UPTEC IT 09 015
Магистерская диссертация 30 hp
Сентябрь 2009

Фреймворк для создания графических интерфейсов систем управления

Йонас Линдхольм

Перевод: https://oscat.ru/

Оглавление

Аннотация

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Промышленная техника становится всё более сложной, и в наши дни часто имеет встроенную систему управления. Эта система управления должна работать в тяжелых условиях эксплуатации. Также всё чаще для мониторинга и взаимодействия с такой системой используется дисплей. Для этого дисплея должен быть разработан графический пользовательский интерфейс (GUI). **Controller Development System** (**CoDeSys**) – это среда разработки систем управления. В ней программируется логика работы системы и создается ее операторский интерфейс.

В данный момент на рынке представлено две версии CoDeSys – CoDeSys V2 и CoDeSys V3. В моей дипломной работе проведен их обзор и сравнение. Обзор охватывает базовую функциональность, рабочее окружение, аспекты разработки и интеграции собственных графических элементов, а также тестирование производительности. Основной акцент сделан на двух последних вопросах. В результате сравнения сформулирован вывод, что обе версии CoDeSys поддерживают разработку собственных графических элементов, но CoDeSys V3 в этом плане шагнул существенно дальше предыдущей версии. Что касается производительности – проведенные тесты показывают, что при грамотном подходе к разработке графического интерфейса он не создаст серьезной нагрузки на систему управления.

Научный руководитель: Тобиас Андерссон
Рецензент темы: Бенгт Сандблад
Экзаменатор: Андерс Янссон
ISSN: 1401-5749, UPTEC IT09 015
Напечатано: Reprocentralen ITC

Предисловие

Я хотел бы поблагодарить своего руководителя в **CC Systems** Тобиаса Андерссона за все советы и поддержку, которую он оказал во время моей работы над дипломным проектом в CC Systems. Я также хотел бы в целом выразить признательность компании CC Systems за возможность работать здесь над дипломным проектом – это был отличный опыт.

Андреас Викенсё, коллега-дипломник, спасибо, что провёл мне экскурсию по офису в мой первый день.

1. Введение

В современной промышленной технике и транспортных средствах использование встраиваемых систем становится всё более обыденным делом. Эти системы выполняют широкий спектр задач: от сбора и обработки диагностической информации до управления движением техники и её рабочих органов (ковша экскаватора и т. д.).

Компания **Cross Country Systems AB** (далее **CC Systems**) разрабатывает и поставляет продвинутые системы управления для промышленной техники и транспорта, предназначенные для эксплуатации в тяжелых условиях. Такие системы делают технику более надежной, интеллектуальной и производительной. Примерами такой техники являются спредеры для контейнеров, лесозаготовительные машины, горнодобывающие машины и поезда. У компании есть офисы и заводы, расположенные в Швеции, Финляндии и Малайзии, в которых разрабатывается как аппаратное, так и программное обеспечение (http://www.ccsystems.com/).

Программное обеспечение системы управления часто разрабатывается с помощью платформы **Controller Development System** (далее **CoDeSys**), разработанной компанией **3S – Smart Software Solutions** (далее **3S**). Эта платформа включает в себя программный контроллер (**softPLC**), который обрабатывает сигналы от узлов приборов и выполняет логику управления техникой. В большинстве случаев для взаимодействия с системой управления используется дисплей. В таких ситуациях CC Systems или заказчик разрабатывают графический интерфейс пользователя (GUI) в редакторе визуализации CoDeSys (**CoDeSys Visu**). Визуализация имеет встроенный механизм обмена данными с системой исполнения CoDeSys, что делает разработку интерфейсов простой и интуитивно понятной за счет прямого доступа к переменным программы и широкому набору готовых графических элементов (CC Systems, 2008).

В настоящее время существует две версии CoDeSys: V2 и V3. Развитие V2 прекращено, но 3S продолжает осуществлять поддержку этой версии, в то время как все ресурсы отделов разработки переключены на версию V3. В продуктах CC Systems в данный момент используется CoDeSys V2; переход на V3 потребует времени, которое уйдет на адаптацию новой системы исполнения и перенос существующего ПО на новую платформу (Andersson, 2009a).

Ранее на кафедре уже было выполнено несколько дипломных проектов, посвященных CoDeSys. Но они рассматривали возможность подключения внешних графических интерфейсов к программному обеспечению, созданному в CoDeSys, и портированию системы исполнения CoDeSys на конкретную аппаратную платформу. Так что возможности визуализации CoDeSys ещё не были тщательно исследованы.

1.1. Цель

Целью данного дипломного проекта является изучение визуализации CoDeSys V2 и CoDeSys V3. CC Systems хочет исследовать возможности этих платформ и оценить, насколько они подходят для создания графических интерфейсов по принятым в компании правилам. Отдельно требуется изучить возможность создания собственных графических элементов для адаптации визуализации под требования конкретного клиента. Эти элементы должны иметь возможность нативной интеграции в CoDeSys, распространяться и устанавливаться на ПК клиентов в виде отдельных файлов, быть достаточно простыми в разработке и доработке. Также будут рассмотрены вопросы производительности обеих платформ.

1.2. Способ исследования

В процессе изучения визуализации CoDeSys использовались качественные методы оценки. Традиционно в технических исследованиях использовались количественные методы[1]. Популярность качественного метода оценки возросла в 90-е годы (Hoepfl, 1997). Было проведено достаточно исследований по применению этого подхода – например, его изучал Майкл Куинн Паттон (1990). Качественный метод применим для любого исследования, которое не базируется на какой-то статистике или других точных измерениях, характерных для количественного подхода.

Метод оценки, использованный в данном дипломном проекте, основан на методе Кронхольма и Голдкула (2003). Оценка проводится путем выбора целевых критериев для исследуемой системы и последующего определения, соответствует ли система этим критериям. Критерии формулируются на основе организационного контекста и позволяют сфокусироваться на системе как на инструменте выполнения конкретной задачи. Упомянутый выше метод комбинируется с методом количественной оценки; такой подход был предложен Хофлом (1997). В работе Хофла определены девять шагов для планирования исследования – они были использованы в рамках проведенной исследовательской работы.

Этот метод позволил мне выступить в роли эксперта, исследующего имеющуюся систему. У CC Systems есть четкое представление о нужной им программной платформе. Цели создания этой платформы, приведенные в п. 3, повлияли на выбор критериев для оценки системы.

Другая причина использования описанного выше подхода заключается в отсутствии в данный момент реальных пользовательских сценариев, которые я мог бы изучить. Целью дипломного проекта является не изучение того, насколько платформа удобна для конечных пользователей, а исследование возможностей платформы – в частности, возможности разработки собственных графических элементов. Для ответов на поставленные вопросы требуется эмпирическое исследование платформы, выходящее за рамки ее документации, а не наблюдение за поведением пользователей. Это позволяет мне как исследователю сосредоточиться на сборе нужной информации.

[1] Здесь и далее в этом абзаце перевод отступает от оригинального текста, в котором, видимо, возникла путаница в применении понятий «качественный» и «количественный» (примечание переводчика)

План исследования, как уже упоминалось выше, основан на девяти шагах, сформулированных Хофлом (1997). Информация, необходимая для оценки платформы, была получена путем изучения самой платформы и ее документации, а также в рамках личных бесед с сотрудниками CC Systems и переписке по электронной почте с сотрудниками 3S. Сотрудники обеих компаний имеют значительный опыт работы с CoDeSys, и при этом 3S является компанией-разработчиком платформы – поэтому их можно рассматривать как экспертов по этой системе. Вопросы, оставшиеся без ответа от этих сотрудников, были эмпирически исследованы в процессе работы с платформой.

Исследование было разделено на три этапа. На первом этапе были сформулированы цели исследования. Частично эти цели уже были заранее известны из спецификации требований от CC Systems, но требовалось разобраться в них и собрать недостающую информацию.

На втором этапе было проведено исследование визуализации CoDeSys V2. Для начала мне потребовалось познакомиться с этой платформой. В процессе этого я изучал документацию, графические интерфейсы, разработанные CC Systems, и API для создания пользовательских элементов. Существует крайне мало документации, посвященной этому API, поэтому мне пришлось исследовать его эмпирически, создав несколько собственных элементов. Эти элементы также использовались при тестировании производительности визуализации CoDeSys V2. После получения всей необходимой информации я провел оценку платформы.

