alexus: *Тот язык хорош и удобен, на котором можно легко (ясно и формально) выразить смысл (уровня системы, в том числе). Благодаря этой тенденции появляются термины языка, то есть слова, в которых умещается целое понятие, представление, набор свойств/качеств и т.п. Но удобство использование терминов зависит от их "интуитивной" понятности. А это уводит в сторону словообразования... правил использования алфавита, то есть, к смыслу букв. Но кому это сегодня интересно...*

Владислав Жаринов: *Вот попробовать бы и определить облик класса языков для каждого уровня... хотя бы в виде "лексика-правила"...*

Прежде всего, я не считаю себя специалистом в вопросе создания языков, поэтому не следует относиться к данному сообщению, как к какой-то рекомендации, это не более, чем размышления на тему.

Схемы

Наверное, не следует начинать работу по формированию языков с «чистого листа». Существует большое количество «языков систем», которые стандартизованы и вполне успешно применяются десятилетиями в самых разных отраслях знания и производства. Можно рассмотреть, например, такие технические «языки», как электронные/электрические/гидравлические схемы, схемы инженерных коммуникаций и т. п. Все эти и многие другие схемы описывают соответствующие системы/подсистемы используемые при создании/обслуживании зданий и сооружений, оборудования, транспортных средств и т. д. Нетрудно увидеть, что по своей сути, эти «языки схем/систем» очень похожи между собой, и содержат два вида сущностей: элементы и связи... то есть, то, что необходимо и достаточно для описания [относительно простых, 2-3 уровней] систем. Отличия между этими языками незначительны и касаются, в основном, условных обозначений элементов и связей. Итак, отметим общие моменты, характерные для схем в различных отраслях:

1. Графическое представление;
2. Формальность;
3. Декларативность.

Поскольку системы низкоуровневые, то их общей характеристикой можно считать относительно небольшое количество типов элементов (от единиц до сотен) и связей (обычно в пределах десятка). При этом элементы и связи являются простыми, выполняющими одну-две однотипные функции, пропускающими один-два типа рабочего сигнала/рабочего наполнения (электроэнергии, жидкости, газа и т.п.).

Каждому типу элементов на схеме соответствуют своё изображение (образ) и каждый элемент, отображаемый на схеме, имеет уникальное обозначение. Часто, но не всегда, обозначения элементов и связей сопровождаются указанием рабочих и/или предельных параметров (5.0V, 220В, 4 атм. и пр.).

Помимо схем, рабочая документация на систему/подсистему включает в себя спецификации. В спецификации для каждого элемента и связи указывают соответствующие типы/модели, отмечают общее количество данных элементов, возможные варианты замен, габаритные и/или весовые характеристики и другие параметры. Наличие спецификаций разгружает схему, делает её чтение более простым и удобным.

В проектной документации, помимо перечисленных схем и спецификаций, могут быть представлены расчётные материалы, которые содержат формулы, расчётные данные и результаты расчётов, на основе которых подбирались конкретные типы элементов и связей, указанных на схемах и в спецификациях.

Аналогичные схемы создаются и при проектировании программных систем. Типичным представителем могут служить ER-диаграммы (entity-relationship diagram), IDEFx, ErWin/BPWin и пр., применяемые при проектировании баз данных. В том же ряду стоят и диаграммы, создаваемые с помощью UML. Однако следует отметить, что во

всех перечисленных программных инструментах схемы совмещены со спецификациями, то есть, элементы схем содержат не только изображение, но и (полный) набор атрибутов/свойств, что загромождает схему, затрудняет её чтение и восприятие. Для схем баз данных характерно наличие только одного типа элемента: отношение/таблица и только одного типа связи: внешнего ключа (справедливости ради надо отметить, что сейчас появилась возможность задавать и другие виды связей, например, ассоциации, но применяются они довольно редко). Для схем на UML, наоборот, характерно огромное количество разных типов элементов (едва ли не каждый элемент схемы имеет уникальный тип/класс), и большое разнообразие связей. Такое положение дел не является специфичным для программирования, это следствие низкой культуры проектирования, которая, хочется надеяться, со временем повысится.

Достоинства и недостатки схем

Широкое распространение схем в различных технических областях связано с их наглядностью и простотой. Как правило, чтение схем не вызывает трудностей даже у тех, кто не является специалистом.

Недостатки схем, как обычно, являются продолжением их достоинств. Схему легко понять, если количество элементов и связей относительно невелико (в пределах 100 элементов). При увеличении количества элементов и связей, восприятие схемы существенно затрудняется. Если количество элементов переваливает за 1000, то разбор логики схемы занимает много времени и представляет собой отдельную задачу. Большие схемы разбивают на логически самостоятельные части. Иногда создают общее представление о логических частях (формируют над-уровень, не путать с верхним уровнем системы), где показывают взаимодействие частей между собой с последующей детализацией каждой части (вертикальное деление схем).

Трудности для чтения схемы нарастают и при увеличении функциональных возможностей элементов. Каждая новая функция требует новых связей, по которым происходит обращение к ней. Схемы становятся всё более пересечёнными, а значит более трудными для восприятия. Примерами многофункциональных элементов могут служить современные микросхемы, обладающие большим количеством входов и выходов. Улучшения восприятия схем, содержащих многофункциональные элементы, добиваются за счёт горизонтального разделения схем. Горизонтальное разделение схемы основано на функциональной декомпозиции, то есть, для каждого входа (каждой функции) системы создаётся своя под-схема, отражающая логику работы схемы, связанную с данным входом. Аналогично могут создаваться под-схемы для каждого выхода (реакции) системы.

