электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Алгоритм автоматического обнаружения ошибок в поиске контуров штриховок в системе nanoCAD

 # 11, ноябрь 2015

Рудаков И.В.[1], Масный П.В.[2,*]

УДК 519.6

[1]Россия, МГТУ им. Н.Э. Баумана
[2] Россия, СиСофт Девелопмент

[*]masnyypavel@bk.ru

Введение

Одной из сложных задач, решаемых в системах автоматизированного проектирования, является поиск контуров при штриховании, под которым понимается заполнение выбранной области каким-либо узором, используемым для условного обозначения различных материалов [1].

nanoCAD – базовая система автоматизированного проектирования, предназначенная для разработки и выпуска рабочей документации (чертежей) [2]. Как и любой другой функционал в системе nanoCAD, поиск контуров постоянно претерпевает значительные улучшения.

При внесении изменений в программном продукте необходимо гарантировать отсутствие ошибок, для этого используют регрессионное тестирование – дорогостоящая, но необходимая деятельность в рамках этапа сопровождения, направленная на перепроверку корректности измененной программы. При этом объем данных, на которых происходит проверка качества, постоянно увеличивается.

С целью снижения затрат на проведение контроля качества программного обеспечения необходимо разработать алгоритм и программный модуль, автоматически определяющий наличие ошибок или их отсутствие при создании очередного контура, что позволит сделать максимальное покрытие ошибок за относительно короткий промежуток времени.

1. Проблемы задачи

Полный перебор всевозможных вариантов контуров (рис. 1) делает задачу практически не решаемой, поэтому необходимо ограничиться конечным множеством, которое можно будет пополнять в случае необходимости.

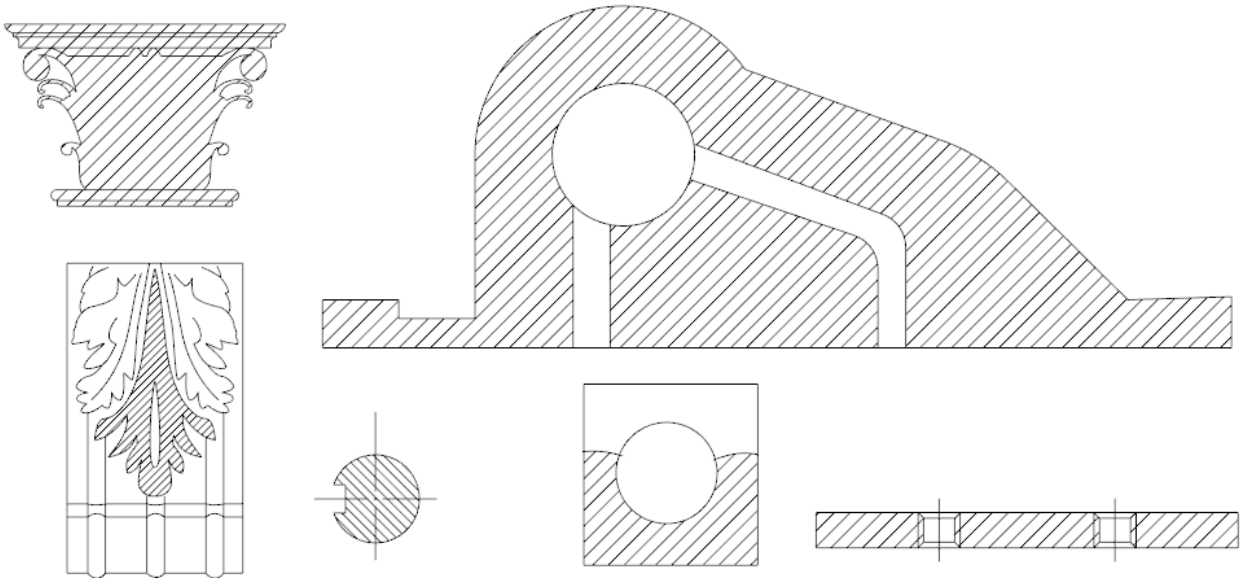

Рис. 1. Примеры контуров штриховки

Одним из методов автоматического обнаружения ошибок в поиске контуров является сравнение анализируемой и эталонной штриховок.