Таким же образом была проведена и оценка визуализации CoDeSys V3. Отличие заключается в том, что в данном случае тестирование производительности не проводилось по техническим причинам (см. подробнее в п. 1.3). API для создания пользовательских элементов в CoDeSys V3 не документировано; более того, к началу работы над дипломным проектом это API вообще отсутствовало. Поэтому я попросил Ларса Лодина из Best Engineering Scandinavia продемонстрировать процесс разработки элементов во время нашего с ним интервью. В следующих пунктах вы увидите, что некоторые моменты описаны лишь для одной из платформ (CoDeSys V2 или CoDeSys V3); это позволяет оценить их отличия. В частности, в визуализации CoDeSys V3 есть новый функционал, который отсутствовал в CoDeSys V2, и он имеет важное значение для рассматриваемой темы.

На последнем этапе был проведен анализ полученной информации. Анализ представляет собой сравнение обеих платформ и выявление их существенных отличий. В процессе анализа рассмотрено соответствие платформ целям CC Systems, сформулированным в п. 3.

Описанный выше метод относительно неплохо подходит для данного дипломного проекта. Его существенным недостатком является зависимость результата исследования от исследователя, производящего оценку. Поскольку в данном случае экспертом выступал я сам, то, осознавая всю важность сохранения объективности, я целенаправленно сосредоточился на том, чтобы не позволить своим субъективным суждениям повлиять на результат исследования. Критическое отношение к источнику исследования – это один из способов предотвратить появление некорректных суждений; важность этого факта рассмотрена в п. 1.4. Еще один недостаток метода исследования – в нем не рассматривается пользовательский опыт и удобство использования платформы. Ранее я уже упоминал, что в задачи моего дипломного проекта не входит тестирование удобства использования, которое будет рассмотрено в п. 1.3. Что касается изучения пользовательского опыта – сотрудники CC Systems могут рассматриваться как пользователи

CoDeSys. Поскольку значительная часть информации в ходе исследования получена непосредственно от них – то можно считать, что косвенным образом был изучен и пользовательский опыт.

1.3. Границы исследования

В рамках дипломного проекта рассматривались только визуализации платформ CoDeSys V2 и CoDeSys V3. Теоретически в рамках исследования можно было изучить и другие графические фреймворки – например, Qt. Это позволило бы оценить отличия CoDeSys от других платформ. Но по ряду причин это не было сделано. Во-первых, это бы изменило исследовательский вектор дипломного проекта. Его фокус сместился бы с изучения отличий визуализации CoDeSys V2 от CoDeSys V3 (что являлось предметом интереса CC Systems) на обзор различных графических фреймворков и создания с их помощью GUI. Во-вторых, CoDeSys состоит из нескольких крупных компонентов, одним из которых является подсистема визуализации. Если бы для разработки GUI использовался бы отдельный фреймворк, то алгоритмы управления всё равно создавались бы в CoDeSys. Поэтому потребовалось бы создание программной прослойки между этими двумя средами. Разработка такой прослойки должна являться задачей отдельного дипломного проекта. Именно поэтому выбор сделан в пользу встроенной визуализации CoDeSys, а не отдельного графического фреймворка (Andersson, 2009a).

Как уже упоминалось ранее, в рамках дипломного проекта не рассматриваются вопросы удобства использования и пользовательские сценарии. Это моё осознанное решение – я предпочёл сосредоточиться на целях, сформулированных CC Systems (см. п. 3). Конечно, анализ удобства использования созданных с помощью редактора визуализации CoDeSys графических интерфейсов был бы крайне интересен. Но в дипломном проекте я рассматривал возможности и характеристики платформ; удобство использования не имело в данном случае существенного значения.

Тестирование производительности визуализации было проведено только для CoDeSys V2. В настоящее время CC Systems еще не использует CoDeSys V3, а портирование системы исполнения CoDeSys Control V3 на один из существующих контроллеров исключительно для нужд дипломного проекта было бы нецелесообразным.

1.4. Использованные источники

Документации по CoDeSys довольно мало. В основном, имеющаяся документация охватывает основной функционал среды; при этом по версии CoDeSys V2 доступно больше документации, чем по V3. Поэтому существенная часть требуемой информации была получена от сотрудников CC Systems и 3S. Эти сотрудники могут быть предвзяты в своих суждениях; особенно это касается компании 3S, которая нацелена на увеличение объема продаж своей платформы. Поэтому вся полученная таким образом информация была критически оценена и проверена на практике.

Из-за используемого качественного метода оценки достоверность отчета во многом зависит от меня как от эксперта-исследователя. Решение по поводу того, являюсь ли я достаточно квалифицированным для этой работы, должны принять сотрудники CC Systems, выступающие в роли моих работодателей. Поскольку они одобрили мою кандидатуру для проведения этого исследования, то могу предположить, что считают меня достаточно квалифицированным.

1.5. Правила именования

В прошлых пунктах использовалось обозначение «визуализация CoDeSys», чтобы показать, что визуализация является лишь одним из компонентов этой среды. Как именно устроен CoDeSys – будет описано в п. 2.2. Далее обозначение «CoDeSys» будет использоваться при любой отсылке к среде – неважно, идет ли речь о визуализации или о другом компоненте. Обозначения «CoDeSys V2» и «CoDeSys V3» будут использоваться для конкретизации версии среды.

2. Материалы исследования

Компания CC Systems разработала несколько продуктов, которые являются частью платформы автоматизации **CrossTalk**. Эта платформа состоит из различных устройств, выполняющих разные функции в рамках системы управления – например, обработку сигналов ввода-вывода, отображение информации и передачу данных по Bluetooth. Примерами таких устройств являются **CrossFire**, предназначенный для обработки сигналов ввода-вывода подсистемы управления гидравликой, и **CrossCode**, предназначенный для создания человеко-машинного интерфейса и обработки сигналов джойстиков и переключателей, управления светодиодами т. д. Связь между устройствами осуществляется по протоколу **CANopen**, который является стандартом для встраиваемых систем и упрощает интеграцию оборудования других производителей. Платформа CrossTalk использует CoDeSys в качестве одного из своих инструментов (CC Systems, 2007).


Примеры промышленного транспорта, системы управления которым разработаны с помощью CrossTalk

CC Systems работают с различными типами промышленной техники и системами управления. Каждая система должна быть адаптирована для конкретной машины; использование платформы CrossTalk сокращает срок разработки и стоимость системы. CrossTalk предназначена для создания систем управления с расширенным функционалом – диагностикой, предиктивной аналитикой, GPS-трекингом и т. д. (CC Systems, 2007).

2.1. The CCP XS

Cross Country Pilot XS (**CCP XS**) – это один из встраиваемых компьютеров CC Systems, который входит в состав платформы CrossTalk и используется в данном диплом проекте. Он представляет собой центральный компонент системы управления транспортным средством. CCP XS имеет малый вес и компактные размеры, что позволяет осуществить его монтаж в ограниченном пространстве, а также может функционировать в температурном диапазоне от -40 °C до +65 °C. Он работает под управлением операционной системы Windows Embedded Compact 5.0 (Windows CE) – специальной версии Windows, разработанной для компьютеров со скромными аппаратными характеристиками и встраиваемых систем (CC Systems, 2008).

Внешний вид CCP XS

CCP XS выпускается в трех модификациях: базовой, расширенной и максимальной. В своем дипломном проекте я использовал «максимальную» модификацию со следующими характеристиками:

Процессор	Intel XScale, IXP425, 533 MHz
Оперативная память	256 MB SDRAM
Flash-память	1 GB
Операционная система	Windows CE 5.0
Дисплей	10.4” Touch screen, SVGA 800x600

Спецификация CCP XS, использованного в дипломном проекте

Как указано в спецификации – CCP XS имеет 10.4” сенсорный резистивный экран. Это открывает множество возможностей для создания графического интерфейса пользователя.

2.2. CoDeSys (Controller Development System)

CoDeSys – это интегрированная среда разработки проектов для программируемых логических контроллеров (ПЛК). Она позволяет разработчику создать систему управления с помощью единственного инструмента. В состав среды входят редакторы программирования, конфигураторы настроек аппаратных средств контроллера и редактор визуализации. Проект CoDeSys включает в себя программные объекты (POU), типы данных, визуализации и «ресурсы» контроллера. Одним из ресурсов является конфигурация ПЛК, с помощью которой вы можете добавлять в проект различные устройства – такие, как CrossCode или CrossFire. После их добавления будут автоматически созданы глобальные переменные для взаимодействия с сигналами ввода-вывода этих устройств, которые можно будет использовать в POU (3S-Software, 2009).

