«Почему техника ломается», или Что такое забытое понятие «надежность» (заблудившимся специалистам и наивным заказчикам посвящается)

Сложная система, спроектированная наспех, никогда не работает, и исправить ее, чтобы заставить работать, невозможно.

(16-й закон системантики — Законы Мерфи)

Владимир КОВАЛЕВ
notes@novus-lab.ru

Для кого эта статья? В первую очередь для специалистов, так как именно от них зависит, в каком «электронном будущем» нам жить завтра. И наступит ли оно вообще (шутка, навеянная фильмами-катастрофами). И во вторую очередь нормальным людям, тем, кто ежедневно пользуется техническими новшествами нашего суперсовременного мира и решил воплотить свои технические идеи, заручившись поддержкой специалистов. Господа заказчики, поймите — все не так просто, как кажется на первый взгляд, но все решаемо при грамотном подходе к делу.

Что побудило меня написать данную статью? Отвечаю: все чаще мне приходится повторять избитую истину — «Если хочешь сделать что-то хорошо, сделай это сам». НЕ ХОЧУ! Хочу, чтобы каждый на своем месте выполнял работу качественно. Чтобы бытовое Audio/Video не нужно было таскать в гарантийный ремонт после месяца эксплуатации, чтобы самолеты не падали по причине отказа бортовой аппаратуры, чтобы мобильные телефоны не «глючили», а уж о программном и аппаратном обеспечении компьютеров — вообще лучше промолчать, чтобы промышленное оборудование требовало замены по моральному, а не физическому износу и т. п. То, что первопричина многих катастроф и аварий — отказы технических систем, секретом не является, и большинство их происходит по причине халатности или неправильных действий персонала или вследствие конструктивных недоработок. Сталкиваясь в последние годы с некачественными «высокотехнологичными» изделиями, причем не только в области бытовой электроники, но и в сфере промышленной автоматики, как-то начинаешь себя не совсем уютно чувствовать в нашем мире, буквально напичканном разными «электронными штучками».

А стоит ли вопрос ухудшения качества разработок радиоэлектронных средств (РЭС) вообще? Вопрос простой, но ответ на него неоднозначный. С одной стороны, техника стала намного сложнее и функциональнее, чем в недалеком прошлом (до эры появления микроконтроллеров, гигабайтных массивов Flash-памяти, всевозможных кодеков, массовых ЦАП/АЦП с фантастическими характеристиками и т. п.). С другой стороны, я как разработчик частенько удивлялся, глядя на схему, а порой и на вышедшее из строя устройство (иногда и производства именитых фирм). Я с трудом верил, что не самые лучшие схемотехнические и конструктивные решения может позволить себе фирма-разработчик, а не кустарь-самоучка. Налицо в большинстве случаев халатность (или неграмотность) разработчиков. Оно и понятно: рыночная экономика требует буквально «выстреливать» продукты на рынок. Многие из нас не понаслышке знакомы с BSOD (Blue Screen Of Death) ОС Windows, для большинства «глюк» мобильного устройства или DVD-плеера — само собой разумеющееся, и если телевизор вдруг начал истошно пищать, то проще и дешевле его выбросить и купить новый, чем вызывать мастера. Да и потом: «Зачем переживать, все равно через год-другой приобрету что-нибудь поновее и «покруче»!».

Таковы основы рынка общества потребления. Соответствующим философии потребления и стал подход к проектированию. Изделия разрабатываются на короткий жизненный цикл (два-три года). О какой надежности тут можно говорить? Да, с моей точки зрения, изделия электронной техники стали гораздо менее надежными, просто пользователи стали более «продвинутыми» и зачастую многие «некритичные» отказы устраняют сами или решают проблему заменой вышедшего из строя устройства. Но кто сказал, что это правильно? По-моему, перефразируя известный из «законов Мерфи» «принцип IBM»: «Машина должна работать — человек думать», пора сказать: «Машина должна РАБОТАТЬ. Чтобы человеку осталось время подумать». В данной статье я неоднократно буду ссылаться на эти законы. Конечно же, «законы Мерфи» нужно воспринимать с известной долей юмора, но попробуйте задуматься и взглянуть на них еще раз, ведь, как известно, «в каждой шутке есть доля шутки».

