к содержанию

УДК 004.415.25

А. В. НОВОСЕЛЬЦЕВ

МЕТОД СОЗДАНИЯ ГРАФИЧЕСКИХ ИНТЕРАКТИВНЫХ МНЕМОСХЕМ С ИСПОЛЬЗОВАНИЕМ ИДЕИ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ PhAB (ОСРВ QNX 4.25)

ВВЕДЕНИЕ

Мнемосхемы традиционно используются для более удобной и эффективной работы оператора; важно то, что мнемосхема не только предоставляет информацию о значениях технологических параметров, но и дает возможность сразу увидеть и оценить картину происходящих одновременно процессов в целом. Моментальное отслеживание отсутствия или наличия нештатных ситуаций, аварийных и предаварийных состояний, ориентировочные уровни и состояния всех (или большинства основных) параметров процесса сразу, визуализация этапов протекания процессов, возможность моделирования технологического процесса и поведения аппаратуры в штатных и в нештатных ситуациях — это то, что выгодно отличает представление информации в виде интерактивных мнемосхем от простого документирования параметров процессов и их «ручного» просмотра.

В основе технологии традиционных мнемосхем заложено управление графическими свойствами (цвет, размер, положение и т. п.) отображенного на мнемосхеме технологического оборудования или специальных сигнализаторов. Интерактивность мнемосхемы (сигнализация об аварии, вращение частей механизмов, движение жидкостей и носителей по трубопроводам, изменение цвета значения параметра при достижении им какого-либо порога и, соответственно, изменение других частей мнемосхемы, зависимых от этого параметра) — служит для более понятного и простого отображения работы технологической установки или процесса, для привлечения внимания оператора к какому-либо отклонению от штатного функционирования технологического процесса.

В SCADA-системах интерактивность мнемосхем обеспечивается процессом динамизации (иногда его еще называют анимацией). Определение термин **динамизация** (в системе MasterSCADA) — это установление соответствия между значением переменной проекта и значением свойства элемента мнемосхемы. Сам термин происходит от слова «динамика» в смысле движения, изменчивости, нестатичности. В этом смысле будем употреблять этот термин: функция динамизации — функция, позволяющая «оживить» элемент мнемосхемы, придать ему динамизм, изменчивость во времени, не нарушая логики причинно-следственных связей всей мнемосхемы. SCADA-системы — это программные пакеты, предназначенные для разработки и обеспечения работы в реальном времени систем сбора, обработки, отображения и архивирования информации об объекте мониторинга и управления. Из определения видно, что SCADA-система – это больше, чем просто построитель мнемосхем. Предлагаемый в настоящей статье метод позволяет

разрабатывать достаточно сложные мнемосхемы технологических процессов без использования SCADA-системы.

ЦЕЛЕВОЕ НАЗНАЧЕНИЕ МЕТОДА

Рассматриваемый метод предназначен для создания элементов интерактивной мнемосхемы, а именно процессов динамизации элементов мнемосхемы в зависимости от состояния связных с ними элементов и процесса визуализации изменения графических свойств элементов мнемосхемы. Метод основывается на принципах автоматного программирования, с созданием программных спецификаций и предлагает нотацию для документирования процесса разработки проекта с применением описаний АТД (абстрактный тип данных) или блок-схем, диаграмм состояний и переходов. Метод использовался в тестовых проектах в операционной среде QNX 4.25 с использованием среды PhAB (Photon Application Builder).

На рис. 1 приведен пример мнемосхемы технологического процесса. Проведя расчет количества блоков элементов, на которые может быть декомпозирована мнемосхема, и точек разграничения этих блоков, получим, что число блоков, которые объединяют участки трубопроводов, составляет около 50, а количество блоков, представляющих технические средства, более 30. Блоки, представляющие технические средства, могут иметь количество входов и выходов более чем один. Количество точек, разграничивающих эти блоки, составляет более ста.

Корректно визуализировать такую мнемосхему, учитывая более ста основных состояний, без применения специализированных SCADA-систем, представляется достаточно сложной задачей. Но применяя методы, основанные на идее автоматного программирования, методы декомпозиции и разбиения задачи на составляющие подзадачи, разрабатывая спецификации для каждого блока элементов и для каждой точки разграничения этих блоков, вполне возможно построение такой интерактивной мнемосхемы в среде PhAB операционной системы реального времени QNX 4.25 без применения SCADA-системы.