Эталон либо задает эксперт, либо вычисляется аналитически. Первый подход является надежным и простым в реализации, но требует дополнительных временных затрат по подготовке исходных данных экспертом. Второй подход позволяет максимально исключить участие эксперта, но предполагает реализацию параллельного алгоритма поиска нахождения контуров, вследствие чего резко возрастает трудоемкость решения.

2. Алгоритм сравнения двух штриховок

При сравнении двух штриховок будем полагать, что анализируемый и эталонный объекты располагаются один под другим. В этом случае сравнение по координатам вершин двух контуров является необходимым, но недостаточным критерием. Так, например, три заштрихованные области, представленные на рис. 2, имеют идентичные координаты вершин, но фактически являются абсолютно разными. Поэтому вводим дополнительный признак сравнения – кривизна в точке.

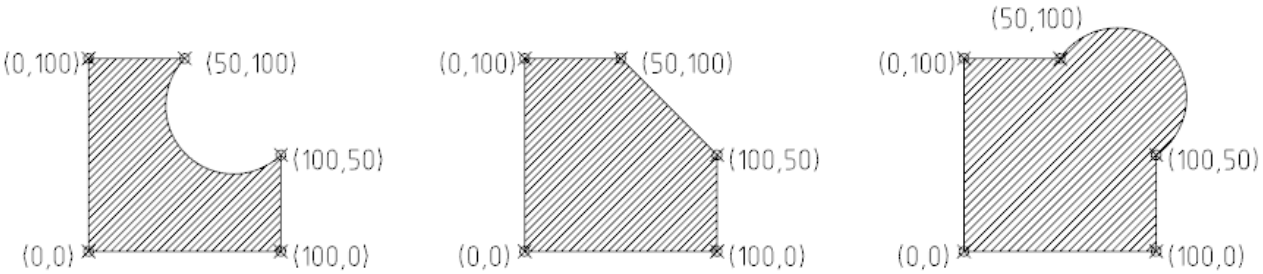

Рис. 2. Сравнение по вершинам

Штриховка, создаваемая в системе nanoCAD, является плоским объектом, поэтому вершины контуров можно представить в некотором абстрактном пространстве XYB – $V_i(x_i, y_i, b_i)$,

где $x_i -$ абсцисса i – ой вершины контура,

$y_i -$ ордината i – ой вершины контура,

$b_i -$ кривизна в i – ой вершине контура.

Тогда под контуром будем понимать конечное упорядоченное множество

$C = \{V_1, V_2, \dots, V_n\}$,

где n – количество вершин контура.

Сравнение двух контуров записывается:

$$r = \text{compare}(C_t, C_s), \tag{1}$$

где $C_t = \{V_1^t, V_2^t, \dots, V_n^t\} -$ анализируемый контур,

$C_s = \{V_1^s, V_2^s, \dots, V_m^s\} -$ эталонный контур,

$r -$ результат сравнения двух контуров

$$r = \begin{cases} 1, \text{если } n = m \text{ и } \overrightarrow{V_i^t V_i^s} = \vec{0} \ (i = \overline{1,n}) \\ 0, \text{иначе} \end{cases} \tag{2}$$

Равенство двух векторов $\overrightarrow{V_i^t V_i^s} = \vec{0}$ из (2) определяется по следующему соотношению:

$$\sqrt{(x_i^t - x_i^s)^2 + (y_i^t - y_i^s)^2 + (b_i^t - b_i^s)^2} < eps, \tag{3}$$

где $eps -$ заданная точность.