Итак, давайте попробуем разобраться, как такой регресс в надежности техники оказался возможен при почти фантастическом прогрессе в области электронных компонентов и технологий. Причина носит комплексный характер. Во-первых: снизилась квалификация разработчиков, во-вторых: увеличилась

сложность проектируемых технических систем, в третьих: спешка (успеть вперед конкурентов), которая, как известно, «нужна в двух случаях». Это, конечно, мое мнение, но оно опирается на опыт общения с другими разработчиками, а также на мой повседневный опыт, как обычного человека, так и специалиста. В данной статье я попробую обосновать свои соображения.

Для начала определим, из чего же складывается понятие надежности технического (радиоэлектронного) изделия:

1) Надежность схемотехнических решений.
2) Надежность конструкционных решений.
3) Надежность элементной базы.
4) Надежность технологических процессов при сборке изделия.
5) Надежность программного обеспечения.
6) Надежность пользователя.
7) Если речь идет о комбинированных системах (например электро/электронно-механических), таких как высоковольтное и другое силовое электрооборудование, автоматизированные линии, дизель-генераторные станции и т. п., то тут еще добавляются такие слагаемые, как:
   – качественные пусконаладочные работы;
   – своевременное техническое обслуживание;
   – качественные расходные материалы.

Но п. 7 — это уже информация не для разработчика, а скорее для пользователя.

Пожалуй, все. Назовем эти перечисленные выше пункты «макрокомпонентами» или «макроопределениями».

Я не зря определил каждый «макрокомпонент» как «надежность», так как, на мой взгляд, любой из этих пунктов вносит свой вклад в общую надежность (а точнее ненадежность) системы наряду с другими ее компонентами, такими как конкретные детали и элементы конструкции. Причем все эти составляющие (разве что кроме п. 5 — «надежность программного обеспечения») всегда присутствовали. Но почему же уровень надежности разрабатываемой аппаратуры резко упал? Не в том ли причина, что недостаточно квалифицированные программисты не могут написать адекватно работающее ПО? Да, отчасти так, но основная причина, на мой взгляд, в том, что изделия стали более сложными. Вспомните иностранное слово complex, так вот и проблему надежности нужно решать комплексно, причем учитывая то, что все «макрокомпоненты» порой настолько тесно переплетаются между собой, что не поймешь, когда кончается схемотехника и начинается конструирование.

Где граница окончания работы конструктора и начало работы дизайнера или технолога, и кто должен больше участвовать в разработке принципиальной схемы — схемотехник или программист? Как же быть? Вывод прост — нужно понимать все, с чем прямо или косвенно имеешь дело. Разработчик должен быть Инженером (именно так — с большой буквы) и должен обладать достаточно широким кругозором и хотя бы элементарными познаниями в смежных областях. Теперь, к сожалению, такие специалисты встречаются все реже и реже. А ведь именно разработчик с глубокими познаниями в своей области и широким кругозором имеет больший шанс на ведение работ в «правильном» ключе, так как в этом случае он ясно представляет себе проект целиком (или хотя бы «близлежащие области»). Именно это позволяет ему избегать многих «подводных камней», которые могут пустить ко дну весь «корабль».

Тут нужно понимать, что, конечно, всего знать невозможно (да и не нужно), но следует совершенно четко представлять себе, с чем мы имеем дело, — весь проект в совокупности, хотя бы на уровне «функциональной схемы», а не конкретную «узкую» его часть. Многие могут возразить, что для этого есть руководитель проекта, чего тут за него думать («…у лошади голова большая — пусть она думает»). Да, есть руководитель проекта, и от него зависит многое, но гораздо больше зависит от разработчика или разработчиков. И если это действительно квалифицированный специалист с широким кругозором, то шансы выпустить качественный продукт в нужные сроки резко возрастают. А уж об административных проблемах руководителя в этом случае можно не беспокоиться — их будет немного. В конечном счете, все выиграют, устройство будет разработано должным образом, в указанные сроки и с минимумом недоделок. Недостаточное количество таких «комплексных» разработчиков и является, на мой взгляд, одной их истинных причин брака в проектировании.

А теперь не будем «забегать вперед» и «пройдемся» по пунктам более внимательно и последовательно. Да, чуть не забыл, естественно предполагаем, что работы начинаются с грамотного ТЗ, в котором глубоко и точно описаны основные требования к надежности, условиям эксплуатации, ЭМС, «климатика», условия и методы контрольных испытаний и прочие «прелести». К несчастью, сегодня многие начинают работу с чего угодно, но только не с ТЗ. В итоге не ясно, что должны получить на выходе, как это оценивать и испытывать, какие должны быть промежуточные контрольные стадии. Кстати говоря, ТЗ, в том или ином виде, всегда присутствует, но чаще всего — только в голове разработчика (хорошо, если он один). А, как известно, «голова — предмет темный и исследованию не подлежит», и за решением текущих технических вопросов неизбежно что-то забывается. Я не говорю уже о том, что при работе коллектива над проблемой отсутствие общего ТЗ просто недопустимо.