ОБЩЕЕ ОПИСАНИЕ ПРЕДЛАГАЕМОГО МЕТОДА

Мнемосхема используется для контроля и управления реальным, уже созданным объектом и должна отражать логику состояния и управления этим объектом. Также мнемосхему можно применять для моделирования технологических процессов, моделирования поведения объектов управления. Обычно при создании мнемосхем их составляют из несколько упрощенных технологических схем, в которые вводятся различные элементы и надписи, с сохранением графического подобия взаиморасположения технологических линий. Естественно предположить, что и визуальное отображение технологического процесса на мнемосхеме в точности соответствует реально происходящим процессам в объекте контроля и управления. Причинно-следственные связи, объективно присущие реальным технологическим процессам, должны в точности отображаться в мнемосхеме как в последовательности процессов отображения элементов, так и во временном масштабе их отображения.

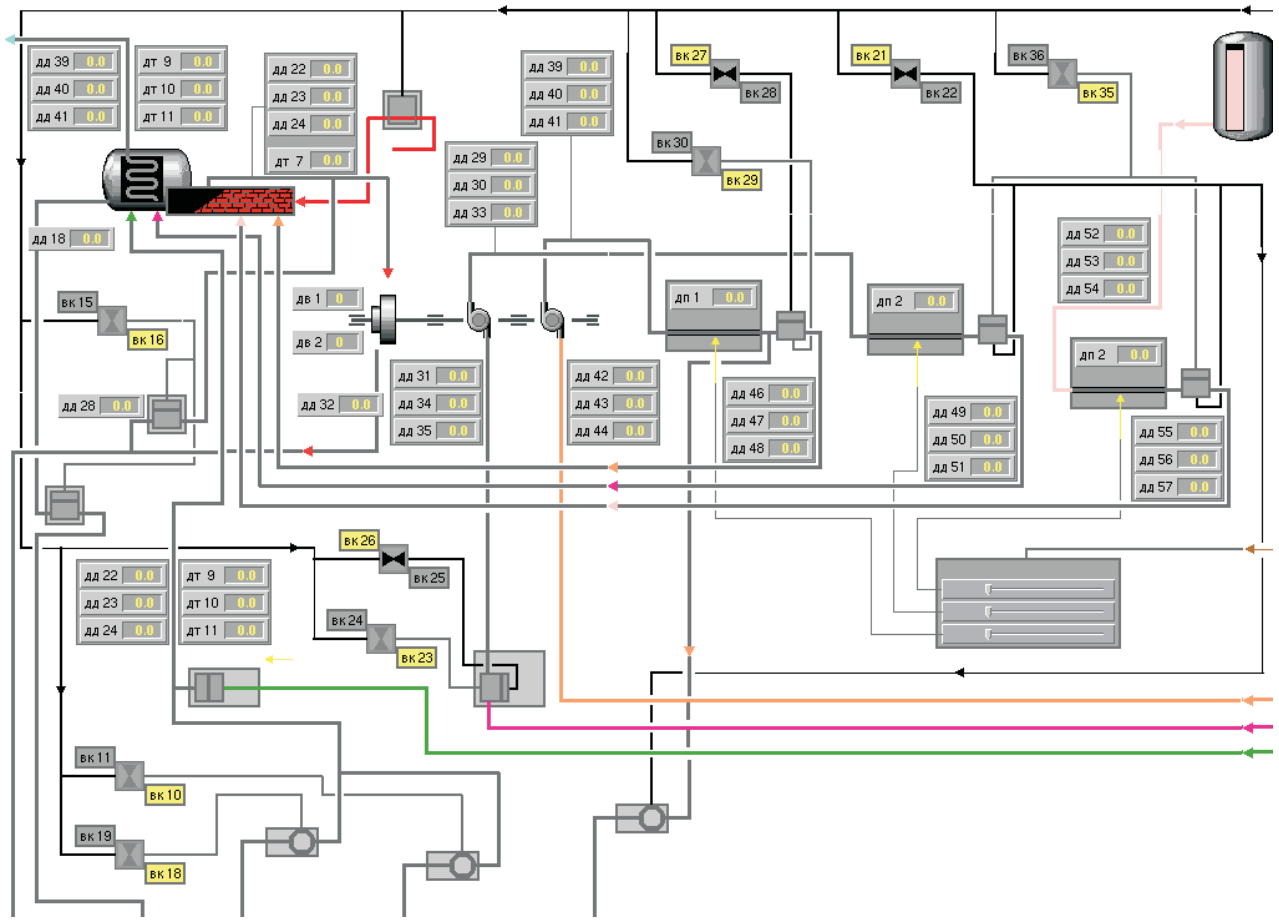

Рис. 1. Общий вид мнемосхемы технологичесского процесса

Как гласит один из принципов построения мнемосхем, важно отображать только существенные для оператора значения данного технологического процесса.

