# О нашей профессии

Из всей информации в главе “О нашей профессии” наиболее полезными мне показались советы автора начинающим программистам в поиске вакансий. В принципе, можно согласиться с мнением автора, но полностью полагаться на его советы не стоит. Как мне кажется, не всегда следует соблюдать краткость в своём резюме, если ты начинающий программист. Нужно максимально понятно объяснить с чем ты уже сталкивался и какие задачи ты можешь решать, чтобы потом не возникло недопонимание. Когда у программиста появится достаточно большой опыт работы, тогда, наверно, ему уже не обязательно стараться подробно рассказывать о всех своих приобретённых навыках, с опытом он будет понимать, что необходимо указать в резюме, а какая информация будет лишней.

Возможно, я смогу написать эссе о том, каким должно быть хорошее резюме начинающего программиста.

# Можно ли конструировать программы как аппаратуру?

Из всего прочитанного материала наиболее интересной мне показалась тема использования конечно-автоматной модели для программного компонента в софтостроении. Эта модель ,как мне кажется, не самая удобная, и пользоваться ей можно только для решения небольшого класса задач, при этом время отклика компонента оценивается приближённо, а тестирование в большинстве случаев происходит выборочно.

Возможно, я смог бы написать эссе о том, какие задачи удобнее решать, используя конечно-автоматную модель.

# ASP.NET и браузеры

Самой интересной информацией из прочитанного материала мне показалась идея универсального программируемого терминала, которым является веб-браузер, поддерживающий стандарты взаимодействия с веб-сервером. Оказалось сложным разработать приложение, корректно работающего хотя бы под двумя типами браузеров .  
 А тестировать приложение нужно не только под разными браузерами, но и под разными операционными системами. С учётом версий браузеров. Поэтому к началу 2000-х годов в корпоративной среде появился фактический стандарт корпоративного веб-приложения: Internet Explorer 6, который не плохо выполнял свои функции, до того как приходило время переходить на новую версию браузера. Тогда выяснилось, что их новые версии далеко не всегда совместимы с находящимися в эксплуатации системами. И по этой причине простое обновление Internet Explorer 6 на 7 могло вызывать паралич информационных систем предприятия.

Возможно, я написал бы эссе о том, какие браузеры не стоит использовать в корпоративной среде для веб-приложений.

# ООП – неизменно стабильный результат

До того как я прочитал этот материал, я не видел никаких недостатков в ООП. Мне казалось, что ООП – это самое удобный и вообще правильный способ программирования. Раньше я не задумывался о тех проблемах, которые могут возникнуть, если использовать ООП в больших проектах. Действительно ведь возникает много трудностей: невозможность быстро разобраться в чужом коде, наличие многих неявных зависимостей, человеческий фактор становится ключевым риском проекта. На практике объектно-ориентированный подход в большинстве случаев превращает проект или продукт, переваливший за сотню-другую тысяч строк, в “Ад Паттернов ”, который, никто без помощи авторов развивать не может.

Возможно я смогу написать эссе о том, почему ООП так быстро стало популярным.

# Прогресс неотвратим

Из прочитанного материала я решил выделить наблюдения автора за некоторыми продуктами Microsoft. В целом все они критикуются. Во многом, наверное, можно согласиться с этой критикой. Например стремительные обновления NET ставит в трудное положение разработчиков, которым приходиться преодолевать появляющиеся риски и тратить средства на переобучение персонала. Обновленная версия SQL 2005-го года оказалась не доработанной, приложения стали работать медленнее. Windows Vista не оправдала ожиданий и оказалась невостребованной с появлением “семёрки”, так и не успев занять сколь-нибудь значительную долю парка “персоналок” и ноутбуков. Microsoft Office критикуется за смену своего интерфейса, который, как мне кажется, не такой уж плохой, чтобы требовать возможность его изменять.

Возможно я смогу написать эссе о том, какие трудности возникают после обновления программного обеспечения.

# Слоистость и уровни

Наиболее полезной информацией из прочитанного материала мне показалась классификация слоёв и уровней корпоративной информационной системы, что важно знать не только разработчикам, но и тем программистам, которые будут наращивать на неё свои приложения. Есть несколько точек зрения, с которых можно рассмотреть любую автоматизированную информационную систему. В них можно выделить слои, кроме этого даже в самых простых программах различают как минимум два уровня. Если мы наложим слои системы на её уровни, то получим достаточно простую матричную структуру, позволяющую бегло оценить, какой из элементов необходимо реализовать своими силами или же адаптировать уже имеющийся готовый.