Итак:

1) Надежность схемотехнических решений.

Первый и, пожалуй, один из наиболее важных аспектов. Здесь все «карты в руки» разработчику. Уж здесь-то есть где разгуляться. Как будет решен вопрос надежности в данном случае — целиком и полностью зависит от квалификации и пристрастий разработчика. Позволю себе отметить, что практически любая техническая проблема имеет более одного решения, и какое из них лучшее — вопрос риторический. Тут нужно руководствоваться суровой действительностью (например, какие технологии и комплектующие заказчику «по карману», разумны ли сроки поставок комплектующих и т. п.) и трезвым инженерным расчетом, которым, кстати, многие пренебрегают, полностью полагаясь на чужие решения и «даташиты» производителей, используя их в своих проектах буквально «под копирку». Такой подход мне кажется абсолютно неправильным. Мало того, что все люди ошибаются (мне неоднократно приходилось находить неточности в «фирменной» документации и схемах), так многие горе-разработчики еще и тиражируют ошибки и непроверенные решения, не утруждая себя элементарными расчетами. Отдельно упомяну и специальные схемотехнические и конструкционные методы повышения надежности, такие как резервирование, минимизация количества механических и электромеханических компонентов, контактных соединений, оптимизация температурного и нагрузочного режимов, экранирование, фильтры и развязки и т. д. Существует ряд других эффективных мер, и разработчик должен их использовать при необходимости.

2) Надежность конструкционных решений.

Не менее важный, чем схемотехника, аспект, особенно при проектировании радиочастотных, высокоскоростных, импульсных и силовых схем (а таких теперь большинство). Сегодня печатная плата с цепями, где «пробегают» сигналы мегагерцовых частот и импульсы с длительностями в наносекундном диапазоне, — рядовое явление. Печатная плата давно стала таким же схемотехническим элементом, как и все остальные. Недаром хороший схемотехник, как правило, — неплохой проектировщик печатных плат, ибо эти два аспекта конструирования тесно переплетены. Причем опыт как в одной, так и в другой области зарабатывается долгим и упорным трудом (вспомним А. С. Пушкина: «И опыт — сын ошибок трудных…»). Вопрос проектирования надежных печатных плат — отдельная тема, и интересующийся специалист может почерпнуть массу материала по этому поводу в специальной литературе, а также на сайтах ведущих производителей печатных плат. Идем далее: ЭМС, влагозащита, тепловые и механические (удары, вибрации и т. п.) расчеты, стандартные конструктивы и т. д. Все эти области довольно непростые, и для грамотной разработки устройства требуется труд коллектива квалифицированных специалистов, так как разработчика с адекватными познаниями во всех этих об-

ластях найти непросто, да и работу он будет выполнять долго. Вот зачастую проектировщики и «прокатывают», что называется, эти пункты, авось, и так сойдет. В результате — имеем то, что имеем (см. начало статьи).

3) Надежность элементной базы.

Очень важная и одновременно очень простая по своей сути, но не простая по трудоемкости задача. Сейчас разработчику настолько «вольготно» живется в плане выбора элементной базы, что вопрос, кажется, и выеденного яйца не стоит. На рынке комплектующих — чего только душа не пожелает. Главное здесь внимательность и аккуратность. Очень внимательно читаем «даташиты» производителей и выбираем элементную базу в соответствии с инженерными расчетами, не забывая, конечно, «держать связь» с конструкторами-технологами на производстве, слушаем их пожелания, и тогда, наверное, все будет хорошо. Работа долгая и нудная, особенно в свете обилия комплектующих, и множества вариантов их выбора (причем не всегда равнозначных). Хорошим подспорьем в этом случае служит опыт и «информационная вооруженность» разработчика, а также помощь специализированных изданий, в которых часто в сжатой форме освещаются последние достижения электронной промышленности. Но труды не пропадут даром. Поверьте — отсутствие рекламаций от заказчика и проблем при производстве стоят потраченного на это времени.

4) Надежность технологических процессов при сборке изделия.