Рассмотрим для примера такой элемент объекта управления как трубопровод. Трубопровод может быть заполнен носителем или быть пуст. Носитель может просто протекать по трубопроводу, а может находиться в нем под давлением (давление, чаще всего, изменяется во времени). Время заполнения и опорожнения трубопровода зависит уже от нескольких параметров: вязкость носителя, коэффициент трения, длина трубопровода, давление. В предлагаемом методе мы абстрагируемся от физики протекания носителя по трубопроводу (в данном случае оператору не нужно знать параметров физического процесса течения жидкости) и учитываем только само время T, требуемое для процесса заполнения трубопровода от точки входа в трубопровод (точка выхода из предшествующего элемента) до точки выхода (точка входа в элемент объекта, в который подает носитель данный трубопровод). Время и способ перехода элемента трубопровода из состояния пустого в наполнено (и обратно) – единственно важные для отображения оператору (в данном конкретном примере).

В общем случае процесс в трубопроводе можно отнести к группе самых простых: один вход и один выход. В случае, когда трубопровод подводит носитель к двум и более элементам объекта, то выходов у этого процесса будет два и более. Сложнее случай, когда к трубопроводу подсоединен, например, датчик давления. Тогда этот выход не будет дискретным (есть или нет носитель). Выход будет сложного, аналогового, типа с учетом диапазона возможного изменения давления.

Для процесса протекания носителя в трубопроводе предлагается по изменению входа (есть/нет носитель) перевести графические элементы мнемосхемы, представляющие трубопровод, в соответствующее состояние (цвет носителя/цвет отсутствия носителя) постепенно, частями, за время T. По истечению времени T соответствующим образом изменить выход. Такой процесс будем называть процессом динамизации.

Предлагаемый метод заключается в следующем.

1. Для каждого элемента мнемосхемы разрабатывается процесс динамизации с определенными количествами входов и выходов, переводящий элемент в одно из определенных состояний (автомат состояния). В общем случае (но могут быть исключения) состояние определяется только входами элемента.

2. Для каждой из точек перехода между элементами определяется специальный *флаг изменения*, по которому определяется момент перехода элемента из одного состояния в другое.

3. Процесс динамизации элемента может происходить во временном масштабе, либо элемент может отображаться по переходу состояния для оператора мгновенно. Поэтому для функции динамизации элемента определяется способ его визуализации. В первом случае функция должна запускать определенный таймер, непосредственно отображающий данный элемент в текущем состоянии с временными задержками. Во втором случае визуализация элемента происходит в самой функции.

4. Производится разработка алгоритмов для функций динамизации (в нотации АТД или в нотации блок-схем).

5. Формируется исходный код программы. Определяются флаги изменения и состояния. Определяются функции динамизации. Основной цикл программы организуется как автомат переходов;

6. Производится верификация программного продукта. Доказательством работоспособности программы служит логическое соответствие исходного кода разработанной документации.

Следует заметить, что при создании логики мнемонической схемы любой сложности ключевым фактором корректности разработки является обязательное соблюдение этапов (описанных ниже) и точное составление документации, сопровождающей разработку (словесных описаний, программных спецификаций, таблиц, диаграмм состояний и переходов, алгоритмов в удобной для разработчика нотации).

Этапы разработки мнемосхемы с использованием предлагаемого метода.

Разработка мнемосхемы включает несколько этапов.

Первый этап.

Общая структура мнемосхемы декомпозируется на функциональные части. Каждая функциональная часть именуется по центральному элементу (блок динамизации виджетов). На этом этапе создаются словесные описания каждой функциональной части.

Второй этап.

По описаниям функциональных частей определяются точки разграничения блоков динамизации и определяются возможные состояния каждого из блоков.

Для каждого блока разрабатываются диаграммы состояния [1].

Третий этап.

По диаграммам состояний и по схемам разбиения функциональной части на блоки динамизации для каждого блока определяется *функция динамизации* (срабатывающая по *флагу изменения*), которая по значению *флага состояния* отображает виджеты данного блока в том или ином виде. *Флаг состояния* в программной реализации имеет стандартный тип и входы блока динамизации соответствуют конкретным битам (биты флагов состояний описываются в спецификациях).

