statistic including weekly and monthly analysis
- Source code optimization forecast for different scenarios
- Source code quality development trend
- Metrics calculations and their changes dynamics

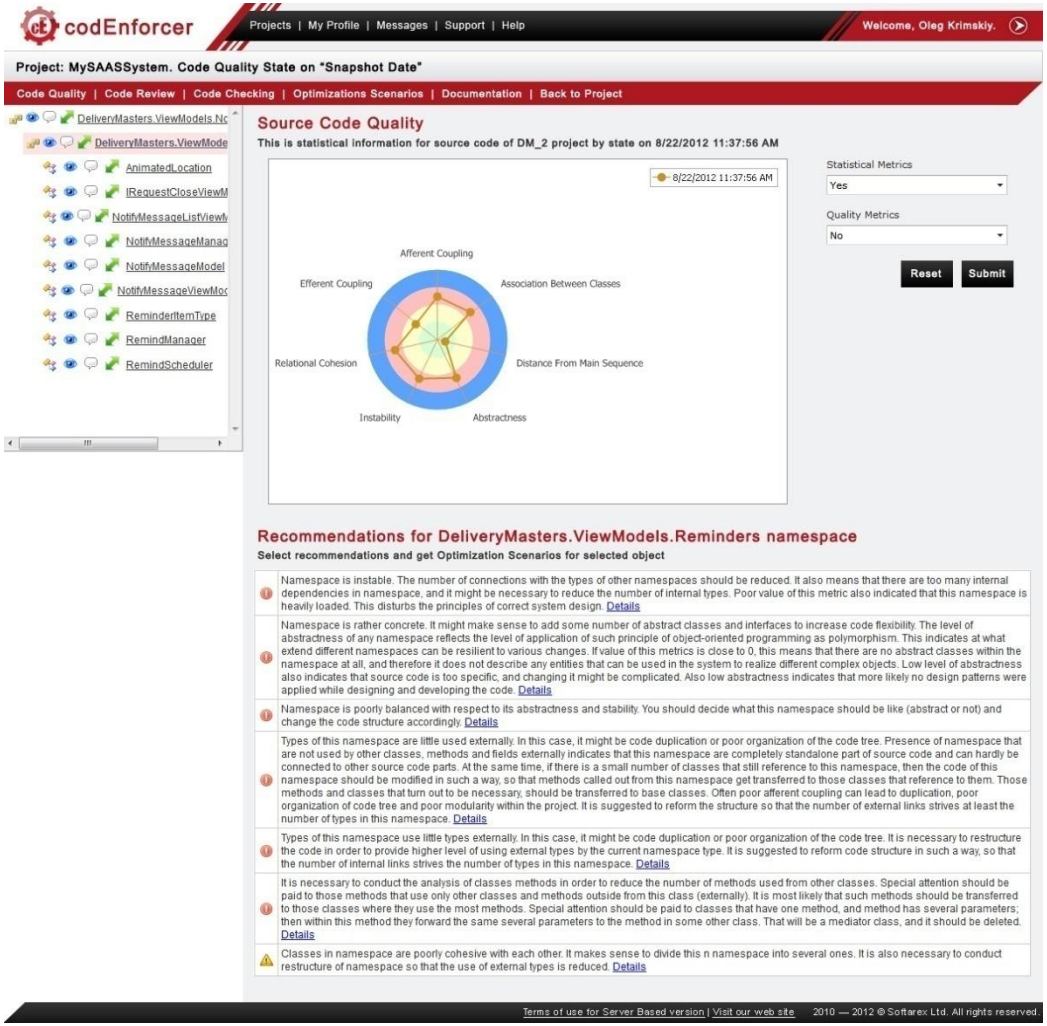
Documenting

- Automatic documentation generation basing on source code
- Source code coverage by documentation checking
- Large set of tools for visual documentation building and editing
- Features for team work on users manuals
- Export of manuals into PDF, CHM and HTML
- Web based project documentation and users manuals portal

Страница проекта в системе codEnforcer



Страница с рекомендациями в системе codEnforcer



codEnforcer.com. Облачная система анализа и совершенствования кода

Для тестирования системы регистрируйте тестовый аккаунт:

http://www.codenforcer.com/

Спасибо за внимание !

Head Office

51, Elizarova str,
Kharkov, Ukraine, 61098
Phone: +380(57) 717 61 54
Web: www.infostroy-software.com
E-mail: info@infostroy-software.com