Введение

Вопросы параллельной разработки сложных систем, в особенности космической техники, технологического оборудования и электроники, в настоящее время становятся чрезвычайно актуальными. Особенностью разработки таких систем является одновременное участие большого количества различных предприятий, а иногда и нескольких стран. При этом возникают проблемы согласования элементов систем, их интеграции, сборки и испытания, а также синхронизации процессов разработки и обеспечения информационного обмена между разработчиками.

Очевидно, что все эти проблемы требуют оперативного и эффективного решения. Однако подходы к их решению могут (и должны) быть различными.

Например, можно проанализировать требования разработчиков, найти "узкие места" в их работе, и создать необходимые технические средства для решения выявленных задач. С другой стороны, можно проанализировать процесс разработки в целом, выделить его основные черты, и, исходя из этого, строить систему поддержки этого процесса.

Конечно, оба подхода имеют право на жизнь, поскольку первый решает задачи сегодняшнего дня, а второй - определяет конечную цель и направление движения.

В данной статье предлагается относительно простая модель процесса параллельной разработки, которая может быть использована при построении системы поддержки этого процесса. Необходимость создания такой системы очевидна, поскольку без привлечения современных технических средств, обеспечить управление и обмен информационными потоками, возникающими в процессе параллельной разработки сложных систем, практически невозможно.

Модель процесса параллельной разработки

Сложность процесса параллельной разработки объясняется большим количеством участников этого процесса и интенсивным взаимодействием между ними. Однако, если взглянуть на этот процесс не с точки зрения взаимодействия его участников, а именно как на процесс, то его сложность оказывается кажущейся: процесс разработки легко разделяется на отдельные подпроцессы и в целом образует четкую иерархическую структуру. Причем такое разделение действует на всех уровнях системы и является универсальным.

Каждый процесс можно рассматривать как решение некоторой конкретной задачи, возникшей в ответ на существующие проблемы. Для решения задачи формируется некоторая группа разработчиков и снабжается необходимыми техническими средствами. При этом надо отметить, что организационная структура предприятий может не совпадать со структурой процесса разработки. Поэтому члены одной и той же группы разработчиков могут одновременно участвовать в решении других задач.

При параллельной разработке одновременно выполняются сразу несколько процессов, и чем сложнее система, тем этих процессов больше. Поскольку процессы выполняются практически изолированно, в них могут возникать противоречивые решения. Чтобы исключить накопление таких решений необходимо время от времени гармонизировать проект, т.е. выявлять противоречия и определять пути их разрешения.

Исходя из этого, можно выделить две чередующиеся фазы процесса разработки: фазу разработки и фазу гармонизации. Схематично это показано на рис. 1.

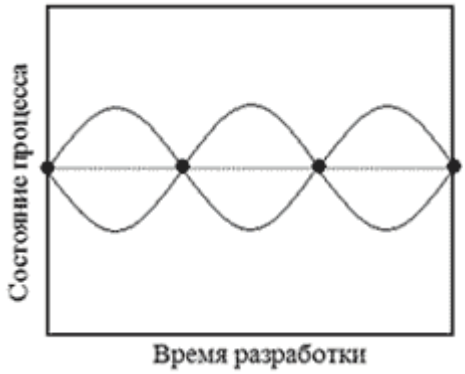

Рис. 1. Две фазы процесса параллельной разработки сложных систем

Фазу разработки схематично можно представить несколькими параллельными и независимыми потоками работ, а фазу гармонизации - точкой слияния этих потоков.

Важно также отметить, что каждый отдельный поток работ имеет точно такую же внутреннюю структуру и может быть разделен на отдельные потоки более низкого уровня.

Очевидно, что обе фазы процесса разработки требуют определенных трудозатрат. Причем интуитивно понятно, что от частоты точек слияния существенно зависит эффективность процесса разработки в целом.

Если слияние будет происходить слишком часто, проработанность подсистем окажется недостаточной. В итоге, придется анализировать многие промежуточные решения, которые, при более детальной проработке подсистем могли бы быть отсеяны. С другой стороны, слишком редкие точки слияния также снижают эффективность разработки поскольку увеличиваются непродуктивные затраты на проработку решений, неприемлемых с точки зрения всей системы.

Таким образом, существует явный оптимум по частоте точек слияния. Чтобы определить положение этого оптимума необходимо выяснить основные закономерности процесса разработки.

Для этого рассмотрим конкретный пример разработки программного обеспечения (ПО).

Процесс разработки ПО легко контролируется и поддается анализу. В то же время, нет оснований полагать, что этот процесс чем-то качественно отличается от любого другого процесса. Поэтому использование принципа аналогии здесь можно считать допустимым.

На рис. 2 представлена диаграмма роста кода программы по времени ее разработки. На диаграмме видны характерные точки излома, в которых скорость процесса разработки скачкообразно изменяется. Это связано с тем, что в этих точках происходит переход от решения одной задачи к другой.

