инженерия программного обеспечения

Сафонов Владимир Олегович

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ НАДЕЖНЫХ И БЕЗОПАСНЫХ ПРОГРАММ (TRUSTWORTHY COMPUTING)

Аннотация

В статье рассматриваются современные подходы к разработке надежных и безопасных программ *(trustworthy computing – TWC)*. Особое внимание уделено взаимосвязи TWC и аспектно-ориентированного программирования (АОП), применению АОП и системы Aspect.NET для разработки надежных и безопасных программ [1].

Ключевые слова: надежность программ, безопасность программ, защита конфиденциальной информации, аспектно-ориентированное программирование.

ВВЕДЕНИЕ. ИНИЦИАТИВА TRUSTWORTHY COMPUTING (TWC) ФИРМЫ MICROSOFT

Данная статья тесно связана по тематике с монографией [1] и является первой работой автора по данной теме на русском языке.

Одним из самых важных и перспективных направлений ИТ является надежное и безопасное программирование. В более широком смысле в современной англоязычной терминологии данный термин звучит как *trustworthy computing*, что в буквальном переводе означает *вычисления, заслуживающие доверия*. Что это означает на практике?

Как известно, в течение всего развития программирования и программного обеспечения при любых подходах к разработке программ наиболее важными их качествами признавались *работоспособность* и *полезность (usability)* – программа должна выполнять то, чего от нее ожидают в заданных условиях, а также быть дружественной к пользователю: быть удобной, иметь комфортный пользовательский интерфейс, «вести себя разумно» с точки зрения житейской и профессиональной логики. Только такая программа, на наш взгляд, заслуживает доверия пользователей, поэтому и ставлю эти качества на первое место при определении понятия *trustworthy computing*.

Если рассматривать понятия, более близкие к предмету настоящей статьи, то следует признать, что важнейшим качеством программ является их *безопасность* (*security*): программа должна уметь защищаться от внешних угроз и атак и сохранять работоспособность после их отражения. В настоящее время этому придается особое внимание, так как, ввиду широкого распространения сетей и ввиду усложнения архитектуры программ, появилось гораздо больше возможностей для хакерских атак, последствия которых могут быть очень серьезны, так как использование программного обеспечения стало частью нашей повседневной жизни.

Другое важнейшее качество программы – *надежность* (*reliability*): программа должна вести себя разумно и предсказуемо в случае некорректных исходных данных и обеспечивать безотказную работу в течение как можно более долгого периода времени. Понятие надежности в ранних работах было принято количественно характеризовать в терминах *Mean Time Between Failures (MTBF) – среднего времени наработки на отказ*, то есть среднего времени, в течение которого система работает безотказно. В настоящее время требуются более детальные и точные количественные оценки и предсказание надежности, над которыми и ведется работа.

Третье качество. Программа должна обеспечивать *защиту и корректность использования конфиденциальной информации (privacy)* – личных и банковских данных, информации, используемой в повседневной работе предприятия и т. д. Такого рода информация должна запрашиваться у пользователя и использоваться только в случае крайней необходимости, для конкретных целей, которые программа должна понятно объяснять пользователю, своевременно предупреждая его об этом и заручаясь его одобрением и доверием.

Четвертое качество. Фирма-разработчик программы должна обеспечивать *целостность и корректность бизнеса* (*business integrity*), связанного с разработкой и использованием программы. Данное качество имеет две стороны. Во-первых, это *четкая организация сопровождения программы* – быстрое исправление ошибок и быстрые ответы на вопросы пользователей, которые, в свою очередь, должны оценить хорошую организацию бизнеса, связанного с разработкой и сопровождением, и, как следствие, продолжать пользоваться программой и приносить ее разработчикам доход. Во-вторых, это *контроль прозрачности, законности, корректности бизнеса компании-пользователя программы*. Имеется в виду, прежде всего, что программа должна подсказывать пользователю, как ее использовать корректно. Если использование программы связано с ведением какого-либо бизнеса – продаж, рекламы, перечисления денежных средств и т. д., – то программа должна контролировать корректность и законность всех операций, которые пользователь выполняет с ее помощью, а в случае обнаружения каких-либо отклонений от этого правила – предупреждать и консультировать пользователя, предотвращая его возможные несанкционированные и незаконные действия. В настоящее время в стадии разработки в корпорации Microsoft находятся интеллектуальные решения для бизнеса, которые осуществляют контроль и консультирование в области законности бизнеса на основе баз знаний.