Связь между ПЛК и другими устройствами системы реализована с помощью протокола CANopen. В состав CoDeSys уже входит драйвер этого протокола, поэтому разработчик может сосредоточиться на разработке приложения, не уделяя времени вопросам коммуникации. Этот факт (а также простота и удобство CoDeSys) позволяют сэкономить разработчику до месяца работы без необходимости писать свой стек CANopen (Wahlström, 2009).

Компании, разрабатывающие продукты с использованием CoDeSys, являются участниками CoDeSys Automation Alliance (CAA). Эти компании могут обмениваться своими программными наработками без необходимости поддержки новых драйверов и т. п., так как все они используют стандарт CANopen. Если продукты CC Systems не подходят для решения задачи заказчика, то CC Systems может купить более подходящее устройство у одного из участников CAA (Wahlström, 2009).

По сравнению с другими подобными платформами CoDeSys является относительно «свободным» с точки зрения доступных контроллеров. При использовании других сред разработки вы можете быть «привязаны» (vendor-lock) к устройствам производителя этой среды. Альтернативой использованию CoDeSys является разработка собственной платформы. Это позволило бы разработчику создать систему, полностью соответствующую его требованиям. Но нужно иметь в виду, что разработка таких платформ с нуля требует много времени (Wahlström, 2009).

2.2.1. МЭК 61131-3

Международная электротехническая комиссия (МЭК) – это интернациональная некоммерческая организация по стандартизации в области электрических, электронных и смежных технологий. Эта комиссия разработала стандарт МЭК 61131-3, которому соответствует CoDeSys. Стандарт определяет 4 языка программирования ПЛК. Два из них являются графическими – LD (Ladder Diagram; язык релейно-контактных схем) и FBD (Function Block Diagram; язык функциональных блоков). Оставшиеся два языка являются текстовыми – IL (Instruction List; похож на язык ассемблера) и ST (Structured Text; похож на Паскаль) (Wikipedia, 2009).

3. Требования к графической платформе

В этом пункте рассмотрены требования, которые CC Systems предъявляет к своей графической платформе. Они носят как технический, так и организационный характер. На эти требования я буду ориентироваться при исследовании платформ CoDeSys V2 и CoDeSys V3. Когда исследование будет завершено, при сравнении платформ его результаты будут сопоставлены с этими требованиями.

Основное требование, которое сформулировала CC Systems – это возможность создания «профиля визуализации», который должен включать в себя собственные графические элементы, интегрированные в CoDeSys. Также должна быть возможность распространения этого профиля – в частности, его установка на ПК клиентов CC Systems (Lans, 2009).

Разработка профиля визуализации должна быть достаточно простой. Простота, по определению Джона Маэды (http://lawsofsimplicity.com/), достигается за счет уменьшения числа сущностей, грамотного подхода к организации разработки и сокращения затрат времени. Это важно, потому что предполагаемыми пользователями профиля визуализации являются клиенты CC Systems. Зачастую они не являются инженерами-программистами и поэтому не всегда знакомы с концепциями разработки графического интерфейса. Кроме того, простота разработки является одним из маркетинговых преимуществ, и поэтому является приоритетной целью. Простота разработки графического интерфейса, удовлетворяющего потребностям клиента, является преимуществом платформы, которую CC Systems хочет создать (Lans, 2009; Wahlström, 2009).

Продукты CC Systems разрабатываются для различных систем и продаются различным компаниям. Это требует адаптации элементов профиля визуализации под требования конкретной системы (Wahlström, 2009). Возможность кастомизации элементов важна; хотя простота разработки и является основным критерием, но у пользователя должна быть возможность настроить графический элемент, чтобы он соответствовал концепции конкретного графического интерфейса. Рассмотрим это на примере элемента «Стрелочный индикатор». Простота его разработки может быть достигнута за счет сокращения его параметров и разделения их по категориям, а возможность кастомизации – с помощью предоставления выбора произвольного изображения в качестве фона элемента.

Последним требованием к платформе является влияние разработанных графических интерфейсов на использование ресурсов CCP XS. Производительность контроллера должна оставаться в разумных пределах. Это будет оцениваться путем измерения загрузки CPU. Не существует жестких требований по поводу того, какое значение загрузки CPU является приемлемым; требуется изучить влияние графического интерфейса на производительность контроллера и определить параметры, с которыми это связано (Andersson, 2009a).

В результате можно выделить следующие основные требования:

- возможность разработки собственных графических элементов со следующими свойствами:
  - интеграция в CoDeSys;
  - возможность распространения для установки на ПК клиентов;
  - простота разработки;
  - возможность адаптации.
- сохранение разумной производительности контроллера при использовании графического интерфейса.

Как вытекает из требований – принципиально важна возможность разработки собственных графических элементов. В данный момент CC Systems использует визуализацию CoDeSys V2 в качестве своей платформы визуализации. Они поняли, что этой платформы (или текущего подхода к ее использованию) недостаточно для их задач. Вот почему так важно, чтобы платформа предоставляла возможность для собственных разработок (Andersson, 2009a).

4. Исследование CoDeSys V2

CoDeSys представляет собой интегрированную среду разработки, в рамках которой создается как логика пользовательского приложения, так и визуализация. За счет использования единого инструмента вы можете бесшовно связать переменные приложения с элементами визуализации. В окне ассистента ввода в визуализации будут отображаться все переменные и константы вашего приложения. Соответственно, не потребуется создавать дополнительную программную прослойку между кодом и визуализацией, что зачастую является довольно трудоемким занятием (3S-Software, 2007 г.).

4.1. Рабочее окружение

Среда разработки включает в себя 4 основные панели:

- дерево проекта (Object organizer);
- панель инструментов (Toolbar);
- редактор визуализации (Visualization object);
- панель сообщений (Project output).

В дереве проекта в режиме разработки визуализации добавляются и отображаются все экраны визуализации текущего проекта. Проект может содержать несколько экранов, которые можно переименовывать. В редакторе визуализации отображается текущий открытый экран. Один из экранов должен быть назначен стартовым – он будет отображаться при запуске проекта. В панели инструментов расположены иконки графических элементов, которые можно перетащить (drag-n-drop) на холст редактора визуализации. Эти элементы будут описаны в п. 4.3 и п. 4.4. Сам экран визуализации также является элементом и может быть размещен в плоскости другого экрана, что позволяет создавать «многостраничные» графические интерфейсы. В панели сообщений отображаются ошибки и сообщения компиляции (3S-Software, 2008).

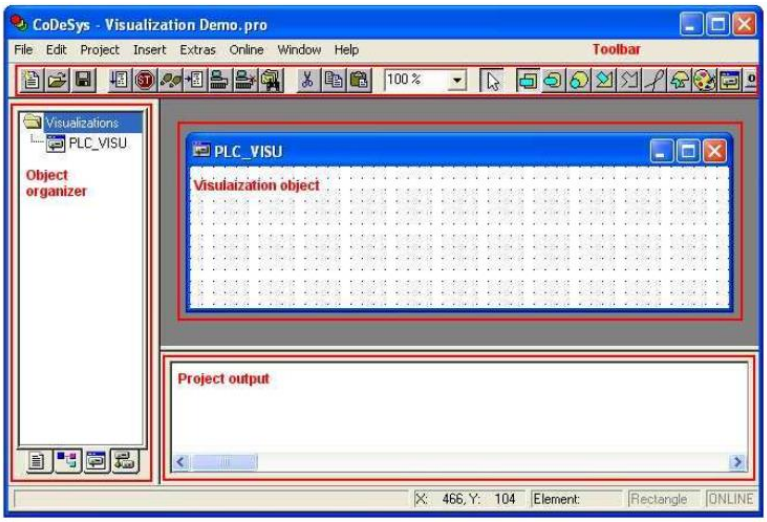

Рисунок 1 – Редактор визуализации CoDeSys V2

4.2. Экраны визуализации

Координаты элементов, расположенных на экране визуализации, можно настроить через список элементов, который представляет собой таблицу с номерами элементов, их типами и координатами. Номер представлен целым числом, тип элемента – строкой, а координаты – набором из 4 чисел, соответствующих координатам верхней левой и правой нижней точки элемента {x1, y1, x2, y2} (3S-Software, 2008).

