Много приходится читать и обсуждать разные стандарты кодирования, ограничивающие применение тех или иных конструкций языка (goto, множественное наследование классов в C++) или приемов программирования (рекурсия, динамическое выделение памяти после инициализации приложенияПод катом немного юмора и серьезных рассуждений о применении различных практик ведения проектов.  
  
Как утверждается, эти стандарты позволяют писать более чистый и легко читаемый код. Они позволяют избегать некоторых видов ошибок кодирования, и облегчают поддержку программного продукта.  
  
Однако, на мой взгляд, незаслуженно мало внимания уделяется тому, как члены команды взаимодействуют друг с другом. Слаженная работа команды, эффективная коммуникация между людьми не менее важна для успеха проекта. Недопонимание, неправильная интерпретация сказанного, трудно понимаемые комментарии в исходном коде в конечном итоге сказываются на качестве программного продукта.  
  
Для того, чтобы заполнить этот пробел, предлагаю ввести 10 правил, которые бы позволили бы избежать наиболее досадных ситуаций, происходящих из неясного изложения своих мыслей:

1. Не использовать предложения длиннее 15 слов.
2. Не использовать более трех определений к одному существительному.
3. Не использовать сложносочиненные предложения.
4. В каждом предложении должно быть подлежащее и сказуемое.
5. В одном предложении не допускается более одного деепричастного оборота.
6. Должен быть подготовлен список из 5000 наиболее распространенных слов русского языка и профессиональных терминов. Слова и аббревиатуры, не входящие в этот список, могут быть употреблены только по согласованию с руководством проекта.
7. Названия конкурирующих организаций могут употребляться только негативном контексте. В письменной речи упоминание конкурирующей организации должно сопровождаться пояснением в скобках: «(конкурирует с нами, контакты с этой организацией должны быть ограничены)».
8. В рабочее время не допускается обсуждение политики компании. В мире много несправедливости, и в том числе несправедливость может исходить от нашей компании. Но это не означает, что компания должна оплачивать время, потраченное на обсуждение этой несправедливости.
9. В случае недопустимого с точки зрения этих правил общения руководство проекта должно быть извещено в письменной форме.
10. Табуированную лексику разрешается использовать только для ясности и четкости изложения своих мыслей. Табутрованная лексика не допускается:
    * Если ее применение приводит к нарушению предыдущих 9 пунктов.
    * Для выражения своего отношения к содержанию предыдущих 9 пунктов.

А теперь абсолютно серьезно.  
  
Есть правила, имеющие универсальное и повсеместное применение. Это уголовный и гражданский кодексы, правила техники безопасности и пр. Соблюдение этих правил иногда позволяет избежать определенного рода неприятностей, но эти правила никоим образом не являются инструкцией, как достичь успеха.  
  
Есть также книжки и психологические курсы о том, как быстро разбогатеть. Но мало кто воспринимает это всерьез. Не существует сколько-нибудь серьезно воспринимаемых правил или инструкций, гарантирующих успех проекта или успех в личной жизни.  
  
Означает ли это, что к стандартам кодирования, разного рада практикам (SCRUM, Continues Integration, Code Review и пр.) не стоит относиться серьезно? Вовсе нет. Но надо понимать, что ни одна из практик в IT не является универсальной. Ни одна практика не приводит к гарантированному повышению эффективности проекта. Только накопленный опыт, интуиция и профессионализм менеджеров и ведущих разработчиков помогает принять правильное решение в каждой конкретной ситуации. Иначе бы наша профессия не была настолько трудной и интересной.  
  
Если какой-то менеджер говорит о необходимости использования юнит тестов и аргументирует это красивыми диаграммами, убеждающими, что стоимость бага в геометрической прогрессии зависит от того, на какой стадии развития проекта этот баг найден – бегите из этого проекта или из этой компании. Я ничего не имею против юнит тестов. Однако, решение о том, использовать их или нет, и как их реализовывать, должно исходить из специфики данного проекта. Вот неполный перечень вопросов, на которые надо ответить прежде, чем принять решение:

* Насколько эффективными могли бы быть юнит тесты при разумных затратах на их реализацию?
* Есть ли другие способы тестирования приложения, и в какой степени они могли бы взять на себя те цели, которые преследуются юнит тестами?
* В какой степени юнит тесты необходимы для эффективного управления качеством продукта (quality assurance)?
* Есть ли необходимые средства в бюджете проекта?
* Как имплементация юнит тестов повлияет на сроки реализации проекта и в какой степени это критично?
* Есть ли консенсус в команде по этому вопросу? Если его нет, то что приоритетнее: сохранение мотивации отдельных членов команды или необходимость настоять на том решении, которое менеджеры считают правильным?

Если мне надо что-то разрубить, я ищу топор. Но если у меня есть топор, и я хорошо умею им пользоваться, то это вовсе не означает, что я буду бегать и искать, что бы разрубить. Точно также и с правилами/практиками в программировании. Если я хорошо владею какой-либо практикой, я не буду требовать, чтобы ее применяли повсеместно.