Четыре перечисленных качества программ – *security, reliability, privacy, business integrity*, которым в наше время придается особое значение, стали основой *инициативы trustworthy computing (TWC)* [2], провозглашенной в 2002 г. корпорацией Microsoft. В меморандуме TWC фирмы Microsoft они названы *«четырьмя колоннами TWC» (four pillars).* Качество *usability* мы добавляем к рассмотренной парадигме, исходя из соображений здравого смысла.

Главная цель инициативы TWC – обратить внимание фирм-разработчиков программного обеспечения на особую важность указанных качеств программ. Разработчики должны стремиться к тому, чтобы учитывать требования TWC, начиная с самых ранних этапов разработки и на каждом ее этапе.

Классической работой по TWC стала книга [3], в которой подробно, на содержательных примерах, объясняются принципы разработки безопасного кода. Книга имеет приложение на Web-сайте, где доступны для скачивания через Web полезные примеры безопасного кода.

В мировой практике использования программного обеспечения Microsoft – операционных систем семейства Windows, пакета Microsoft Office, обозревателя Internet Explorer и др. – как широко известно, имели и имеют место постоянные попытки хакеров с помощью *злонамеренных программ (malicious software,* или, ко-

ротко, *malware*) – *вирусов (viruses), «червей» (worms), троянских программ, или троянов (Trojan programs)*, – использовать *уязвимости (vulnerabilities)* программного обеспечения Microsoft для широкомасштабных сетевых атак.

В ответ на это Microsoft буквально еженедельно выпускает обновления своих программных продуктов, основная часть которых – исправление уязвимостей их безопасности.

Необходимо отметить, что, на наш взгляд, все это отнюдь не свидетельствует о ненадежности программных продуктов Microsoft. Это подтверждает только лишь их широкую популярность. Надеюсь, что моя мысль читателю понятна.

Любопытно, что, по признанию самих экспертов Microsoft [3], корпорация поддерживает *неофициальные контакты с хакерами*, – видимо, это наиболее надежный способ вовремя узнавать об их планах.

С целью коренного улучшения ситуации, фирма Microsoft разработала и применяет особую схему *жизненного цикла разработки безопасных программ (Security Development Lifecycle, SDL)* [4]. Она является результатом анализа серьезных уроков, извлеченных Microsoft из опыта эксплуатации предыдущих версий Windows до Windows XP. Для улучшения безопасности уже выпущенной и распространенной по всему миру ОС Windows 2000, разработанной еще до введения SDL, Microsoft пришлось срочно прервать дальнейшую разработку системы, «посадить за парты» всех ее разработчиков, обучить их в кратчайшие возможные сроки принципам разработки безопасных программ, а затем за короткий срок исправить уже существующий гигантский код операционной системы и распространить новую, более надежную и безопасную версию. В мировую историю программирования этот беспрецедентный шаг вошел под названием *«security push»* [3].

Чтобы избежать подобных дорогостоящих экстренных мер при разработке нового программного обеспечения, Microsoft рекомендует применять схему SDL – *проектировать, реализовывать и тестировать реализацию подсистемы безопасности на каждом этапе жизненного цикла программы*, то есть *заниматься безопасностью программы постоянно, начиная с самых ранних этапов ее разработки*. В этом – основная суть схемы SDL.

Поскольку для практического осуществления данного принципа требуются высококлассные эксперты по компьютерной безопасности, которыми большинство инженеров-программистов, даже в Microsoft, к сожалению, отнюдь не являются, – Microsoft рекомендует в каждом проекте иметь небольшую *группу экспертов по безопасности*, либо хотя бы одного эксперта, возможно, приглашенного из другой компании, – «*security buddy*». Цель экспертной группы по безопасности – *постоянный контроль соблюдения правил разработки безопасного кода на всех этапах разработки*, консультации разрабочиков по безопасности и т. д.