Еще одной особенностью визуализации является возможность установки произвольного растрового изображения в качестве фона экрана и элемента. Это изображение будет размещено в самом нижнем слое экрана, и его перерисовка будет производиться только в той области, в которой произошли изменения внешнего вида графических элементов. (3S-Software, 2008). Это будет продемонстрировано в тесте производительности в п. 4.6.

4.3. Базовые графические элементы

Базовые графические элементы являются «кирпичиками», из которых строится визуализация. Все базовые графические элементы устроены по общему принципу. К таким элементам относятся:

Элемент	Описание
Прямоугольник	Объект в форме прямоугольника
Эллипс	Объект в форме эллипса
Скругленный прямоугольник	Объект в форме прямоугольника со скругленными углами
Полигон	Объект в форме многоугольника
Ломаная линия	Две или более точки, соединенные прямыми линиями
Кривая	Две или более точки, соединенные кривыми
Растровый рисунок	Импортированный графический файл формата .bmp, .tif или .jpg
Кнопка	Кнопка в стиле интерфейса Windows

Каждый элемент имеет диалог конфигурации, в котором настраивается функциональность элемента. Функционал можно разделить на две основные категории: внешний вид и обработка нажатий. Для всех базовых элементов доступны следующие настройки внешнего вида:

Настройка	Описание
Текст	Текст, отображаемый в элементе. Он может быть статическим (введенным вручную на этапе конфигурации) или динамически формироваться на основе значения переменной, привязанной к элементу. Если в состав текста входит спецификатор ANSI C (например, «%s»), то в режиме исполнения он будет заменен на значение переменной, привязанной к элементу, преобразованной к типу спецификатора
Переменные цвета	Цвета заливки и контура элемента. Могут быть заданы вручную или через переменные

Видимость/невидимость	Видимость элемента может изменяться с помощью переменной типа **BOOL**
Сдвиг	Здесь привязываются целочисленные переменные, с помощью изменения значений которых можно будет перемещать элемент по экрану визуализации
Угол	Здесь привязывается целочисленная переменная, с помощью изменения значения которой можно будет поворачивать элемент вокруг своей оси
Масштаб	Здесь привязывается целочисленная переменная, с помощью изменения значения которой можно будет изменять размер элемента

Также для всех базовых элементов доступны следующие настройки обработки нажатий:

Настройка	**Описание**
Переменная переключения/переменная нажатия	Здесь привязываются переменных типа **BOOL**, состояние которых будет изменено при нажатии на элемент. Переменная переключения инвертирует свое значение при каждом нажатии. Переменная нажатия принимает значение **TRUE** на время зажатия элемента и принимает значение **FALSE** после его отпускания
Ввод в переменную «Вывод текста»	В случае установки галочки при нажатии на элемент появляется поле ввода или экранная клавиатура (в зависимости от настройки элемента). Введенное значение будет записано в переменную, отображаемую элементом (т. е. привязанную к параметру **Переменные/Вывод текста)**
Переход на визуализацию	По нажатию на элемент можно осуществить переключение отображаемого экрана визуализации
Выполнение программы	Эта настройка позволяет определить специальные действия, выполняемые при нажатии на элемент – например, переключение языка визуализации, выполнение программы и т. д.

4.4. Продвинутые графические элементы

Продвинутые графические элементы являются более узкоспециализированными, чем базовые, и предназначены для выполнения конкретных задач. Окно настройки каждого такого элемента является уникальным. Я не буду подробно описывать продвинутые элементы, потому что они являются специфическими и используются довольно редко. Вместо этого я приведу краткий обзор некоторых из них:

Элемент	Описание
Гистограмма	Используется для отображения массива значений. Каждое значение представлено столбцом соответствующей высоты
Столбчатый указатель	Аналоговый индикатор с рамкой и шкалой. Доступны настройки цвета, размера и т. д.
Стрелочный индикатор	Представляет собой полуокружность со шкалой и стрелкой, указывающей на значение привязанной к элементу переменной. Доступны различные настройки внешнего вида и управления
ActiveX	Элемент из внешней .dll-библиотеки, разработанный по технологии ActiveX. Связь между библиотекой и системой исполнения является односторонней – можно передавать данные из библиотеки в систему исполнения, но нельзя передать данные из системы исполнения в библиотеку
Визуализация	Этот элемент позволяет отобразить один экран визуализации в плоскости другого

Большинство продвинутых элементов созданы на основе базовых элементов; каждый из них имеет отдельную иконку на панели инструментов и собственный диалог настройки. Это реализовано с помощью API создания собственных элементов визуализации, которое будет описано в п. 4.5 (Lodin, 2009a).

4.5. API для создания собственных графических элементов

Базовые графические элементы могут комбинироваться для создания продвинутых графических элементов – например, стрелочный индикатор может быть создан с помощью прямоугольников, полигонов и линий. Такие продвинутые элементы требуются достаточно часто, поэтому CoDeSys предоставляет API для их создания. Эти элементы создаются в виде DDL для Windows, а их интеграция с CoDeSys осуществляется через XML-файлы. Таким образом разработчик может создавать собственные элементы визуализации. Для элемента также можно задать название, иконку для панели инструментов, иконку указателя курсора (которая появляется при наведении курсора на элемент) и создать диалог настройки (Pfob, 2002).

4.5.1. Интеграция элементов с CoDeSys

Каждый графический элемент имеет XML-описание, которое используется для связи CoDeSys и DLL. При запуске среда CoDeSys считывает из DLL иконки доступных элементов и отображает их на панели инструментов. При добавлении или редактировании элемента в редакторе визуализации, CoDeSys отправляет XML в DLL. Затем DLL считывает из XML настройки текущего элемента (только в случае его редактирования). Далее DLL открывает диалог настройки. После изменения настроек и закрытия диалога DLL передает в CoDeSys актуальные значения настроек (Pfob, 2002).

4.5.2. Структура элементов

Как указывает Михаэль Пфоб (Pfob, 2002), каждый графический элемент представлен классом, который выполняет 3 задачи:

- извлечение данных из XML и их запись в XML;
- отрисовка и обработка диалога настройки элемента;
- отрисовка элемента и обработка нажатий.

Интерфейс класса элемента можно представить следующим образом:

```
class ExempleElem : public ExtElement
{
   public:
      // Конструктор
      CCMeterElem(void);
      // Деструктор
      ~CCMeterElem(void);
      // Парсер XML-структуры
      virtual bool ParseXML(XMLTagNode* pXMLRoot);
      // Запуск диалога настройки
      virtual bool Configure(HWND hParent);
      // Создание нового экземпляра элемента
      virtual bool CreateElement();
      // Создание XML для нового экземпляра
      virtual XMLTagNode* CreateXMLTree();
}
```

4.5.3. Доступные компоненты

Как уже упоминалось, продвинутые элементы создаются путем комбинирования базовых. Базовые элементы и их возможности описаны в п. 4.3. API для создания собственных элементов реализован для языка C++. Иерархия классов базовых элементов приведена ниже на рисунке 2. Базовый класс **VisualElem** содержит большую часть универсальных функциональных возможностей, общих для всех элементов. Список методов этого класса приведен в приложении IV.