Это, пожалуй, самый беспроблемный участок, так как большинство «правильных» сборочных производств работают по самым современным технологиям, на оборудовании лучших мировых производителей и с качественно обученным обслуживающим конструкторско-технологическим персоналом. Хорошее сборочное производство можно рассматривать как «черный ящик». На вход даем комплектующие — на выходе получаем готовое изделие или полуфабрикат для следующей технологической операции. Что делается «внутри», в большинстве случаев нас мало волнует, хотя грамотный разработчик, конечно же, будет контактировать с конструкторами сборочного производства для обеспечения прохождения проекта «как по маслу», да и учесть пожелания конструкторов-технологов в процессе разработки схемы и конструктива — лишним не будет.

5) Надежность программного обеспечения.

Современное электронное устройство все чаще снабжается «умной» начинкой. Цифровые сигнальные процессоры, микроконтроллеры, программируемая логика (даже программируемые аналоговые схемы) — вот далеко не полный список компонентов, требующих для своей работы специализированного программного обеспечения (ПО). Нужно ли говорить о том, что без него это груда ни на что не способного «железа»? О возможностях современных программируемых компонентов знают многие разработчики, поэтому без лукавства могу смело утверждать, что зачастую половина схемы устройства — это программа. И от того, как выполнена эта программа, зависит порой напрямую не только надежность работы устройства, но и «жизнь» этого самого устройства, или, что еще хуже — здоровье и жизнь людей.

Надежность ПО. Все чаще эти понятия рассматриваются как антиподы. Я не буду пускаться здесь в дискуссию по поводу, как лучше писать программы, какими языками пользоваться и нужна ли вообще блок-схема. Отмечу лишь, что моду на «сырое» ПО и последующие многочисленные «заплатки» ввел… (нет — не многострадальный Билл Гейтс). Моду задал рынок, точнее конкуренция в условиях современного рынка, которая диктует необходимость «выбросить» (это слово тут как нельзя лучше подходит) продукт раньше других. Когда это касается программного продукта — тут все понятно (хотя я и не согласен с таким подходом), но, к сожалению, данная мода распространилась и на ПО встроенных систем. Как вы уже, наверное, поняли, доля программируемых компонентов в современных разработках весьма высока. Высока и цена ошибки.

Вопрос создания качественного ПО очень непростой и уж точно «не помещается» в рамки данной статьи. Рассмотрим «последствия».

Вся опасность ошибок ПО заключается в том, что никто не знает, что они есть, а в соответствии с «законами Мерфи» — «В любой программе есть ошибки»:

- «Самая грубая ошибка будет выявлена, лишь когда программа пробудет в производстве, по крайней мере, полгода».
- «Указание начинающему программисту: если вы с первого раза сумели написать программу, в которой транслятор не обнаружил ни одной ошибки, сообщите об этом системному программисту. Он исправит ошибки в трансляторе».
- «Если бы строители строили здания так же, как программисты пишут программы, первый залетевший дятел разрушил бы цивилизацию».

Как ни печально — но это так (правда, строители уже вплотную приблизились к программистам).

Схемотехнические и конструкционные ошибки, если не «лежат на поверхности», то по крайнее мере довольно быстро проявляют себя, находятся и устраняются, хотя зачастую и более дорогим (в материальном смысле) путем. Ошибки программные — более изощренные. Они могут «жить» в программах годами, периодически «отравляя» пользователю и разработчику жизнь. Отловить их зачастую бывает сложнее, чем написать новую программу, и самое неприятное, что заниматься этим после сдачи устройства в эксплуатацию особенно никто не стремится. А последствия печальные.

Вот яркие примеры некоторых программных ошибок:

- Устройство работает не совсем по тому алгоритму, как задумывалось. В принципе все работает (до поры до времени), но результаты работы устройства с трудом укладываются в рамки ТЗ.
- Все работает отлично, но иногда устройство делает «финт ушами» и ведет себя непредсказуемым образом.
- «Еще вчера все работало нормально»…
- Типичная ошибка оператора (пользователя) приводит к критическому отказу системы.
- Данные в энергонезависимой памяти (например, статистика за год работы) оказываются поврежденными или неверными.
- Устройство «не заводится» или не выключается «нормально» (так, как должно) или делает это «через раз».
- И т. д, и т. п.