Важно отметить, что, согласно принципам TWC и SDL, безопасность системы начинается еще *до ее разработки*. В документе, описывающем требования (requirements) к системе, должны быть отдельными разделами сформулированы *требования к безопасности, типичные возможные угрозы (threats) системе и методы их отражения (mitigation)*. Для осуществления этих принципов необходимо еще на начальных этапах разработки заниматься *моделированием угроз (threat modeling)* [5].

Вот некоторые важнейшие принципы разработки безопасного кода, согласно TWC:

- *Принцип минимизации атакующей поверхности программы (minimizing the attack surface)*. Необходимо проектировать и реализовывать программу так, чтобы хакеры имели как можно меньше возможностей «взломать» данные работающей программы и вывести ее из строя.
- *Принцип минимальных привилегий (least privilege)*. В большинстве случаев разработка и отладка программ ведется разработчиками «от имени администраторов», то есть программа тестируется от

имени пользователя, имеющего права администратора, например право изменения реестра или других системных файлов. При попытке использовать такую программу от имени обычного пользователя могут возникнуть проблемы, связанные с отсутствием полномочий. Чтобы этого не случилось, программу необходимо обязательно тестировать от лица обычного непривилегированного пользователя. Функциональность программы не должна опираться на то, что пользователь имеет права администратора.

- *Принцип обеспечения безопасности по дизайну, по умолчанию, при развертывании (secure by design, by default, by deployment).* Как уже говорилось, еще на этапе проектирования (дизайна) программы необходимо предусмотреть для нее меры безопасности. Программа должна быть реализована так, чтобы быть *безопасной по умолчанию*, то есть все меры и проверки безопасности по умолчанию должны быть включены (даже если это приведет к некоторому замедлению программы или некоторым неудобствам для пользователя). Мы с вами хорошо знаем, что последнее означает на практике: ОС и браузер запрашивают дополнительные подтверждения от пользователя в случае потенциально небезопасных для системы действий – просмотра незнакомого сайта, скачивания и инсталляции чужих программ и др. Это характерно для новых версий Windows. Первоначально подобные предосторожности могут раздражать пользователя, однако они вполне оправданы, так как предотвращают многие типичные атаки.

Мировое сообщество программистов, к сожалению, в целом пока скептически, осторожно и инертно относится к инициативе TWC. Между тем, проблемы с безопасностью и с внешними атаками испытывают программные продукты всех фирм-разработчиков (и тем бо́льшие, чем фирма известнее) и миллионы их пользователей. Microsoft своей инициативой TWC как бы подает пример остальным компаниям, индивидуальным разработчикам и пользователям, и теперь дело за нами – за мировым сообществом программистов: мы должны эту инициативу подхватить и развивать, во имя нашей общей пользы и безопасности разрабатываемых и используемых нами программ.

Применительно к нам, уважаемые коллеги по ИТ-образованию, это означает, прежде всего, что мы должны учить наших студентов принципам, технологиям и инструментам TWC и их практическому применению.

К сожалению, пока не только в России, но и во всем мире обучение TWC недостаточно распространено. Этим занимаются, главным образом, университеты и колледжи в США и Канаде, ориентированные на военные разработки, – исторически именно военные и близкие к ним организации по вполне понятным причинам испытывают особый интерес к компьютерной безопасности.

Такая ситуация должна быть срочно исправлена – невозможно больше мириться с высококвалифицированными (как приходится признать) атаками хакеров, взламывающих защиту банков, важнейшие сайты, проникающих к нам с недобрыми намерениями через IP-сети, электронную почту и т. д. и при этом наносящих многим из нас ежедневно ощутимый материальный и моральный ущерб. Ответом же на это (отнюдь не только в России), к сожалению, подчас являются лишь общие слова об укреплении безопасности, не подкрепленные достаточными знаниями и адекватными инструментами для борьбы с атаками.