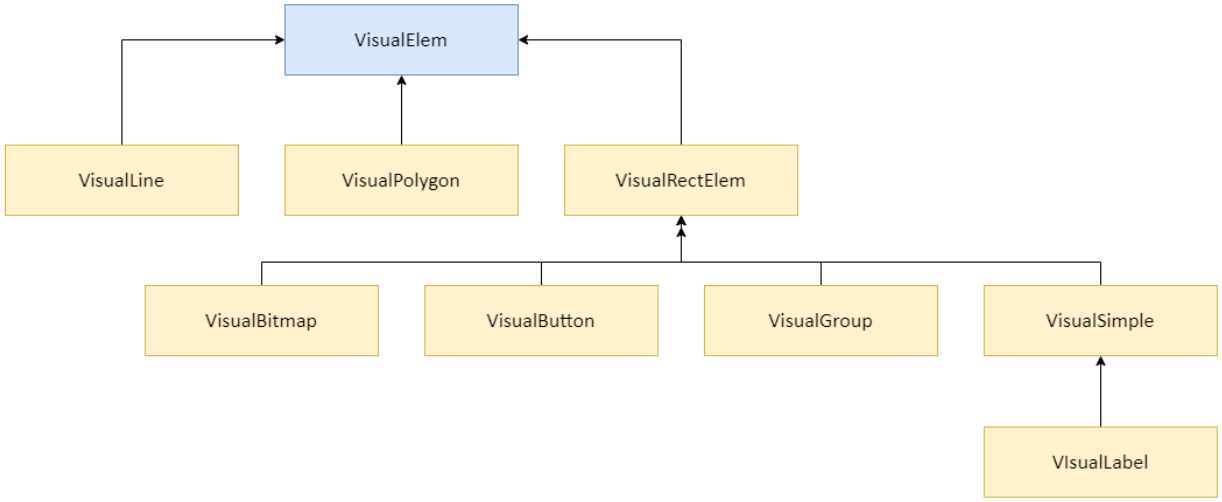

Рисунок 2 – Иерархия компонентов API, используемых при создании собственных элементов

В качестве примера элемента, который может быть создан с помощью API, можно привести стрелочный индикатор. Для его разработки можно использовать **VisualBitmap** в качестве фона, а поверх наложить несколько **VisualLabel** для создания шкалы.

4.6. Ограничения

API для создания собственных графических элементов имеет определенные ограничения. Во-первых, не поддерживается обработка событий. DLL в основном рассчитан на сборку новых элементов на базе существующих, и поведение этих новых элементов является статическим. Единственный способ взаимодействия с элементом для DLL – использование переменных нажатия и переключения. Но нет возможности по событию в элементе выполнить какой-то пользовательский код. Кроме того, не поддерживается прозрачность и градиент цвета. Если требуется эффект градиента, то единственная возможность его реализовать – расположить рядом друг с другом несколько линий разного цвета (Pfob, 2009). Поскольку API является расширением CoDeSys, то для разработки DLL необходим другой инструмент – например, Microsoft Visual Studio.

4.7. Производительность

Производительность визуализации оценивалась с помощью запуска проектов с различными графическими интерфейсами на CCP XS и измерении загрузки CPU. Для определения загрузки CPU использовался диспетчер задач операционной системы Windows CE. Тестовая программа включала в себя один POU, в котором происходил циклический инкремент значения переменной от 0 до 100, затем декремент от 100 до 0 и т. д. Значение этой переменной передавалось в визуализацию проекта. У разных тестовых проектов были разные визуализации.

Первое, что требовалось выяснить – влияние на загрузку CPU задач проекта. Задача, в которой выполнялся POU, называется PLC_TASK, а задача визуализации – VISU_TASK. В ходе эксперимента было проведено три серии измерений – с заданным интервалом вызова задачи PLC_TASK, равным 50 мс, 100 мс и 150 мс соответственно. В пределах каждой серии проводилось 9 испытаний – с заданным интервалом вызова задачи VISU_TASK в диапазоне от 50 мс до 150 мс (с шагом в 12.5 мс). Результаты эксперимента приведены на диаграмме 1.

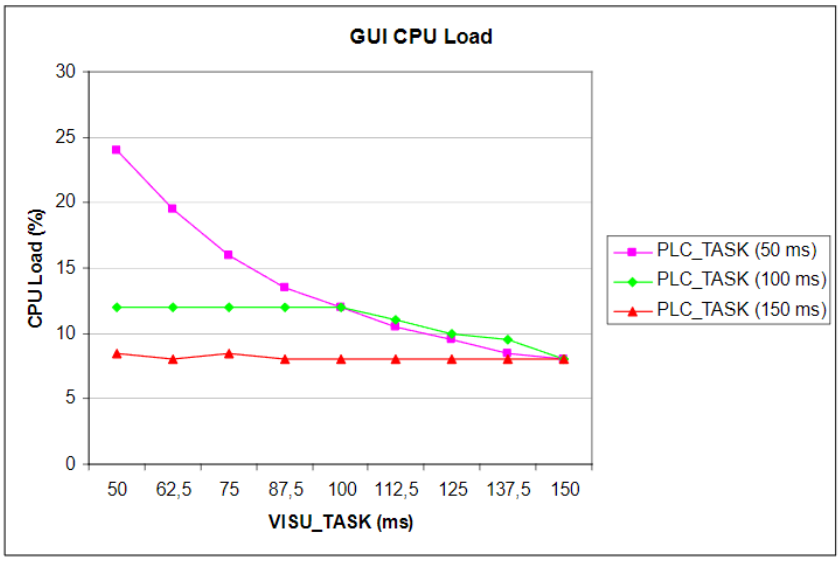

Диаграмма 1 – Влияние на загрузку CPU задач проекта

В ходе эксперимента было определено, что загрузка CPU зависит от задачи с самым большим интервалом вызова – неважно, является ли ей VISU_TASK или PLC_TASK. Из этого можно сделать вывод, что использование для задачи PLC_TASK интервала вызова, равного 50 мс, и изменение интервала вызова VISU_TASK в диапазоне от 50 мс до 150 мс даст достаточно точные измерения для серии тестов с различными типами элементов визуализации. Если использовать для PLC_TASK интервал вызова 100 мс или 150 мс, то загрузка CPU не будет корректно определена для случаев, когда интервал вызова VISU_TASK составит менее 100 мс или 150 мс соответственно.

Еще один важный вопрос – являются ли эти интервалы вызова задач адекватными с точки зрения реальной системы? Это зависит от конкретного приложения. Если бы графический интерфейс использовался бы исключительно для отображения часов или таймеров с дискретностью, равной 1 секунде, то устанавливать интервал вызова задачи VISU_TASK менее 1 секунды не имело бы смысла. И даже интервал в 1 секунду имел бы смысл только в том случае, если требуется видеть каждый «тик» часов. Если же в графическом интерфейсе отображаются переменные, значения которых следует обновлять существенно чаще – то можно задать интервал вызова VISU_TASK равным 20…50 мс (Moon, 2009).

4.7.1. Влияние различных типов элементов на производительность

После выбора интервалов задач можно приступать к эксперименту. Было проведено две серии экспериментов, в каждой из которых использовалось 3 проекта с различными типами графических элементов:

- изображение и полигон;
- полигон с фоновым изображением;
- мой собственный графический элемент, разработанный с помощью API.

Во всех случаях предпринимались попытки создать стрелочный индикатор. Изображение представляло собой фон этого индикатора. С помощью элемента Полигон в визуализации отрисована стрелка. Для нее настроен поворот в зависимости от значения привязанной переменной, обрабатываемой в задаче PLC_TASK.

Собственный элемент состоит из изображения, шкалы (созданной из 6 прямоугольников с текстовыми подписями без рамок), стрелки и прямоугольника, отображающего текущее значение переменной, привязанной к элементу.

4.7.2. Измерение времени цикла задачи визуализации

В первой серии экспериментов для задачи PLC_TASK был задан интервал вызова 50 мс, а для задачи VISU_TASK этот интервал менялся в диапазоне 50…150 мс (с шагом 12.5 мс). В каждом эксперименте использовался только один тип графического элемента. Результаты экспериментов приведены на диаграмме 2.