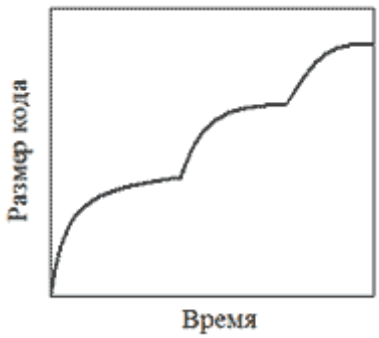

Рис. 2. Динамика роста кода программы в процессе ее разработки

В этих точках принимается решение о том, что и как надо делать дальше. Сразу после этого процесс разработки идет очень быстро, пока не обнаружатся новые проблемы. Замедление темпов роста происходит на этапе отладки программы, т.е. на этапе поиска и устранения противоречий. Здесь могут быть периоды, когда скорость разработки падает с 50-100 строк кода в час до 1-2 строк в сутки.

Из рассмотренного примера видно, что процесс разработки программы протекает в три фазы:

A. выбор пути разработки;
B. разработка;

C. анализ, верификация, доводка;

Фаза (A) является наиболее ответственной, поскольку от правильности выбора пути зависит эффективность процесса разработки. Если путь будет выбран неверно, вся последующая работа может оказаться выполненной впустую. Но, как показывает практика, заключение о правильности выбора можно сделать только после того, как решение будет реализовано. Поэтому здесь приходится полагаться на прозорливость разработчиков и на удачу. Для снижения риска принятия неверных решений желательно иметь возможность обсудить это решение с разработчиками подсистем и с архитекторами системы.

Фаза (B) менее ответственна, поскольку она незначительно влияет на продолжительность процесса и протекает сравнительно автономно. Здесь обсуждений с коллегами почти не требуется, поскольку большинство решений, принимаемых на этом этапе, относится к внутренней реализации подсистемы.

Фаза (C) требует больших трудозатрат на поиск возможных противоречий в подсистеме, но не изменяет ее облика. Поэтому эта фаза протекает практически без обмена информацией с внешним миром.

Теперь предположим, что весь этот процесс протекает внутри другого процесса разработки сложной программной системы и зададимся вопросом - на каком этапе разработки целесообразно производить слияние подсистем. Ответ очевиден - после завершения фазы (B). До этого момента принятое решение еще не приобрело конкретных очертаний и его невозможно анализировать. Ждать же завершения фазы (C) нецелесообразно, поскольку облик подсистемы уже сформирован, и его дальнейшая проработка не вносит в него качественных изменений. Более того, если данное решение было принято неверно и его придется отвергнуть, то трудозатраты на его верификацию и доводку окажутся в списке непроизводительных расходов и снизят эффективность

разработки в целом. Поэтому очень важно с максимальной точностью уметь определять переход от фазы разработки к фазе анализа и доводки.

Из рисунка 1 следует, что для прогноза времени завершения фазы (B) необходимо отслеживать динамику процесса разработки. Для этого можно измерять информационные потоки, циркулирующие в среде разработки. Применительно к программному обеспечению - это может быть поток символов кодов программы. Для конструкторских проектов - количество выпускаемых чертежей, число элементов этих чертежей и т.п. По-видимому, нет принципиального значения, какой параметр отслеживать √ главное, чтобы он был достаточно информативен, легко измерялся и обеспечивал наилучшую точность прогноза.

Для грубой оценки времени завершения фазы (B) можно использовать эмпирический закон 20:80, который гласит, что 80% работы делается за первые 20% времени, а остальные 20% - за оставшиеся 80%. Следовательно, аппроксимируя зависимость накопленной в системе информации от времени функцией:

$$N = N_0\left(1 - \exp\left(-\frac{t}{\tau}\right)\right)$$

где t √ время разработки; τ√ постоянная времени; N_0 √ максимальный объем информации при достижении стабильного состояния разработки, можно легко спрогнозировать время окончания фазы (B) и наступления сроков точки слияния.

Однако при параллельной разработке моменты времени завершения фазы (B) у каждой подсистемы будут различными. По логике, необходимо дождаться завершения процессов разработки всех подсистем, однако это также может привести к неэффективности проекта. Чтобы избежать этих потерь, необходимо анализировать информационные потоки разработки не отдельных подсистем, а всей системы в целом и строить прогноз именно на основании этих суммарных потоков.

Таким образом, анализируя информационные потоки в процессе разработки систем можно прогнозировать дальнейший ход этого процесса и планировать положение точек слияния, обеспечивающих максимальную эффективность разработки системы в целом.

1. Зиндер Е.З. Новое системное проектирование: информационные технологии и бизнез-реинжиниринг // Системы управления базами данных. √ 1995, ⊥4.- с. 50-58.