Отсюда вывод — программные ошибки, как и болезни, проще предупредить, чем лечить, а посему советую разработчикам-программистам браться за написание ПО самым ответственным образом и со всем тщанием, правда, немногие разработчики понимают, зачем же это нужно. Деградация программистов, на мой взгляд, значительно превышает деградацию схемотехников-конструкторов. Современные системы «визуального программирования» настолько избаловали разработчиков всевозможными «шаблонами», «визардами» и готовыми библиотечными процедурами, что редко какой программист вообще сможет САМ написать что-то стоящее (эпоха виртуозных программистов уже прошла). Программы лепят «из кубиков» как попало, не задумываясь, как и где их можно применять. Такой подход напоминает художника, пишущего свои полотна методом вырезания и наклеивания кусков работ других художников. Потом где-то «подкрасил», «подклеил» и готово. О какой надежности ПО в этом случае можно вообще говорить? Все та же спешка в разработке: рынок требует…

6) Надежность пользователя.

Начну данный абзац с нескольких цитат (законы Мерфи):

- «Создайте систему, которой сможет пользоваться и дурак — и только дурак захочет ею пользоваться».
- «Если существуют два способа сделать что-либо, причем один из которых ведет к катастрофе, то кто-нибудь изберет именно этот способ».
- «Можно сделать защиту от дурака, но только от неизобретательного».

Противоречие? Скорее дилемма для разработчика, но, к счастью, решаемая. Не будем бросаться в крайность и ориентироваться на «изобретательного идиота» или пациента кли-

ники им. П. П. Кащенко. Возьмем, что называется, «среднего пользователя». Прежде всего, он — человек, следовательно, тут как тут пресловутый «человеческий фактор». Да, все делают ошибки, но как их избежать? Да никак. А вот снизить их вероятность и минимизировать наносимый ими вред — задача для настоящих интеллектуалов! Ну что, поехали?

1. Вопрос эргономики и дизайна. Да-да, вот так. А кто сказал, что будет легко? Пора вспомнить «основы художественного конструирования» (это совет для тех, кому их преподавали), ну а для остальных поднапрячься и вспомнить пару эпизодов из своей жизни, которые приукрашивают крепким выражением вроде: «…Руки бы выдернуть тому, кто это сделал!» Какова же здесь роль разработчика (допустим, схемотехника или программиста, не говоря уже о конструкторе)? Как ни странно, она довольно весомая. Ведь от того, как расположены органы управления и индикация устройства, каково количество и цвет «лампочек» на передней панели, насколько информативны текстовые и мнемонические сообщения и т. д., и т. п., на самом деле очень многое зависит. Ведь устройство спроектировано зачастую для того, чтобы с ним работали люди, и чем более «дружелюбно» оно будет по отношению к своему более высокоразвитому «товарищу», тем меньше этот самый «товарищ» сделает ошибок при работе. «Умное» устройство «простит» оператору его неправильные действия или просто не допустит их, и, как следствие, тот самый «человеческий фактор» оказывается не таким уж фатальным. Про дизайн изделия вообще распространяться не буду, и так ясно, что он не менее важен. Вопрос пользовательского интерфейса ПО также полностью соответствует этому пункту, да к тому же последнее время все больше и больше устройств выпускаются с «виртуальными» панелями управления (особенно портативные мультимедийные устройства, смартфоны и т. д., да и в область «серьезной» аппаратуры, например промышленной, такие новшества проникают все больше).
2. Вопрос схемотехники. Странно? Ничуть. Устройство должно быть разработано таким образом, чтобы неправильные действия пользователя не привели бы к тяжелым последствиям, как для самого устройства, так и для пользователя. Тут все зависит от фантазии разработчика и технических требований, регламентирующих надежность устройства и безопасность его использования.
3. Вопрос конструктивного исполнения — решает задачи, полностью аналогичные п. 2.

Угодить «на всех и каждого» — задача довольно трудная, если не сказать неосуществимая, но здесь разработчик в разных ипостасях (и как схемотехник, и как конструктор, и как программист или дизайнер) как раз и должен проявить свою проницательность и живость ума. Ведь как я и предупреждал — это развлечение для настоящих интеллектуалов.

Все-таки п. 7 (см. начало статьи) тоже можно рассмотреть как условие для разработки. Конечно, схемотехник тут вряд ли внесет весомый вклад, а вот конструктор и дизайнер — вполне. Как раз именно с последующей технической эксплуатацией изделия связано множество проблем пользователя. Для примера можно привести такие распространенные явления, как неудобная компоновка устройства, когда оно становится практически неремонтопригодным, недостаток контрольных точек, отсутствие поясняющих надписей и должной маркировки на печатной плате, неудачное расположение клеммных колодок, отсутствие нормальной конструкторской и пользовательской документации и прочая, и прочая, и прочая… Все эти недостатки, если учесть, что работают с этими устройствами люди, вполне могут при монтаже, ТО или эксплуатации спровоцировать появление отказов.

