Доклад о сложности ПО

(Урусов А.О. и Кузин К.В.)

*Врач, строитель и программист поспорили о том, чья профессия древнее.*

*Врач заметил: "В Библии сказано, что Бог создал Еву из ребра Адама. Это мог*

*сделать только хирург, поэтому моя профессия самая древняя в мире". Его*

*перебил строитель: "Однако, как сказано в Книге Бытия, еще раньше Бог сотворил*

*из хаоса сначала небо, а потом — землю. Это было первое и, несомненно,*

*наиболее впечатляющее строительство. Следовательно, дорогой доктор, вы ошибаетесь.*

*Именно моя профессия самая древняя в мире". Услышав это, программист*

*откинулся на спинку кресла, улыбнулся и спросил доверительным тоном: "Ну а кто*

*же, по-вашему, создал хаос?"*

В нашем докладе речь пойдет о сложных программных продуктах в промышленном масштабе. Это тысячи и десятки тысяч строк кода, огромное количество взаимодействующих элементов, скрещивание различных технологий, очень обширные предметные области. Доклад о том, почему такое ПО – сложное и о способах эту сложность свести к минимуму.

ПО – это сложная система, как, н-р, дерево, устройство компьютера, опорно-двигательная система человеческого тела, система образования и прочие. Однако, что именно делает ПО сложным?

1. Самое главное – вы сами. Человеческий мозг устроен так, что максимальное количество объектов, за которыми он способен следить – не более 8-9, а чтобы осознать и закрепить какой-либо факт, мозгу требуется от 2 до 5 секунд. Это обусловлено ограничениями кратковременной памяти. Отсюда следует, что человек не способен быстро и легко понять или разработать систему, в которой одновременно взаимодействует большое множество элементов. Именно поэтому сложные системы разрабатываются в команде, а в процессе проектирования принимают участие несколько архитекторов, которые могут иметь разные точки зрения на то, как должна быть построена та или иная часть системы. Отсюда появляется новая проблема - проблема управления командой разработки, ведь необходимо согласование действий каждого архитектора с каждым.
2. Вторая проблема такого ПО – слишком широкая предметная область, в которой, порой, плохо разбирается даже сам заказчик. Также бывают ситуации, когда заказчик и исполнитель по-разному понимают одну и ту же проблему. После прохождения начальных этапов, заказчик может переосмыслить собственный заказ и поменять требования. Кроме исполнителя и заказчика система зависит и от пользователя. ПО такого рода стоит дорого и менять его при каких-либо нововведениях экономически невыгодно. Потребитель привыкают к продукту, становятся зависимыми от него. Это говорит о том, что необходимость обеспечения гибкости системы изначально является важным и значимым требованием. Программу нужно спроектировать так, чтобы можно было легко обеспечить её эволюционирование, дополнить без вреда для уже работающего кода.
3. Состояние приложения в момент времени описывается набором состояний переменных - это дискретная система. Чаще всего сложное ПО опирается на реальные объекты. Их сложно описывать, потому что в дискретных системах все очень зависит друг от друга и небольшие изменения в исходных данных могут сильно повлиять на выходны.
4. Сложному ПО свойственна иерархическая структура с множеством уровней абстракции, каждый из которых в свою очередь может представлять собой другую систему, тоже сложную. Здесь очень важно изначально определить элементарные компоненты, на которых будет работать нижний уровень абстракции. Проблема в том, что выбор этих компонентов – штука субъективная. Наблюдатель произвольно выбирает, какие элементы считать основными.

Итак, мы столкнулись с серьезной дилеммой. Сложность систем программного обеспечения возрастает, а способности справиться с этой сложностью остаются ограниченными. Как же выйти из этого логического тупика? Существует несколько методов и рекомендаций о том, как упростить процесс проектирования сложного ПО. Их нужно использовать комплексно.

Фактически, все сводится к нескольким принципам построения структуры приложения:

1. Нужно рассмотреть систему с двух точек зрения. Как part-of и как is-a. Проанализировав эти 2 представления, необходимо уменьшить избыточность, определить слабые и сильные стороны и уделять особое внимания им. Определить наилучшие подходы, шаблоны, модули.
2. Алгоритмическая и ООП декомпозиция.
3. Модульность, сокрытие информации. Связи внутри компонентов сильнее. Общая структура – повторное использование.

Для анализа поможет абстрагирование. Это расширения возможностей человека. Устойчивые промежуточные формы. Поведение целого сложнее, чем поведение суммы его составляющих

Задача разработчиков программных систем — создать иллюзию простоты.

*"Для проектирования новой структуры необходим полет фантазии, а также синтез опыта и знаний, которые художник воплощает на холсте или бумаге. Сначала инженер должен стать художником и сформулировать свой замысел, а затем превратиться в ученого и тщательно проанализировать проект, применяя строгий научный метод"*