В 2005 г. Microsoft Research объявила конкурс грантов «Trustworthy Computing Curriculum», целью которого была поддержка внедрения принципов и методов обучения TWC в мировую университетскую практику. Мне посчастливилось выиграть такой грант (другими победителями конкурса стали 14 коллег из университетов США). Результатом моей работы в 2006–2008 гг. по проекту TrustSPBU.NET [1], удостоенному гранта Microsoft Research, стало введение в учебные программы и в практику обучения на мат-мехе СПбГУ нового спецкурса «Инженерия надежных

и безопасных программ» (в англоязычном варианте – «Secure Software Engineering») [1, 6, 7] для студентов 4 курса, а также расширение тематики следующих моих курсов принципами TWC:

- «Разработка надежных компиляторов» («Trustworthy Compiler Development») [1, 8, 9] – обновленный и расширенный спецкурс по компиляторам для студентов 5 курса;
- «Надежное и безопасное программирование для платформы Microsoft.NET на языке C# « («Trustworthy Programming for .NET Platform in C#») [1, 10, 11] – обновленный и расширенный спецкурс для студентов 4 курса;
- «Операционные системы и сети» («Operating Systems and Networking») [1, 12, 13] – обновленный и расширенный обязательный курс (64 часа) для студентов 2 курса.

В 2009–2010 учебном году планируется также организация практикума (hand-on labs) по инженерии надежных и безопасных программ, надежным компиляторам, надежному и безопасному программированию для платформ .NET и Java. На этих занятиях студенты будут осваивать на практике столь интересные для них вопросы, как, например, моделирование и отражение сетевых атак, разработка безопасного кода на языках Java и C# и др. Практикум будут вести мои ученики.

Как важный результат и промежуточный итог работы по проекту TrustSPBU.NET (как, впрочем, и по проекту Aspect.NET [1]) в 2008 г. в США опубликована моя монография на английском языке [1] издательством John Wiley & Sons. Она, кроме материалов по АОП и по системе Aspect.NET, содержит также подробное описание моего подхода к преподаванию ИТ – *ERATO* (о нем я уже писал ранее в журнале «Компьютерные инструменты в образовании» [14]), учебные программы всех моих курсов и пояснения к ним. Готовится русская версия книги, а материалы большинства курсов доступны на русском языке. Ссылки на материалы всех моих курсов доступны через сайт моей лаборатории Java-технологии [15], раздел УЧЕБНЫЕ МАТЕРИАЛЫ.

Как уже отмечалось, одной из самых серьезных проблем развития TWC как научного направления является выработка *количественных оценок (quantitative assessment)* безопасности, риска при разработке и использовании программ, надежности программ. В этом отношении следует отметить работы [16, 17], благодаря которым впервые в более чем 50-летней практике программирования становится возможным использование реалистичных и практически ориентированных количественных оценок качеств и свойств программ, основополагающих для TWC. Методы таких оценок основаны на методах математической статистики.

Подчеркнем, что *без количественной оценки характеристик TWC применение принципов TWC в значительной степени теряет как научный смысл, так и практическое значение.*

ВЗАИМОСВЯЗЬ АОП И TWC

В журнале «Компьютерные инструменты в образовании» опубликованы мои работы [18, 19, 20] об аспектно-ориентированном программировании (АОП) и нашей системе Aspect.NET.

Как выяснилось в результате исследований, столь новые, интересные и современные направления, как АОП и TWC, опень тесно, самым коренным образом связаны между собой. Суть реализации TWC (или усиления надежности и безопасности программ) в большинстве случаев состоит в систематических групповых, «сквозных» добавлениях в существующий код каких-либо действий или проверок. Но ведь именно для этого и предназначено АОП!

ПРИМЕНЕНИЕ АОП И ASPECT.NET ДЛЯ РАЗРАБОТКИ НАДЕЖНЫХ И БЕЗОПАСНЫХ ПРОГРАММ

Вопросам применения АОП для разработки надежных и безопасных программ посвящена моя книга [1]. В ней же дан

подробный обзор работ по TWC, АОП и применению АОП для TWC.