Заштрихованная область может состоять из нескольких контуров (рис. 3). Поэтому под штриховкой будем понимать конечное упорядоченное множество контуров:

$$H = \{C_1, C_2, \dots, C_n\}$$

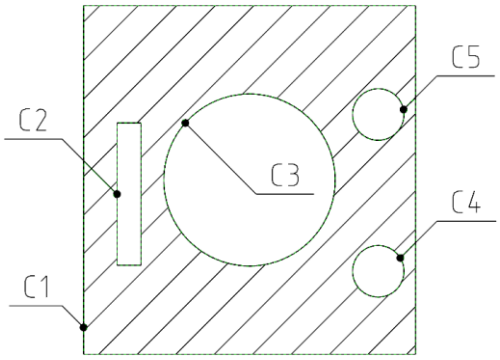

Рис.3. Контуры штриховки

Тогда сравнение двух штриховок можно представить в виде:

$$R = \text{COMPARE}(H_t, H_s), \tag{4}$$

где $H_t = \{C_1^t, C_2^t, \dots, C_n^t\} -$ анализируемая штриховка,

$H_s = \{C_1^s, C_2^s, \dots, C_m^s\} -$ эталонная штриховка,

R – результат сравнения двух штриховок.

$$R = \begin{cases} 1, \text{если } n = m \text{ и } r_i = 1 \ (i = \overline{1,n}) \\ 0, \text{иначе} \end{cases} \tag{5}$$