Так как же обеспечить надежность при проектировании РЭС? Ключ к ответу — комплексный подход к решению вопроса надежности на всех этапах проектирования. Какие же должны быть способы обеспечения надежности? Казалось бы, здесь все просто: каждому свое, как говорится, «богу — богово, кесарю — кесарево», то есть схемотехник обеспечивает надежность на своем фронте, технолог на своем, программист пишет надежное ПО и т. п. Да, конечно, но не нужно забывать, что все эти «макрокомпоненты» надежности (схемотехнический, конструкторский, технологический и т. п.) настолько тесно «переплетены» между собой, что грамотный разработчик просто обязан быть сведущим во всех этих областях.

Примеры? Пожалуйста!

- Схемотехник должен распределять внешние ресурсы (выводы) микроконтроллера (или ПЛИС), советуясь с программистом.
- Программист обязан писать программу с пониманием того, какие сигналы (их амплитудные, временные и частотные параметры) должна обрабатывать микросхема, для которой он пишет ПО.
- Дизайнер должен знать состав и функциональность органов управления и индикации и функциональность пользовательского интерфейса ПО.
- Конструктор обязан быть в курсе особенностей схемотехники данного устройства для успешной разводки печатной платы и компоновки всего устройства.

Попробуйте придумать еще пару-тройку «перекрестных связей», уверяю — вы без труда это сделаете.

Таким образом, мы подошли к тому, что, во-первых, вопрос обеспечения надежности РЭС при разработке — вещь сложная, многогранная и крайне ответственная. И, во-вторых, под силу это действительно специалистам с широким кругозором (таких еще можно встретить на необъятных просторах нашей Родины, но их все меньше и меньше). Причина тому — ориентированность многих молодых инженеров на западный «узкоспециализированный» стиль работы. Иногда это допустимо, но, на мой взгляд, от такого подхода больше вреда, чем пользы. Про «халяву» и проектирование методом copy-paste с чужих разработок вообще не говорю, таких инженеров нужно просто не допускать до работы. Но это уже совсем другая тема.

Я попробовал собрать воедино основные, как мне кажется, проблемы обеспечения надежности РЭС при разработке. Некоторые из них «вскользь» упомянуты в учебниках «Основы конструирования РЭС», но лишь упомянуты. Основной упор в учебных пособиях сделан на расчет надежности, основанный на анализе структурных схем, схемотехнических и конструктивных решений и учете надежности элементной базы. Да, все это замечательно, но почему-то подразумевается, что разработчик идеален. Практически нигде не рассматривается взаимосвязь определенных в данной статье «макрокомпонентов», вероятно, оттого, что дать им количественную оценку — невозможно, а, следовательно, бесполезны будут и аналитические выражения с ними. Но разработчик, вооруженный представлением, из чего же складывается надежность РЭС, должен решать эту проблему на стадии проектирования, чтобы эти «макрокомпоненты» имели «минимальный вес» при оценке надежности конечного изделия.

Цель данной статьи — напомнить заказчикам и разработчикам о сложности и актуальности проблемы обеспечения надежности РЭС, о ее комплексном характере. Решать эту проблему следует с самого начала — на всех стадиях проектирования, и спешка тут совершенно ни к чему, особенно с учетом возросшей сложности разработок. В заключение проведите простой тест (совет для разработчиков): доверили бы вы системе, которую разработали, какое-нибудь ответственное дело, и чтобы все видели какую-нибудь табличку, что разработал сие чудо «такой-то и такой-то» с вашей фотографией и подписью? Думаю, далеко не все могли бы действительно гордиться своими разработками. А ведь раньше инженер, который строил мост, во время испытания частенько стоял под ним…

Все. Если я что-то не осветил — не обессудьте. Дать универсальные рекомендации всем и каждому невозможно по определению. Здесь каждый наступает на свои «грабли».

Теперь осталось «переварить» всю эту информацию. «Ответ на первый известный русский вопрос — «Кто виноват?» — я попробовал дать, на второй — «Что делать?» — ищите в следующей публикации «Что такое «не везет» и как с ним бороться, или Обеспечение надежности РЭС при разработке». ■