Рассмотрим лишь некоторые важные принципы и методы применения АОП для TWC. Общая идея заключается в следующем. АОП обеспечивает возможность систематических групповых вставок или изменений кода, управляемых *аспектами* – специальными модулями кода, предназначенными для *внедрения (weaving)* в целевые программы.

Безопасность (security). Как уже отмечалось, для решения задачи усиления безопасности какой-либо программы, как правило, требуется *добавление проверок безопасности (security checks)*, например:

– *аутентификации* (*authentication*) – проверок имени и пароля пользователя, совпадение которых, как предполагается, гарантируют его аутентичность;

– *авторизации (authorization*) – проверок наличия у пользователя необходимых полномочий для выполнения тех или иных действий, например изменения файла или открытия сокета.

В качестве примера рассмотрим платформу .NET и следующую типичную задачу. Требуется вывести на консоль список всех IP-адресов заданного компьютера в сети (host). Простая программа на языке C#, решающая эту задачу без учета требований безопасности (использующая установки безопасности по умолчанию), может иметь вид [1] (см. листинг 1).

Здесь используются пространство имен *System.Net* для сетевого программирования в .NET и содержащийся в нем класс *Dns* (от *DNS – Domain Name Service*).

Код достаточно прост, но не вполне безопасен.

В соответствии с принципами TWC, безопасный код должен явно запрашивать полномочия безопасности перед выполнением того или иного действия, в данном случае – запроса IP-адресов компьютера.

Представим себе теперь, что уже разработан код сетевой программы на C# (объемом в несколько миллионов строк), в которой подобные действия по запросу IP-адресов, реализованные методом *PrintHostIP,* встречаются буквально на каждом шагу. Однако в первоначальном варианте программы не были предусмотрены меры безопасности – проверки правомочности запроса IP-адреса.

Как теперь быть? Средствами Visual Studio, практически *вручную*, просматривать и модифицировать весь этот огромный код, добавляя там, где необходимо (перед каждым вызовом метода *PrintHostIP*), проверки полномочий? Понятно, что подобное решение может привести к ошибкам: например, если в исходном коде всего 101 вызов метода *PrintHostIP*, то очень велик риск, что при выполнении таких чисто рутинных действий – однотипных многократных вставок кода проверки безопасности – какой-

Листинг 1

```
using System.Net;
public class Utilities
{
   public static void PrintHostIP(string hostName)
   {
      Console.WriteLine
         ("IP address of the host=" +
        Dns.GetHostEntry(hostName).AddressList[0]);
      Console.WriteLine("Press ENTER to exit");
      Console.ReadLine();
   }
   static void Main(string[] args)
   {
      PrintHostIP(Dns.GetHostName());
   }
}
```

нибудь (например, сто первый) вызов метода *PrintHostIP* модифицирован не будет (например, если сотрудника, выполняющего эту работу, срочно вызвал начальник или пригласили попить кофе, после чего он начисто забыл, что не выполнил всю работу до конца). Как предотвратить подобную ситуацию?

Адекватным средством решения этой проблемы является применение АОП и нашей системы Aspect.NET. Опишем *аспект*, который выполнит эту работу автоматически – перед каждым вызовом метода *PrintHostIP* вставит в целевую программу вызов другого метода – *DemandDnsPermission*, проверяющего наличие необходимых полномочий.

Все необходимые для понимания этого примера детали организации системы Aspect.NET и метаязыка АОП Aspect.NET.ML, на котором (в сочетании с языком C#) написан приведенный код, описаны в работах [18–20] в журнале «Компьютерные инструменты в образовании».

Аспект вставляет перед каждым вызовом метода *PrintHostIP* в целевую программу вызов метода *GetHostAction*, определенного в аспекте, который требует от системы наличия необходимых полномочий (вызывая метод *Demand* – отсутствие полномочий приводит к бросанию исключения *SecurityException*), и выдает на консоль трассировочные сообщения, поясняющие ход работы программы.