На рис. 4 представлен разработанный алгоритм сравнения двух штриховок

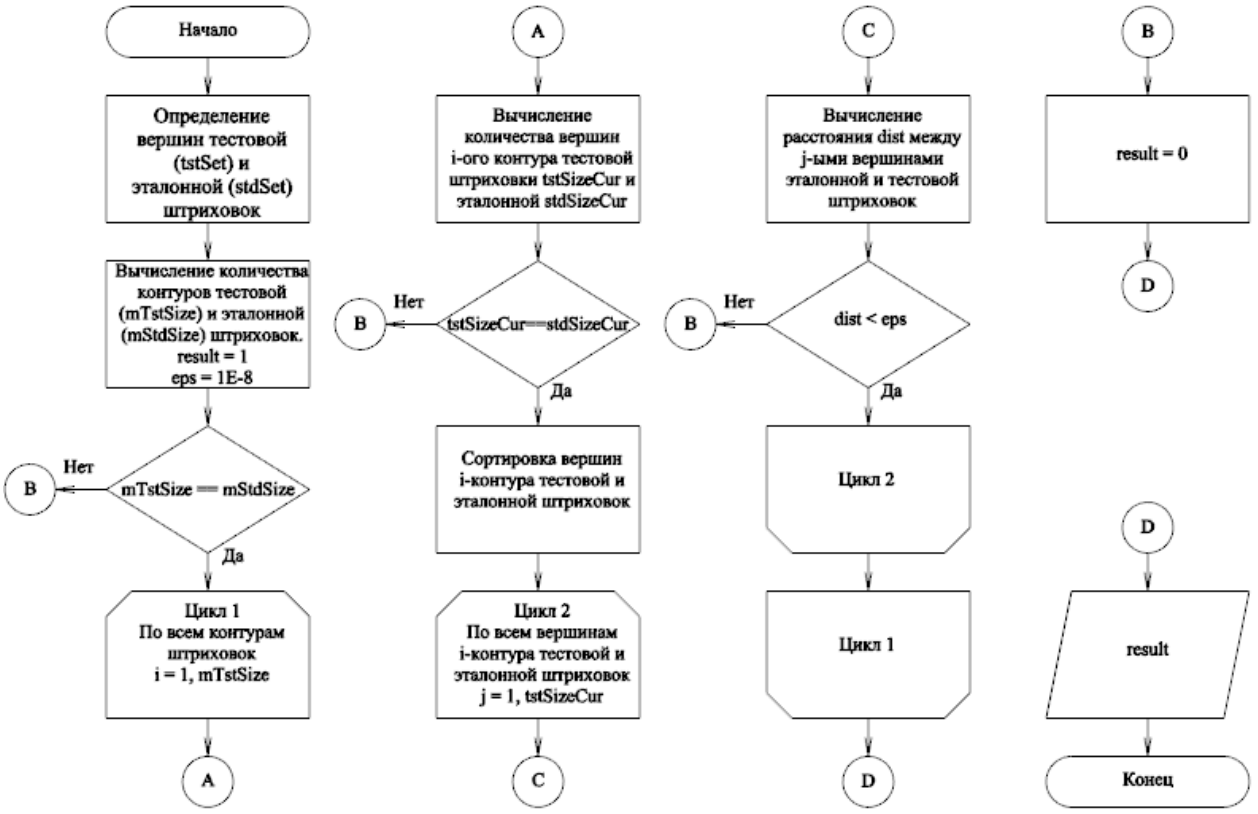

Рис. 4. Алгоритм сравнения эталонной и тестовой штриховок

2.1. Вычисление временной сложности алгоритма

Исходными данными для получения функций вычислительной сложности от входа задачи является время выполнения операторов в условных операциях или тактах и количества повторений каждого оператора, которое определяется размером входа задачи и структурой алгоритма[3].

Расчет временной сложности алгоритма выполнен на базе одного контура, состоящего из 1332 вершин. Результаты вычислений представлены в таблице 1.

Таблица 1. Время выполнения операторов в условных операциях

Операция алгоритма	Количество повторений	Время за одно повторение, с
Определение вершин контура	1	0.00531126
Вычисление количества контуров	1	4.10516e-007
Провекра условия mTstSize == mStdSize	1	4.10516e-007
Вычисление количества вершин i-ого контура	1	4.10516e-007
Провекра условия tstSizeCur == stdSizeCur	1	4.10516e-007
Сортировка вершин контура	1	3.72913
Вычисление рассятояние dist между j-ыми вершинами контуров	1332	0.000271351
Проверка условия dist<eps	1332	1.03245e-007

Время выполнения операций алгоритма с учетом количества повторений:

$$T = \sum_{i=1}^{r} k_i \cdot T_i \approx 4.096\ c.$$

3. Разработка программного модуля

Алгоритм сравнения двух штриховок реализован средствами разработки nanoCAD SDK, а именно NrxGate – C++ API, обеспечивающий непосредственный доступ к структурам базы данных nanoCAD и определениям встроенных команд.

На основе разработанного алгоритма на рис. 5 представлена диаграмма классов модуля.