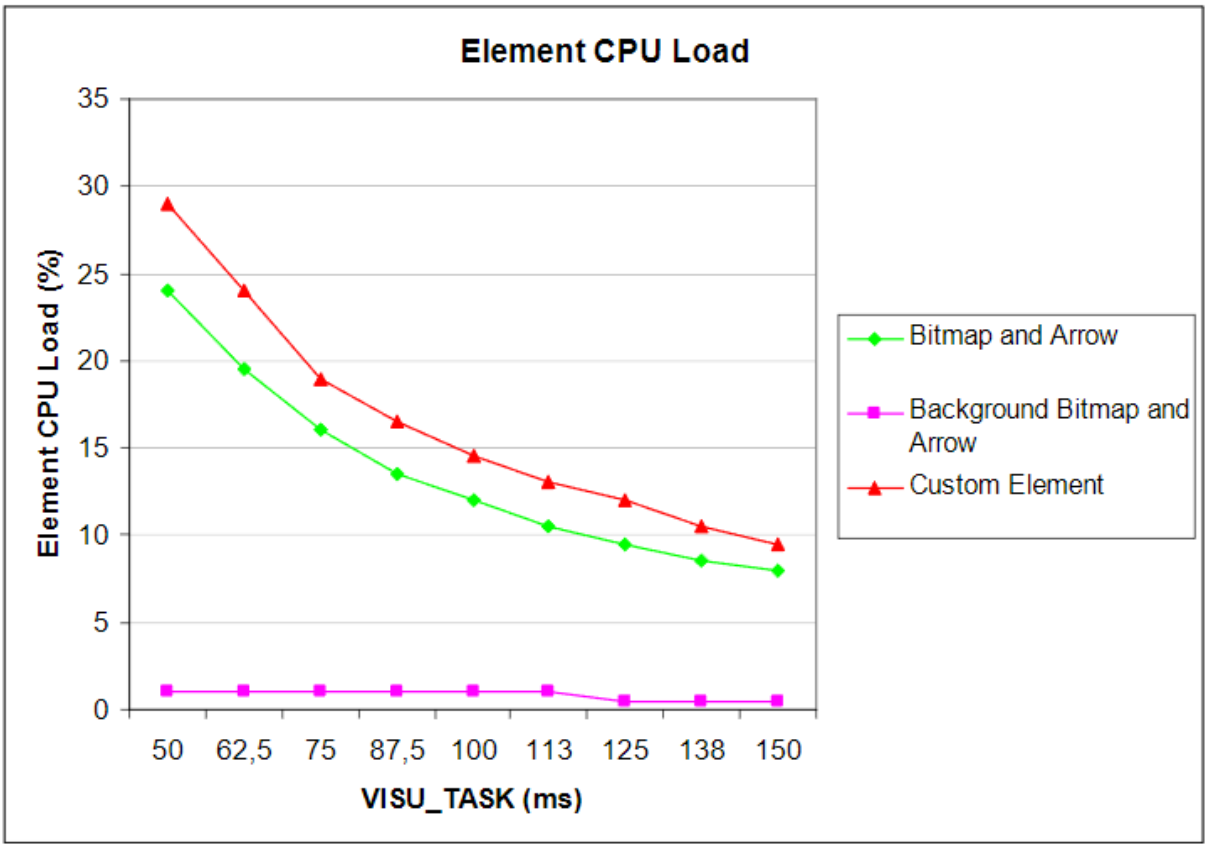

Диаграмма 2 – Влияние интервала вызова задачи VISU_TASK на загрузку CPU контроллера

4.7.3. Влияние количества элементов на производительность

В этой серии экспериментов интервал вызова задач PLC_TASK и VISU_TASK был фиксированным на протяжении всей серии. В каждом эксперименте менялось количество элементов, отображаемых на экране – от одного до шести (шесть элементов занимали весь экран CCP X). В каждом эксперименте использовался только один тип графического элемента. Результаты экспериментов приведены на диаграмме 3.

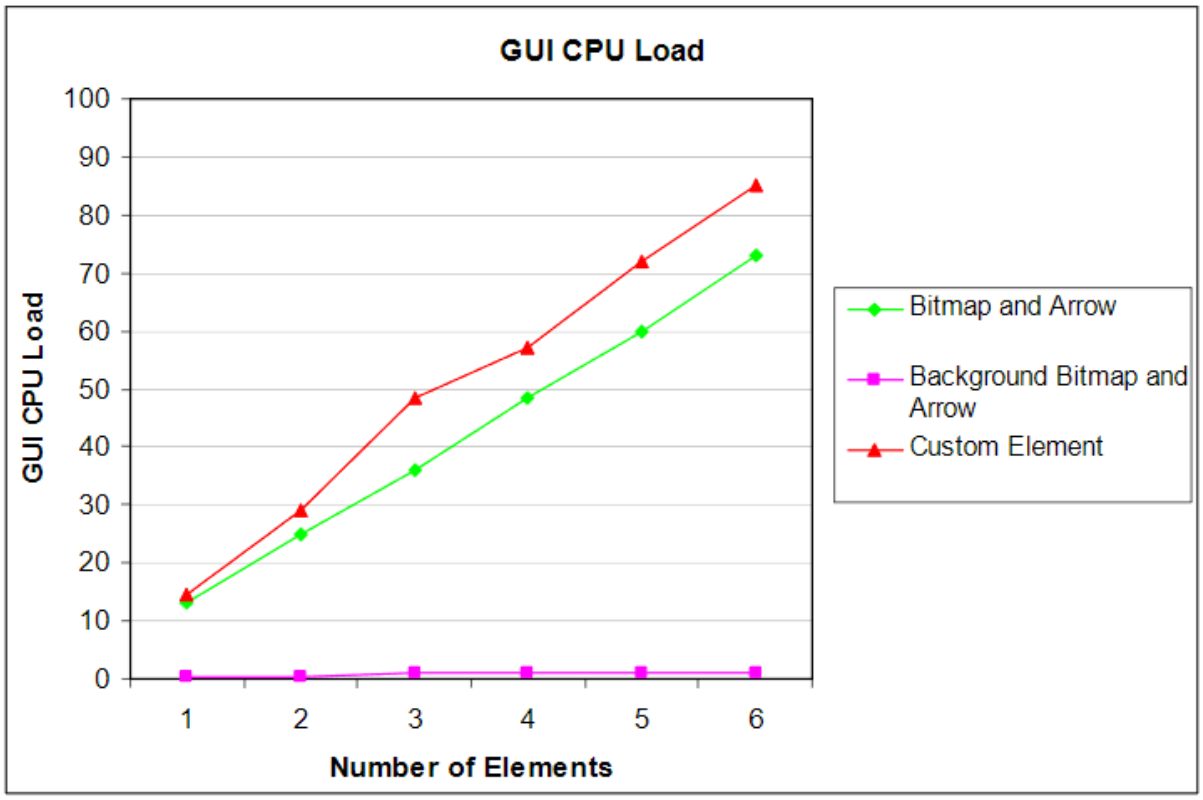

Диаграмма 3 – Влияние количества элементов визуализации на загрузку CPU контроллера

5. Исследование CoDeSys V3

CoDeSys V3 был разработан «с чистого листа». Это позволило изменить «фундамент» среды разработки (по сравнению с CoDeSys V2). В версии V3 появился объектно-ориентированный подход при разработке POU и библиотек. Удобство использования библиотек повышено, а в менеджер библиотек добавлено отображение встроенной документации. После завершения разработки можно легко создать ее скомпилированную версию для дальнейшего распространения и предоставления заказчикам. Еще одна новая функция – в визуализации происходит обновление только той части экрана, в которой произошли какие-то изменения (а не всего экрана, как в CoDeSys V2) – это сокращает время, затрачиваемое на обработку визуализации. Данная функция была анонсирована 3S, но не тестировалась в рамках данной дипломной работы (Lodin, 2009b).

5.1. Рабочее окружение

Среда разработки CoDeSys V3 имеет модульную архитектуру. Отдельные окна среды можно перемещать по экрану и закреплять произвольным образом. На рисунке 3 показан внешний вид среды при стандартных настройках. Окно POUs содержит общие ресурсы проекта – POU, визуализации и т. д., отображаемые в виде дерева. Проект может включать в себя один или несколько контроллеров (Devices); работа с ними производится в окне Devices, а их объекты также представлены в виде древовидной структуры. Редактор (Editor) используется для разработки программного кода или визуализации. Панель сообщений (не показана на рисунке) отображает сообщения и ошибки компиляции. Панель инструментов (Toolbox) содержит графические элементы, используемые при создании визуализации. Эта панель включает в себя отдельные папки для базовых (Basic), оконных (Windows) и продвинутых (Complex) элементов. Под панелью инструментов расположена панель свойств (Element Properties), используемая для редактирования настроек элементов – при этом можно настраивать несколько элементов одновременно. Все графические изображения, используемые в визуализации проекта, размещаются в отдельных объектах в дереве проекта – пулах изображений (Lodin, 2009b).