Флаг изменения предназначен для отслеживания состояний переходов в каждой из точек перехода из блока в блок. Эти флаги имеют битовый тип (два значения: есть изменение или нет изменения) и для экономии ресурсов ЭВМ объединяются в один из стандартных типов данных языка программирования (восемь флагов в один байт).

Четвертый этап.

Вся информация для каждого из блоков сводится в специальные таблицы.

Пятый этап.

Как было указано, для каждого блока определена своя функция динамизации.

Функция динамизации может непосредственно визуализировать виджеты, входящие в состав данного блока, т.е. для оператора виджеты блока будут меняться практически моментально, а может запускать специальный таймер, который будет постепенно переводить виджеты блока из начального состояния в конечное. Например, носитель по трубопроводу не может распространяться моментально. Для последовательной имитации распространения носителя по трубопроводу от входа

блока к его выходу и служит запускаемый функцией динамизации таймер.

На этом этапе создаются алгоритмы функций динамизации в нотации АТД. Также можно применять и блок-схемы.

Шестой этап.

По разработанным алгоритмам формируется исходный код программы. Основная работа программы выполняется в цикле постоянно работающего таймера. Обработчик таймера проверяет все *флаги изменения* на появление установленных битов и вызывает соответствующие функции динамизации.

При этом проверка условий по битам флага состояния выполняется, начиная со старшего бита. Этим гарантируется, что обработка ошибочных и аварийных состояний будет обработана, в первую очередь, в текущем цикле таймера (непосредственно биты ошибок и аварийных состояний мы определяем в спецификации как самыми старшими).

В результате применения такого метода разработки мнемосхемы технологического процесса любое изменение в любой точке мнемосхемы будет автоматически распространятся на всю мнемосхему согласно принципам причинно-следственных связей. Для примера на рис. 2 выделена часть мнемосхемы (см. рис. 1), визуализирующая процесс прохождения условного носителя от точки T1 до точки T2.

При визуализации процесса прохождения условного носителя по трубопроводам, появления значения датчиков, процессов разгона или торможения двигателей насосов и т.п. возникает необходимость имитации транспортных задержек. Эти задачи выполняют функции динамизации и соответствующие программные таймеры.

Из рис. 2 видно, что для появления носителя в точке T2 обязательными условиями должны предшествовать следующие процессы:

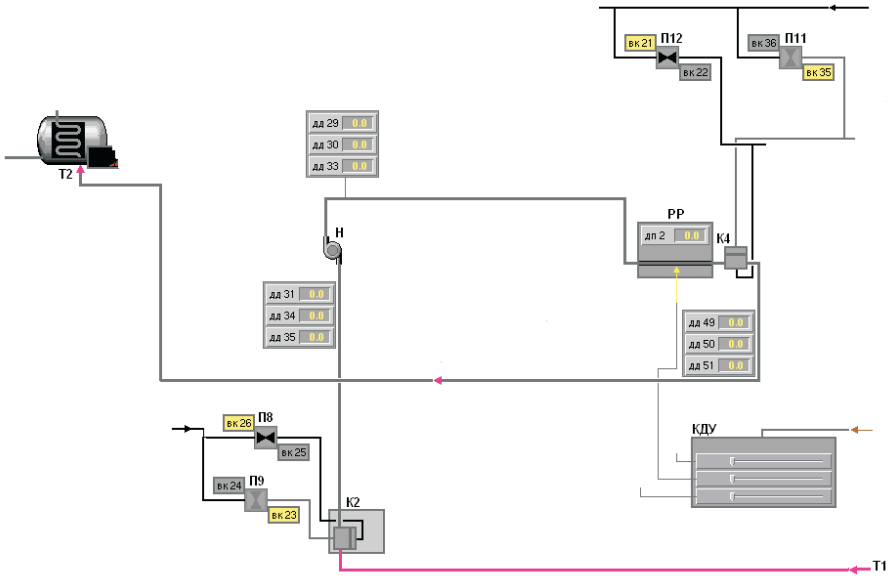

Рис. 2. Участок мнемосхемы технологического процесса

• присутствие носителя в точке Т1 и, как следствие, появление носителя на входе клапана К2;