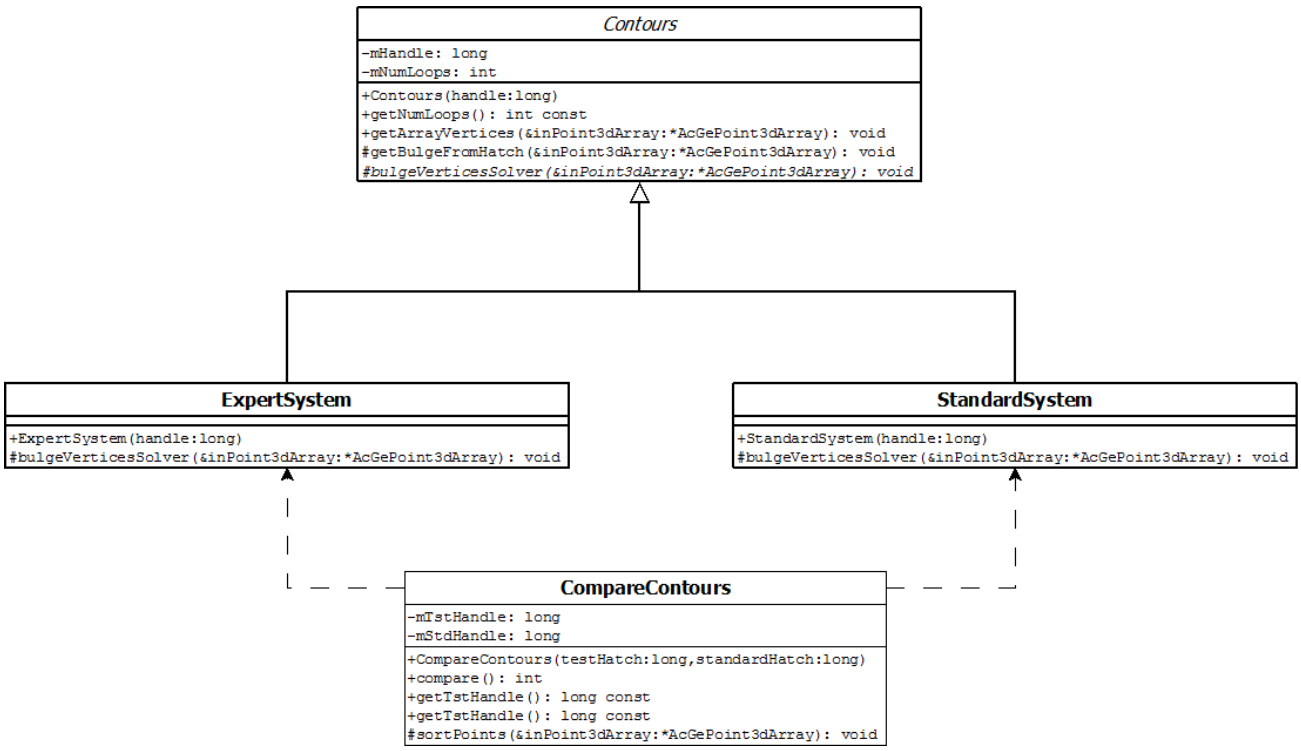

Рис. 5. Диаграмма классов

Одним из основных методов является bulgeVerticesSolver, определяющий количество контуров в штриховке и формирующий массив точек для i-ого контура. Сравнение двух штриховок реализовано в методе compare класса CompareContours.

Для удобного формирования списка эталонных штриховок реализован графический интерфейс (рис. 6). При добавлении очередной эталонной штриховки происходит запись в .dwg-файл следующей структуры:

```
struct strXRecord
{
   AcDbHandle  handle;
   ads_point       point3d;
};
```

point3d – координаты затравочной точки для тестируемой штриховки,

handle – уникальный номер объекта (в данном случае эталонная штриховка).

Данная операция необходима для того, чтобы в дальнейшем различать эталонные объекты от обычных.

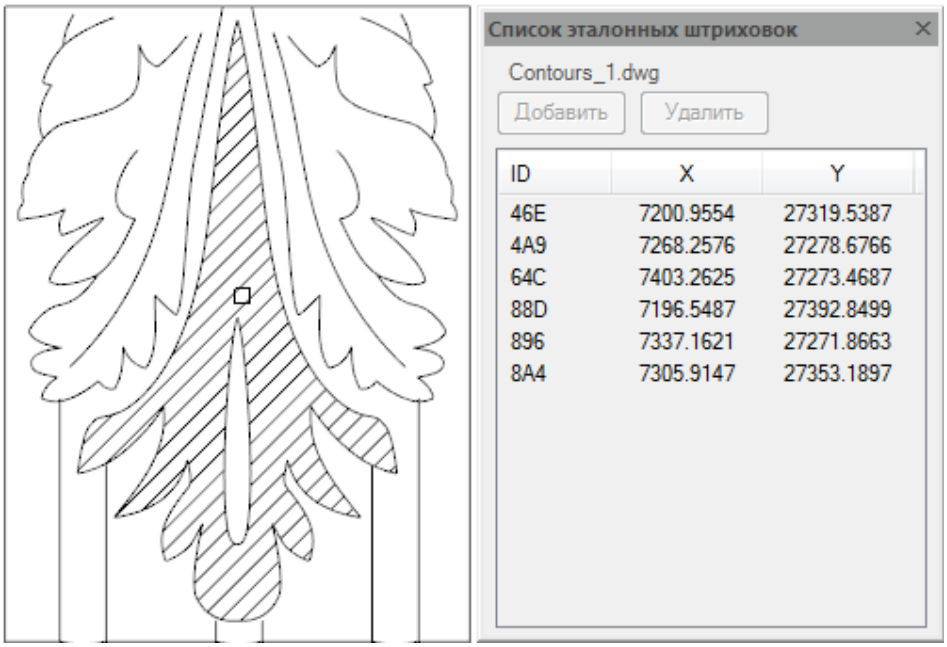

Рис.6. Добавление эталонной штриховки

3.1. Запуск в автоматическом режиме

Для автоматического запуска используется универсальный проигрыватель, написанный на Visual Studio Coded UI Tests (далее Runner) [4].

На рис. 7 представлена диаграмма взаимодействия универсального проигрывателя и системы nanoCAD.