Самое удобное, что код целевой программы будет модифицирован автоматически с помощью среды Aspect.NET Framework и вкладки *Aspects*. Ни один из вызовов метода *PrintHostIP (*называемый в терминологии АОП *точкой присоединения – join point*) не будет «забыт». Более того, пользователю перед фактическим внедрением аспекта будет предоставлена возможность просмотреть все найденные потенциальные точки присоединения и, если необходимо, отказаться от какой-либо из них.

Но самое замечательное в том, что эти изменения поведения целевой программы (усиление ее безопасности) могут быть произведены, даже если *исходный код целевой программы недоступен*, а имеется лишь ее *двоичный код – сборка (assembly*).

Надеюсь, что этот простой, но содержательный пример достаточно наглядно

Листинг 2

```
%aspect Aspect1
using System;
using System.Net;
using System.Security.Permissions;
class Aspect1
{
   %modules
   private static void DemandDnsPermission() {
      DnsPermission p
    = new DnsPermission(PermissionState.Unrestricted);
   p.Demand();
   }
   %rules
   %before %call *.PrintHostIP(string) && args(..)
   %action
   public static void GetHostAction(string hostname) {
      Console.WriteLine("Hostname="+ hostname);
      Console.WriteLine("Demand DNS permission");
      DemandDnsPermission();
      Console.WriteLine("Demand DNS permission finished");
   } // GetHostAction
} // Aspect1
```

поясняет силу и возможности АОП и системы Aspect.NET для решения задач TWC.

Аналогичным образом путем определения и применения соответствующих аспектов могут быть решены другие типичные задачи TWC [1]:

– *обработка ошибок (error handling)* – обработка исключений или кодов возврата;

– *вставка синхронизирующих вызовов (synchronization calls)* для различных способов синхронизации процессов и потоков;

– *вставки кода, обеспечивающего безопасность выполнения кода в многопоточной среде (multi-threaded safety)*;

– *вставка проверок в стиле design-by-contract* (проверки предусловий, постусловий и инвариантов);

– *вставка трассировочного кода (logging)*;

– *вставка кода, обеспечивающего криптование и декриптование информации, а также проверки полномочности обращения к какой-либо информации для обеспечения ее конфиденциальности (privacy)*;

– для целей *business integrity* – вставка кода, исправляющего заявленные пользователями ошибки или реализующего необходимые расширения функциональности;

– *вставка других необходимых проверок для улучшения надежности программ – например, вставка проверок на ненулевые указатели, вставка проверок допустимых диапазонов для целочисленных аргументов и т. д.*

ЗАКЛЮЧЕНИЕ

Чем больше исследований и практических разработок по TWC ведется в настоящее время, тем более ясно, что значительная часть работы еще впереди. Необходимы, прежде всего, научные основы количественной оценки TWC и удобные методы и инструменты их применения. Необходимы инструментальные средства, обеспечивающие проектирование и реализацию надежного и безопасного кода и предотвращающие появление кода, чреватого уязвимостями и опасного с точки зрения возможных атак. Все мы уже убедились, что гораздо дольше, сложнее и дороже улучшать надежность и безопасность уже находящегося в широком использовании кода «задним числом», чем с самого начала писать правильный, надежный и безопасный код.

АОП – лишь одна из технологий, удобных для TWC. Другим подходом, применимым для TWC, как уже отмечалось, является *инженерия знаний*, более точно – *разработка и практическое использование баз знаний о надежности и безопасности программ и о методах разработки надежного и безопасного кода*. Более простой подход – применение *шаблонов кода* (*code patterns*), позволяющее в некоторых, наиболее простых случаях, избежать ошибок, связанных с безопасностью.

В идеале «умный» программный инструмент сам должен подсказывать пользователю, что, например, код, который он только что набрал в текстовом редакторе интегрированной среды, подвержен атаке «переполнение буфера» (buffer overrun) [3]. Очевидно, что для реализации данной идеи могут понадобиться шаблоны правильного кода, методы инженерии знаний и формализации семантики программ, методы доказательства корректности программ.

Уважаемые читатели, уверен, хорошо понимают, что данная статья – только начало для широких обсуждений на эту важную тему. Инициатива TWC – это «интеллектуальный вызов» и формулировка целого комплекса актуальных проблем, требующих своего решения. Сейчас мы лишь в начале пути к этому.