• открытие клапана К2, т.е. срабатывание переключателя П8 (при обязательном выключении П9). При этом контролируются состояния переключателей по состоянию концевых выключателей: ВК24(-), ВК23(+), ВК26(+), ВК25(-). Носитель появляется на входе насоса Н, и давление отображается на датчиках ДД31, ДД34, ДД35;

• работа насоса Н и, как следствие, появление носителя на входе регулятора расхода РР (давление отображается на датчиках ДД29 - ДД30);

• дальнейшее прохождение носителя зависит от работы регулятора расхода РР (датчик положения ДП2), который, в свою очередь, управляется клапаном дискретного управления КДУ. Работоспособность КДУ зависит от наличия среды управления;

• после РР носитель поступает на вход клапана К4. Открытие клапана К4, т.е. срабатывание переключателя П11 (при обязательном выключении П12). При этом контролируются состояния переключателей по состоянию концевых выключателей: ВК21(-), ВК22(+), ВК36(+), ВК35(-). Носитель появляется в точке Т2, и давление отображается на датчиках ДД49 - ДД51.

Диаграмма состояний и переходов для клапана К1 показана на рис. 3.

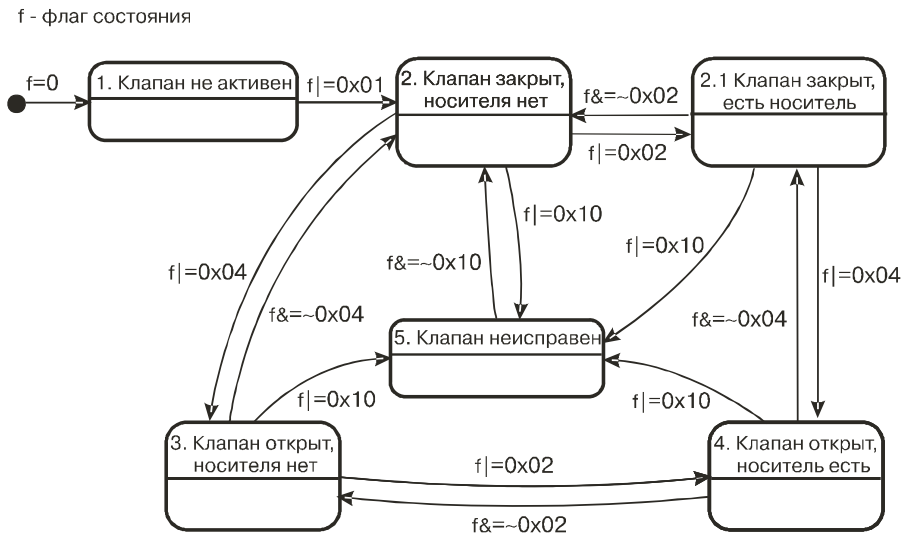

Рис. 3. Диаграмма состояний элемента К1

ЗАКЛЮЧЕНИЕ

SCADA-системы позволяют выполнять многие этапы разработки мнемосхем сравнительно быстрее, имеют богатые библиотеки графических символов, обладают универсальностью и гибкостью настроек, при этом разработчику не обязательны знания языков программирования. Однако SCADA-системы имеют и недостатки: они «тяжеловесны», существенно снижают быстродействие ОС, требуют больших размеров

оперативной памяти. Выбор SCADA-систем для разработки под ОС QNX достаточно ограничен (SCADA RealFlex, пакет SCADA Phocus/OPUS), стоимость самих пакетов и стоимость владения ими существенны, особенно если проект пилотный, разовый или в проекте не требуется вся мощь SCADA-системы. SCADA-системы могут быть уязвимы для хакерских атак.

Использование описанного в настоящей статье метода позволило с незначительными затратами запрограммировать логику состояний приведенной на рис. 1 мнемосхемы модуля и еще ряда подобных модулей, которые работают на нескольких компьютерах, объединенных в сеть Ehternet. Элементы мнемосхем отображаются синхронно, с заданными транспортными задержками, с учетом причинно-следственных связей реально протекающих технологических процессов.

СПИСОК ЛИТЕРАТУРЫ

1. *Поликарпова Н. И.*, *Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2009.
2. *Хопкрофт Д. Э., Мотвани Р.*, *Ульман Д. Д.* Введение в теорию автоматов, языков и вычислений, 2-е изд. Пер. с англ. М.: Издательский дом «Вильямс». 2002.
3. Динамизация в MasterSCADA. Обзор возможностей. – InSAT Company.