Вертикальные и горизонтальные деления схем позволяют незначительно понизить трудности восприятия больших схем, но они не могут принципиально решить проблему увеличения количества элементов и их многофункциональности/многосвязности, поскольку качественное решение проблемы возможно только при формировании следующего уровня системы.

Создание нового языка, ориентированного на разработку систем, должно учитывать многолетний мировой опыт по формированию языков схем, поскольку программные системы — это частный случай систем, и развитие программных систем, включая инструментальные и языковые средства, происходит в рамках общих системных правил и закономерностей.

Формирование языков последующих уровней систем

Системы могут создаваться в порядке убывания уровней (сверху-вниз) или в порядке возрастания уровней (снизу-вверх). Проектирование «сверху-вниз» характерно для тех систем, которые уже имеют готовые аналоги, а, следовательно, сформирована и языковая терминологическая среда. Например, ставится задача создать новый самолёт. На основании исследований формируются лётно-технические (тактико-технические) характеристики. Под эти характеристики подбираются типы фюзеляжа, несущих плоскостей, двигателей, авионики и пр. В случае, если нужных характеристик добиться не удаётся, проводятся новые исследования, ставятся задачи перед теми коллективами, которые разрабатывают соответствующие материалы, устройства, агрегаты, подсистемы. Цикл повторяется до тех пор, пока не будет найдено нужное решение или не будут изменены лётно-технические характеристики.

Создание принципиально новых систем используя метод «сверху-вниз» невозможно, поскольку нельзя задать начальные характеристики неведомо чего. Принципиально новые системы создаются интуитивно, посредством нового взгляда на предметную область, но не на проблемы существующих систем. Новые системы, как правило, предлагаются и создаются непрофессионалами, поскольку профессионалы погружены в решение частных проблем у существующих систем или систем, создаваемых на основе существующих. Выйти за рамки проблем системы данного уровня, очень непростая задача, связанная с ломкой представлений и сменой парадигм. Принципиально новые системы не имеют своего языка и принятой терминологии. Язык и терминология формируются и развиваются вместе с системами. Однако появление систем более высокого уровня может сильно повлиять на языки и терминологию, принятые у систем более низких уровней, включённых в новую систему. Эти изменения связаны с тем, что меняется точка зрения на системы нижних уровней, расширяется их предметная область (область применения) и даже структура ролей, в которых выступали элементы низкоуровневых систем. Отсутствие готовых языков выносит создание новых систем методом «снизу-вверх» за рамки заявленной темы.

Язык системы более высокого уровня почти не отличается (по семантике) от языка систем предыдущих уровней, поскольку на любом уровне, структура системы — это элементы и связи. Однако терминология и условные обозначения у систем разных уровней различаются. Изменения в языках у систем разных уровней связаны, прежде всего, с тем, что с повышением уровня системы происходит её отдаление от исполнительского уровня. При прохождении сигнала от верхнего уровня к нижнему (равно как и обратного сигнала) через несколько уровней происходит его многократное преобразование, а значит вносятся задержки и «шумы». Чтобы повысить уверенность в том, что сигнал дойдёт до исполнителя, система должна запоминать текущее состояние, входящих в неё элементов (которые, в свою очередь, тоже являются системами), и, при необходимости, влиять на изменение состояний своих элементов так, чтобы они могли выполнить поставленную перед ними задачу. Однако запомненное состояние, может отличаться от реального состояния элемента. Для поддержания соответствия, система может опрашивать элементы о их состоянии через какие-то (фиксированные) промежутки времени или перед тем, как поставить перед элементом очередную задачу (выдать новое задание). Другой путь поддержания соответствия заключается в оповещении системы её элементом тогда, когда происходит изменение его состояния. Оба варианта поддержания соответствия между запомненным и реальными состояниями имеют свои плюсы и минусы, рассмотрение которых в данном сообщении не излишне. Сейчас важно только то, что в рассмотрение теперь попадают не только элементы с их функциями и характеристиками, но и состояния элементов и связей. Система и сама обладает состояниями, которые являются логическими функциями состояний её элементов. Состояния системы влияют

на её поведение, на механизмы выработки реакций. Для быстрого вычисления состояний и траекторий перехода в нужное состояние (через некоторое подмножество промежуточных состояний) система должна иметь развитый логический аппарат, которому необходима языковая поддержка. А значит, помимо функциональных языковых средств, в язык необходимо вводить средства исчисления предикатов. И помимо функциональных схем, язык должен позволять создавать логические схемы, определяющие процессы перехода системы из одного состояния в другое.

Дальнейшее повышение уровней систем несомненно потребует и новых языковых средств, помимо тех, что перечислены выше, их можно специфицировать, но на современном уровне перечисленных требований к языкам вполне достаточно.

Основная цель этого сообщения состояла в том, чтобы показать, что

1. На любом уровне системы, применяемые языковые средства, имеют одну и ту же семантическую основу: элементы и связи;
2. В различных областях человеческой деятельности применяются схожие графические описания систем, и данная однородность языковых средств не случайна, а исходит из семантики систем;
3. Повышение уровней системы приводит к одинаковым изменениям в языковых средствах, независимо от области применения систем.
4. Требования к языковым средствам достаточно просты от функциональных на исполнительских уровнях до логических на уровнях управления и целеполагания, что, конечно, тоже не случайно.

Александр Усов.
Екатеринбург,
26.01.2012.