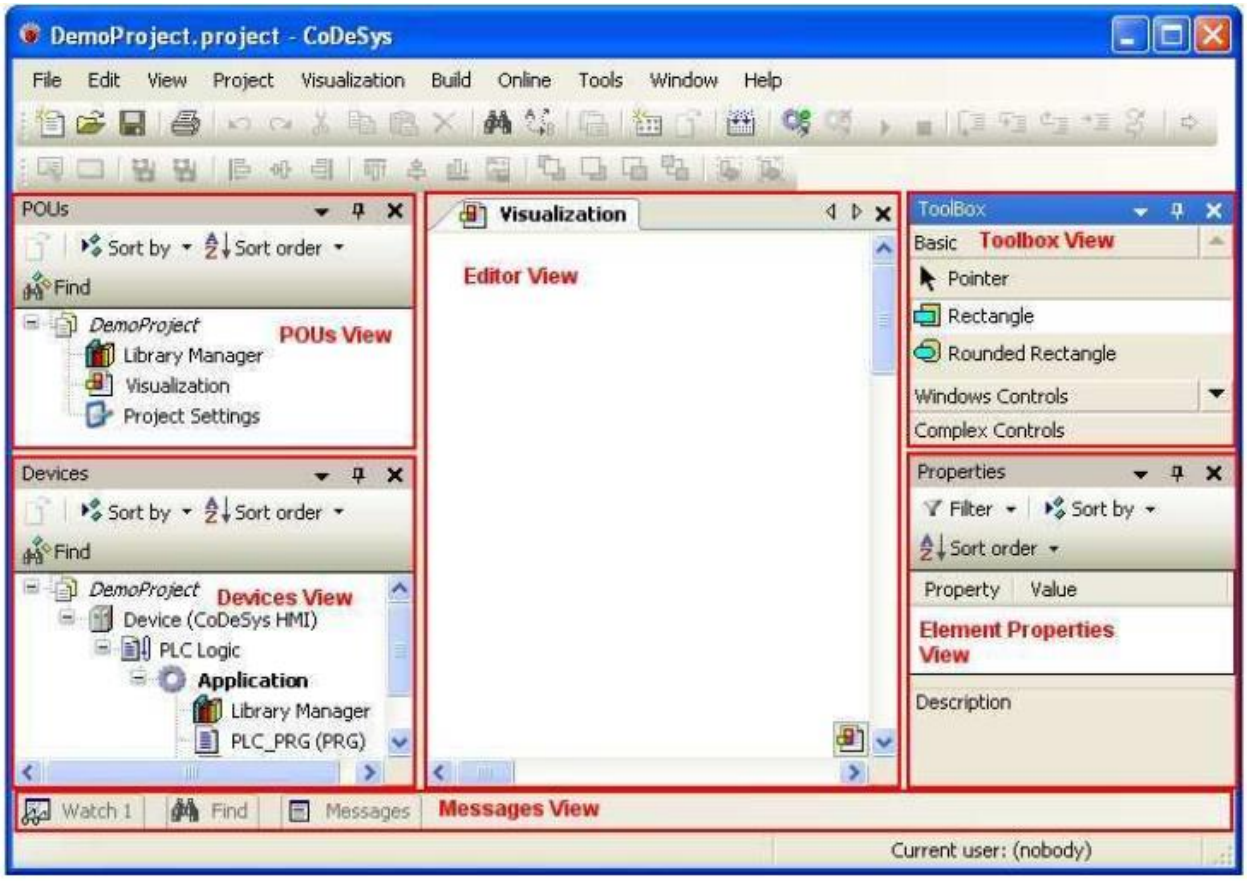

Рисунок 3 – Внешний вид среды CoDeSys V3

5.2. Графические элементы

Набор графических элементов в CoDeSys V3 в целом соответствует CoDeSys V2 с некоторыми улучшениями – например, теперь можно использовать в элементе **Изображение** графические файлы формата .wmf (Windows MetaFile). В панели инструментов графические элементы распределены по 3 папкам – базовые (Basic), оконные (Windows) и продвинутые (Complex). Дополнительные папки можно добавить с помощью API для создания собственных графических элементов (документация CoDeSys V3).

5.3. API для создания собственных графических элементов

Разработка собственных элементов в CoDeSys V3 производится непосредственно в самой среде – использование дополнительного ПО для этого не требуется. Графический элемент представляет собой функциональный блок на одном из языков стандарта МЭК 61131-3, который нужно создать в библиотеке (каждая библиотека может включать в себя один или несколько функциональных блоков элементов). Каждый из таких блоков должен реализовывать интерфейс IVisualElement, который содержит общие для всех элементов методы. Список методов этого интерфейса приведен в приложении V. После завершения разработки элемент следует установить через репозиторий визуальных элементов, который отображает список всех графических элементов, установленных на ПК, и их версии. Для созданного графического элемента можно настроить имя, иконку и папку на панели инструментов, в которой он будет размещен (Lodin, 2009b).

Собственные графические элементы, как и в CoDeSys V2, создаются с помощью комбинирования базовых элементов. Но в CoDeSys V3 появилась возможность обработать в коде блока элемента событие перерисовки элемента на экране визуализации, а также формировать собственные события. Например, для элемента **Ползунок** метод перерисовки будет вызываться при каждом перемещении «ручки». Также можно определить, какую именно часть элемента нужно перерисовать – это позволяет динамически менять внешний вид элемента в процессе работы приложения контроллера (Lodin, 2009b).

6. Сравнение и результаты

Давайте вернемся к требованиям к графической платформе, определенным в п. 3, и сопоставим их с информацией, полученной в результате исследования CoDeSys V2 и V3.

6.1. Возможность разработки собственных элементов

У этого требования есть 4 вложенных требования, но давайте сначала определимся, поддерживают ли вообще CoDeSys V2 и V3 разработку собственных графических элементов. Как указывают Pfob (2002) и Lodin (2009b) – обе эти среды позволяют создавать собственные элементы. Заметим, что собственные элементы создаются на основе базовых элементов, причем набор базовых элементов в CoDeSys V2 и V3 совпадает. В CoDeSys V2 разработка собственного элемента начинается с создания плана, по которому будут скомбинированы базовые элементы. Такой подход сильно ограничивает возможности разработчика – собственный элемент будет крайне статичным и единственный доступный способ взаимодействия с ним для пользователя – нажатие на элемент для изменения значений привязанной к нему логической переменной. В CoDeSys V3 элементы могут формировать события и изменять свой внешний вид в процессе работы контроллера. Это позволяет, например, создать элемент **Ползунок** и определить коллизию двух или более элементов при их перемещении (например, при создании анимации перемещения изделий по конвейеру и т. п.).

Разработка собственных элементов в CoDeSys V2 выполняется на C++, а в CoDeSys V3 – на языках стандарта МЭК 61131-3. Здесь нет плохого или хорошего варианта – программист, имеющий опыт работы с C++, но не использовавший языки стандарта МЭК, сочтет C++ API из CoDeSys V2 интуитивно понятным, и наоборот. Можно сказать, что Win32 API, используемый в CoDeSys V2 для создания диалогов настройки элементов, к настоящему моменту устарел и не соответствует современным стандартам GUI. В CoDeSys V3 диалог настройки формируется автоматически на основании доступных параметров элемента, заданных в его функциональном блоке.

6.1.1. Интеграция элементов с CoDeSys

Теперь рассмотрим вложенные требования к разработке собственных элементов. Первое из них – это возможность интеграции таких элементов в CoDeSys. Под «интеграцией» подразумевается механизм добавления собственных элементов в панель инструментов редактора визуализации. Такой механизм поддерживается как в CoDeSys V2, так и в CoDeSys V3. В CoDeSys V2 имя элемента и его иконка указывается в файле ресурсов (.dll). В CoDeSys V3 имя элемента, иконка и папка в панели инструментов задается в функциональном блоке элемента. Поэтому обе среды удовлетворяют требованию, но CoDeSys V3 в этом плане является несколько более удобным, так как позволяет выделить пользовательские графические элементы в одну или несколько папок в панели инструментов редактора визуализации.

6.1.2. Распространение и установка

Еще одним вложенным требованием является возможность распространения собственных графических элементов для установки на ПК клиентов. В CoDeSys V2 все элементы входят в состав внешней библиотеки (.dll), которая должна быть расположена в директории установки CoDeSys. Поэтому вместе с библиотекой потребуется распространять какой-то скрипт для ее установки. Андерссон указывает, что это можно реализовать в рамках уже существующего скрипта, который CC Systems использует для установки в CoDeSys своих компонентов (таргет-файлов, других библиотек и т. д.). В CoDeSys V3 удобство работы с библиотеками значительно повысилось. В менеджере библиотек теперь отображается встроенная документация со списком переменных каждого объекта и его описанием. Графические элементы в CoDeSys V3 создаются в виде функциональных блоков в рамках CoDeSys-библиотеки. При этом можно в одной библиотеке создать несколько элементов – или же создавать каждый элемент в собственной библиотеке. Для распространения собственных графических элементов в CoDeSys V3 требуется распространять эти библиотеки. Их можно предварительно скомпилировать, чтобы у пользователей не было доступа к исходным кодам. После установки библиотеки ее элементы появляются в репозитории визуальных элементов. Это было продемонстрировано в статье Lodin (2009b).