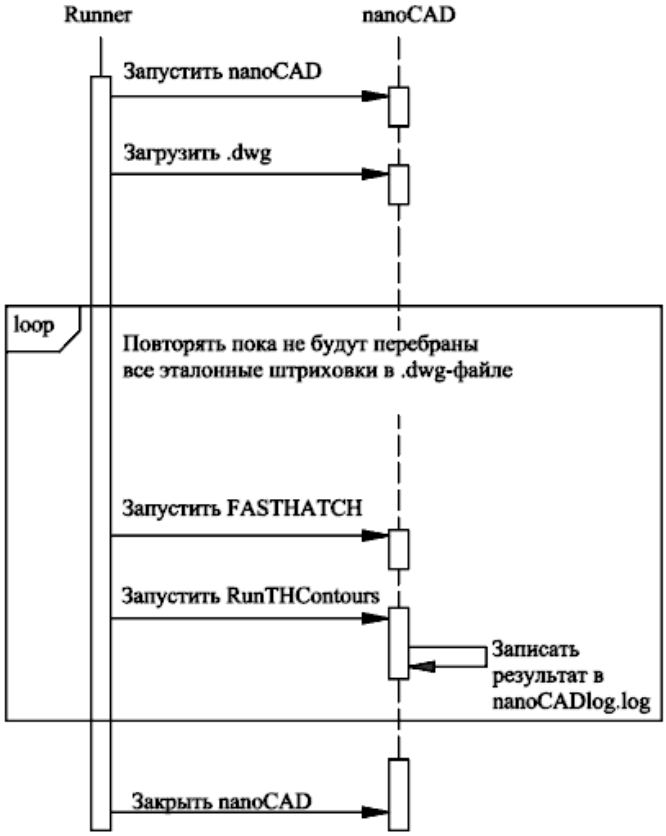

Рис.7. Диаграмма взаимодействия

4. Сравнение ручного и автоматизированного обнаружения ошибок в поиске контуров штриховок

Результаты сравнения приведены в таблице 2. Сравнение показывает тенденцию тестирования, ориентирующую на максимальную автоматизацию процесса тестирования [5] для рассматриваемого функционала САПР.

Таблица 2. Сравнение ручного и автоматизированного подхода

	Ручное	Автоматизированное
Задание входных значений	Гибкость в задании данных. Позволяет анализировать разные контура, расширяя покрытие	Поддерживается. Для этого необходимо лишь добавить новый эталон
Проверка результата Выявление признаков ошибки	Гибкая. Позволяет эксперту оценивать нечетко сформированные критерии	Строгая. Позволяет сравнивать только с эталонным значением и выдать результат либо истина, либо ложь
Повторяемость	Низкая. Человеческий фактор и нечеткое определение данных приводит к не повторяемости тестирования	Высокая
Надежность	Низкая. Длительные цикловые проверки приводит к снижению внимательности эксперта	Высокая
Скорость выполнения тестового набора	Низкая	Высокая

Для расчета показателя экономической эффективности воспользуемся формулой:

$$E_n = \frac{A_a}{A_m} = \frac{V_a + n \cdot D_a}{V_m + n \cdot D_m}$$ [6]

Vm – затраты на создание контуров в .dwg-файле,

Va – затраты на создание контуров и выбора эталонов в .dwg-файле,

Dm – затраты на анализ одного подготовленного файла, содержащего контуры,

Da – затраты на анализ .dwg-файла в автоматическом режиме.

Под подготовкой понимается создание анализируемых контуров в .dwg-файле. В среднем подготовка одного файла, содержащая 300 контуров, для ручного тестирования занимает 1.5 часа. Для автоматизированного тестирования помимо создания контуров штриховок необходимо выбрать эталон. В результате подготовка V для автоматизированного поиска ошибок составляет 1.7 часа.