Отзывы о статье присылайте, пожалуйста, по электронной почте: ***v_o_safonov@mail.ru***.

Литература

1. *Safonov V.O.* Using aspect-oriented programming for trustworthy software development. Wiley Interscience. John Wiley & Sons, June 2008.

2. Web-страницы корпорации Microsoft, посвященные Trustworthy Computing // *http://www.microsoft.com/mscorp/twc/default.mspx*

3. *Howard M., LeBlanc D.* Writing Secure Code. Microsoft Press, 2002.

4. *Howard M., Lipner S.* Security Development Lifecycle. Microsoft Press, 2006.

5. *Swiderski F., Snyder W.* Threat Modeling. Microsoft Press, 2004.

6. *Safonov V.O.* Secure Software Engineering. University Course Curriculum // *http://www.msdnaa.net/curriculum/?id=6753*, February 2007.

7. *Сафонов В.О.* Инженерия надежных и безопасных программ. Спецкурс для студентов 4 курса // *http://www.microsoft.com/rus/msdnaa/curricula*, раздел: «Технология разработки программного обеспечения», ноябрь 2007.

8. *Safonov V.O.* Trustworthy Compiler Development. University Course Curriculum // *http://www.msdnaa.net/curriculum/?id=7698*, December 2008.

9. *Safonov V.O.* Compiler Development.
University course curriculum // *http://www.msdnaa.net/curriculum/?id=5938.*

10. *Safonov V.O.* Microsoft.NET Architecture and the C# Language. University course curriculum // *http://www.msdnaa.net/curriculum/?id=5911*, June 2004.

11. *Сафонов В.О.* Надежное и безопасное программирование для Microsoft.NET на языке C#. Спецкурс для студентов 4 курса // *http://www.microsoft.com/rus/msdnaa/curricula*, раздел: «Технология разработки программного обеспечения», ноябрь 2007.

12. *Сафонов В.О.* Операционные системы и сети ЭВМ. Основной курс для студентов 2 курса специальности 351400 // *http://www.microsoft.com/rus/msdnaa/curricula*, раздел: «Операционные системы», ноябрь 2007.

13. *Safonov V.O.* Operating Systems and Networking. University course curriculum // *http://www.msdnaa.net/curriculum/?id=6006*, December 2004.

14. *Сафонов В.О.* Актуальные проблемы преподавания технологии программирования в России // Компьютерные инструменты в образовании, 2008, № 5. С. 30–36.

15. Сайт лаборатории Java-технологии математико-механического факультета СПбГУ // *http://polyhimnie.math.spbu.ru/jtl /*

16. *Bernstein L., Yuhas C.M.* Trustworthy Systems Through Quantitative Software Engineering. Wiley-IEEE Computer Society Press, 2005.

17. *Sahinoglu M.* Trustworthy Computing. Analytical and Quantitative Engineering Evaluation // Wiley Interscience. John Wiley & Sons, 2007, 320 pp.

18. *Сафонов В.О.* Aspect.NET – инструмент аспектно-ориентированного программирования для разработки надежных и безопасных программ // Компьютерные инструменты в образовании, 2007, № 5. С. 3–13.

19. *Сафонов В.О.* Практическое руководство по системе аспектно-ориентированного программирования Aspect.NET // Компьютерные инструменты в образовании, 2008, № 3. С. 3–16.

20. *Сафонов В.О.* Практическое руководство по системе аспектно-ориентированного программирования Aspect.NET. Часть 2 // Компьютерные инструменты в образовании, 2008, № 4. С. 12–20.

Abstract

The article covers modern approaches to trustworthy computing (TWC). Special attention is paid to tight relationship of TWC and aspect-oriented programming (AOP), to applying AOP and the Aspect.NET toolkit for trustworthy software development [1].

Сафонов Владимир Олегович,
доктор технических наук,
профессор кафедры информатики
СПбГУ, руководитель лаборатории
Java-технологии,
v_o_safonov@mail.ru