Возможно я смогу написать эссе о том, как определять необходимые слои корпоративной информационной системы.

# Ultima-S – КИС из коробки

Наиболее полезным мне показался рассказ про КИС Ultima-S, над которой трудился автор. Интересно было узнать о двух основных подходах в разработке КИС, а именно «от производства» и «от бухгалтерии». Ultima-S являлась примером подхода «от бухгалтерии» как и многие другие разработки КИС того времени . Эта система функционально была хорошо обеспечена для своего времени, но не получила широкого применения всех своих технических возможностей, которое наступило спустя десять лет в других системах. Главной проблемой проекта Ultima-S и многих других, по мнению автора, было отсутсвие маркетолога, определяющего вид продукта и программу продвижения, что сделало бессмысленным большое количество технологических действий.

Возможно я смогу написать эссе о том, почему не стоит выбирать маркетолога главой компании по разработке программного обеспечения.

# Журнал хозяйственных операций

Из прочитанного материала я решил выделить раздел про учётные приложения, в котором рассказывается про такой антишаблон как «таблица остатков». Суть его заключается в том, что приложение использует лишь остатки, не принимая во внимание историю их изменения, что приводит к некоторым сложностям. Надёжнее использовать в приложении журнал операций и вместе с этим хранить вычисленные остатки. Хранить таблицу остатков следует для того, чтобы не ждать окончания их расчётов, которые могут затянуться.

Возможно я смогу написать эссе о том, какие есть известные антипаттерны в разработке учётных программ.

# Архитектура сокрытия проблем

Из прочитанного материала я решил выделить рассказ автора, где главным героем была крупная фирма-разработчик, которая занималась созданием ERP-систем для розничной торговли. Через какое-то время у компании появились проблемы разрастания баз данных: время отклика как интерактивной работы, так и пакетов увеличилось, производительность понемногу деградировала. Найти грамотное решение не получилось, смена СУБД на более мощную и более производительное «железо» не привело к результату. Спустя долгое время попыток найти выход из сложившейся ситуации, компания была поглощена холдингом Cegid за свою некомпетентность в области СУБД.  
  
Возможно я смогу написать эссе о том, какие проблемы могут возникать, при сильном разрастании базы данных.

# Тесты и практика продуктового софтостроения

Из прочитанного материала я решил выделить рассказ автора, про модульное тестирование. Начинается всё с причин, по которым использовать модульные тесты для тиражируемых веб-приложений нецелесообразно. Наверное, главные причины: сильное затруднение рефакторинга и отсутствие необходимости выпускать обновления слишком часто. Была затронута интересная проблема: в некоторых случаях соотношение тестирующего кода значительно превышало объёмы рабочего. В отличие от функциональных тестов, завязанных на интерфейсы подсистем, модульные тесты требуют переработки одновременно с рефакторингом рабочего кода. Это увеличивает время на внесение изменений и ограничивает их масштаб.

Возможно я смогу написать эссе на тему: использование модульного тестирования в разработке программного обеспечения.

# Программная фабрика

Из прочитанного материала я решил выделить тему: программная фабрика, в которой рассказывается про инструменты CASE , позволяющие генерировать код, скомпилировав который, можно сразу получить работающее приложение. Первые CASE инструменты, выросшие из редакторов графических примитивов, были представлены в 1980-х годах. Можно заметить некоторые недостатки CASE-средств:   
1) Если ручное написание кода принять за максимальную гибкость, то CASE может навязывать стиль кодирования и шаблоны генерации частей программ.  
2)CASE работает только при условии, что ручные изменения генерируемого кода исключены или автоматизированы.  
В качестве решения перечисленных проблем появились так называемые двусторонние CASE-инструменты, позволяющие редактировать как модель, непосредственно видя изменения в коде, так и, наоборот, менять код с полу или полностью автоматической синхронизацией модели.

Возможно я смогу написать эссе на тему: эффективность использования средств генерации кода в разработке программного обеспечения.

# Что-то с памятью моей стало

Из прочитанного материала я решил выделить тему с забавным названием “Что-то с памятью моей стало”, в которой рассказывается про утечку оперативной памяти сервера. Программа, создавшая эту проблему, забирала около одного гигабайта памяти в сутки, причём никак не возвращая ее обратно даже после отключения всех пользователей. Таких неприятностей можно было избежать, если бы в процессе разработки ПО проводилось тестирование. Но в результате совместных усилий проблему решили, хоть и остались некоторые другие.

Возможно я смогу написать эссе на о том, как багом программы могут воспользоваться злоумышленники.