На основе измерений, полученных в результате однократного выполнения тестирования, можно вычислить расходы, которые потребовались бы для повторного тестирования 5, 10 или 100 раз. Например, в случае 10 автоматизированных выполнений тестов коэффициент уменьшения расходов на тестирование по сравнению с проведением тестирования вручную составит:

$$E_{10} = \frac{A_a}{A_m} = \frac{V_a + 10 \cdot D_a}{V_m + 10 \cdot D_m} = \frac{1.7 + 10 \cdot 0.25}{1.5 + 10 \cdot 0.5} \approx 0.65$$

Результаты показателя экономической эффективности представлены в таблице 3. На рис. 8 изображен график снижения стоимости тестирования от количества повторений тестового набора.

Таблица 3. Определение показателя экономической эффективности

Подготовка V, ч		**Выполнение D, ч**		**Коэффициент уменьшения расходов E на выполнение n автоматизированных тестов, %**			
Ручное	**Автоматизированное**	**Ручное**	**Автоматизированное**	**1**	**5**	**10**	**100**
1.5	1.7	0.5	0.25	98	73	65	52

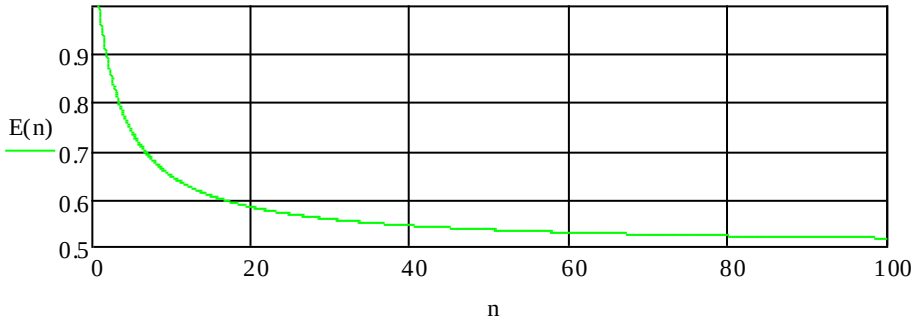

Рис. 8. Снижение стоимости тестирования от количества тестов

Заключение

Созданный алгоритм и разработанный программный модуль позволяют автоматически сравнивать эталонную и анализируемую штриховки. Для удобного формирования списка эталонных штриховок разработан графический интерфейс, интегрируемый в систему nanoCAD. Подготовка .dwg-файла для автоматического обнаружения ошибок по времени незначительно выше по сравнению с ручным анализом. Основные преимущества разработанного модуля проявляются при повторном проведении тестировании, т.к. автоматический поиск ошибок исключает человеческий фактор и нечеткое определение данных, увеличивает скорость обнаружения ошибок, обеспечивает гибкость при добавлении новых исходных данных. Кроме того, архитектура модуля позволяет реализовать экспертную систему, которая даст возможность максимально исключить участие пользователя при формировании исходных данных.

Список литературы

[1]. Руководство пользователя nanoCAD // Сайт компания ЗАО «Нанософт». Режим доступа: http://www.nanocad.ru/products/download.php?id=371 (дата обращения 12.11.2015)

[2]. nanoCAD // Википедия. Режим доступа: http://ru.wikipedia.org/wiki/NanoCAD (дата обращения 12.11.2015)

[3]. Иванова Г.С. Автоматизация анализа вычислительной и емкостной сложности алгоритмов на множествах и графах. // Инженерный журнал: наука и инновации. 2013. №11. Режим доступа: http://engjournal.ru/catalog/it/hidden/1043.html (дата обращения 12.11.2015) **DOI:** 10.18698/2308-6033-2013-11-1043

[4]. Слободин И.Б. Универсальный автотест, или как мы автоматизировали ручные тесты API в nanoCAD. // CADMASTER. 2013. № 6. С. 58-60. Режим доступа: http://www.CADmaster.ru/assets/files/articles/cm_73_08.pdf (дата обращения 12.11.2015)

[5]. Документирование и оценка индустриального тестирования. // Сайт Интуит. Национальный открытый университет. Режим доступа: http://www.intuit.ru/studies/courses/48/48/lecture/1442?page=1 (дата обращения 12.11.2015)

[6]. Дастин Э., Рэшка Д., Джон П. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатации. М.: Лорри. 2003. 590 с. С.53