В двух приведенных материалах профессор Эдвард Ли, высказывает свое мнение по поводу таких фундаментальных методов в разработке программного обеспечения, как моделирование кибер-физических систем и потоки. Оба материала получились очень интересными и как по мне действительно важными как с научной, так и с практической точки зрения.

Начну с доклада Эдварда на тему наука и техника для кибер-физических систем. В своем выступлении Ли рассказал, как к нему обратилась компания с просьбой сделать их компьютерную симуляцию более точной и приближенной к реальности, Эдвард посоветовал им разработать физическую модель конечного продукта и тестировать её. Это действительно очень полезное и простое решение. Возможно многие люди привыкли что при тестировании необходимо работать с виртуальной моделью, которая представляет собой программу, или её урезанного в возможностях клона, и для разработки ПО это действительно так, но это совсем не работает с реальными объектами. При создании подобного цифрового двойника необходимо учитывать очень много факторов, например время, погода, скорость интернета, возможные перебои. Также стоит учитывать, что двойник пишется на определенной ОС, и будет тестироваться на одном железе, а в будущем работать совсем на другом. Здесь возникает сразу ряд проблем – необходимо учесть все эти факторы, их необходимо обработать (если они произойдут их нужно вывести), тестируя двойника нужно понимать, что мы тестируем именного его в данной реализации с данным софтом на данном железе. Ошибки и неточности накапливаются, что ведет плохой точности и некорректным результатам. Вместо этого можно создать реальную симуляцию нашей вещи (наделив её только нужными нам для тестирования свойствами) и проводить тестирование на ней. Благодаря этому мы сможем как проверить наш рабочий прототип в деле и учесть все физические особенности реализации вещи, так и протестировать её программно как изначально и хотели.

Во втором материале (статье) Эдвард привел свою критику потокам. При реализации параллельных программ основным средством взаимодействия принято (де-факто) использовать потоки, однако при более детальном анализе данной концепции, и сравнении с другими методами, данная модель по мнению профессора sucks. Потоки делают программу недетеренированной (хаотичной), что является грубым противоречием самой парадигмы программ и алгоритмам. Применяя концепцию потоков, мы начинаем избавляться от неопределённостей в задаче. У этого есть ряд недостатков:
Необходимо тщательно анализировать программу и искать места общего доступа потоков к ресурсам
Могут возникать дедлоки и гонки
Необходимо следовать шаблонам многопоточных программ
Иногда данные шаблоны приходится комбинировать
Все это создает ряд проблем и делает многопоточное программирование действительно сложным. Есть и другие методы межпоточного взаимодействия, (рендеву, сети процессов, дискретные события, mpi), однако т.к. они не являются напрямую частью языка их необходимо реализовывать самому, из за чего они так не популярны. Решение Эдварда изобрести координационный язык – язык, который расширяет синтаксис базового и определять только те конструкции, которые нужны для удовлетворения его координационных целей. Благодаря этому люди продолжат писать на своих любимых языках, а их функционал расширится и решит данную проблему. Мне кажется это единственное реальное решение. Люди привыкли писать на своих «базовых» языках программирования и не особо хотят переходить на другие. Расширить функционал языка не сильно меняя его синтаксис – красивое решение.

Для меня оба материала были полезны и интересны, однако если доклад о тестировании был полезен с практической точки зрения, то материал про параллельные программы был более научным и новаторским. Он заставил меня углубится в методы межпроцессного взаимодействия, их нюансы и альтернативы.