6.1.3. Простота разработки и возможность адаптации

Два последних вложенных требования относятся непосредственно к процессу разработки графических элементов. Простота разработки является маркетинговым преимуществом, которое позволяет сократить время и затраты на создание собственных элементов. Возможность адаптации необходима для удовлетворения требований клиентов, которым нужны специфические графические интерфейсы. И в CoDeSys V2, и в CoDeSys V3 работа с таким элементом не отличается от работы с базовым или продвинутым элементом. Обе среды обеспечивают достаточный уровень простоты разработки – встроенный редактор визуализации позволяет размещать элементы на экране с помощью мыши и не требует декларативного программирования в стиле html. Настройка выполняется с помощью диалоговых окон, в которых размещены все доступные параметры элементов. Такой подход соответствует законам простоты (https://lawsofsimplicity.com/).

Разработчик определяет набор параметров элемента (от которых зависит его функциональность) и способ их представления в диалоге настройки – таким образом, он формирует баланс между возможностями адаптации элемента к разным задачам и простоту его использования. Ненужные элементу параметры, полученные «в наследство» из стандартного диалога настройки, следует удалить.

6.2. Влияние на производительность

Результаты исследования влияния графических элементов на производительность контроллера, приведенные на диаграммах 2 и 3, наглядно демонстрируют, что при использовании фонового изображения элемента вместо отдельного элемента **Изображение** загрузка CPU значительно снижается. В двух других тестах загрузка CPU практически не отличается, хотя собственный элемент состоит из значительного количества базовых элементов. В первом тесте видно, что чем меньше интервал вызова задачи VISU_TASK, тем больше загрузка CPU – и для интервала 50 мс она становится неприемлемо высокой (≈25-30%). Как упоминалось в п. 4.7 – для величины загрузки CPU в требованиях к графической платформе не задано жесткого ограничения, но при обсуждении результатов тестирования с Тобиасом Андерссоном (2009c) было определено, что такие значения загрузки CPU слишком высоки. Еще отчетливее это видно в третьем тесте, в котором шесть собственных графических элементов обеспечивают 85% загрузки CPU. Одно из возможных решений проблемы – переход на CoDeSys V3, в котором экран визуализации обновляется не целиком, а перерисовываются только те элементы, в которых произошли изменения.

6.3. Инструменты для создания собственных элементов

При разработке собственных графических элементов в CoDeSys V2 требуется использовать Microsoft Visual Studio. Сам по себе этот инструмент не является плохим, но недостаток такого подхода заключается в том, что разработчику требуется использовать две среды разработки. В CoDeSys V3 разработка элементов происходит непосредственно в интерфейсе самой среды – это позволяет разработчику изучить только один инструмент, а компании сэкономить на лицензиях Microsoft Visual Studio. Эту возможность высоко оценил Тобиас Андерссон (2009a).

6.4. Рабочее окружение

И CoDeSys V2, и CoDeSys V3 имеют приблизительно одинаковую базовую функциональность – в обоих средах есть POU, дерево проекта, редакторы и т. д. Редакторы визуализации в обоих версиях выглядят схоже. Но CoDeSys V3 имеет принципиальное отличие от V2 с точки зрения внутреннего устройства – его модульная архитектура делает среду значительно более гибкой и настраиваемой для пользователя по сравнению с монолитной архитектурой CoDeSys V2. Важным улучшением CoDeSys V3 является принцип настройки элементов – теперь это происходит не в модальных диалогах настройки, как в CoDeSys V2, а в панели свойств. Это ускоряет процесс разработки, так как эта панель всегда отображается в редакторе визуализации, не является модальной и позволяет настраивать свойства нескольких элементов одновременно. Также в CoDeSys V3 появились пулы изображений, которые позволяют группировать графические файлы, используемые в визуализации проекта. Изображениям можно присвоить строковые идентификаторы, что упрощает их использование. CoDeSys V3 ощущается более цельным и гармоничным программным продуктом по сравнению с CoDeSys V2.

7. Заключение

На основании проведенных исследований я могу сделать вывод, что и CoDeSys V2 и CoDeSys V3 соответствуют требованиям к графической платформе, сформулированным в п. 3, за исключением требований к производительности. В случае CoDeSys V3 тестирование производительности не проводилось, так как соответствующая система исполнения не портировалась на CCP XS. В случае CoDeSys V2 тестирование было проведено, и его результаты показывают, что для обеспечения разумной производительности все графические изображения должны использоваться в качестве фоновых, а интервал вызова задачи VISU_TASK должен быть как можно меньше для обновления отображаемых значений в реальном времени.

Но интереснее не тот факт, что обе платформы соответствуют требованиям, а **как именно** они соответствуют. Обе платформы поддерживают разработку собственных графических элементов. Эти элементы можно интегрировать среду, распространять и устанавливать на другие ПК. Можно соблюсти баланс между простотой разработки элементов и удобством их использования. Это факты, но чем же концептуально отличаются рассмотренные платформы?

Так как среда CoDeSys V3 разрабатывалась с нуля – у компании 3S была возможность изменить ее фундаментальные аспекты (по сравнению с CoDeSys V2). Внедрение объектно-ориентированного подхода отразилось на создании POU и графических элементов. Теперь разработка элементов происходит непосредственно в самой среде и не требует использования дополнительных инструментов; визуализация CoDeSys V3 глубоко интегрирована в саму среду, а не является программным интерфейсом, использующим внешние графические библиотеки Windows. В CoDeSys V3 графические элементы, разработанные пользователем, не имеют отличий от встроенных элементов. Пользователь может взаимодействовать с элементом в процессе работы контроллера, используя его события.

Менеджер библиотек CoDeSys V3 позволяет просмотреть список библиотек проекта и ознакомиться со встроенной документацией на каждую из них. В CoDeSys V2 такой функционал отсутствовал, и в менеджере библиотек отображались только комментарии к POU и их переменным. Такой функционал в сочетании с появившимся репозиторием визуальных элементов и интеграцией процесса разработки элементов в саму среду существенно упрощает создание собственных графических элементов в CoDeSys V3.

CoDeSys V3 является более цельным продуктом по сравнению с CoDeSys V2. Появившиеся в нем пулы изображений позволяют группировать графические файлы, используемые в визуализации проекта, и упрощают контроль над ними. Создание библиотек стало более удобным. Разработка собственных графических элементов теперь происходит в самой среде путем создания библиотеки функциональных блоков, реализующих нужные системные интерфейсы. Созданные таким образом элементы могут распространяются в виде скомпилированных библиотек.

Необходимо отметить, что в моем исследовании не комментируется отсутствие в CoDeSys какого-либо функционала. Вполне возможно, что есть другие графические платформы, которые в большей степени соответствуют требованиям CC Systems. Но в рамках своей дипломной работы я сфокусировался на сравнении CoDeSys V2 и CoDeSys V3, поэтому не рассматривал другие платформы.

8. Будущие направления исследований

Есть несколько задач, которые я не смог выполнить по техническим причинам, а также из-за недостатка опыта и отсутствия необходимой информации. Во-первых, это тестирование производительности CoDeSys V3, которое не было проведено, так как соответствующая система исполнения не портировалась на CCP XS. Это тестирование можно было бы провести на виртуальном контроллере, запущенном на ПК, но такое исследование не позволило бы получить объективного представления о производительности платформы.

Еще один нерассмотренный вопрос – степень зрелости платформы CoDeSys V3. Сложно будет что-то сказать по этому поводу до того момента, пока система исполнения не будет портирована на контроллеры CC Systems. Определенную информацию о состоянии платформы можно получить, изучив отчеты об ошибках на сайте 3S Software (http://www.3s-software.com). В период с 1 января до 5 июня 2009 года было зарегистрировано около 900 ошибок – и приблизительно столько же было зарегистрировано за аналогичный период 2008 года.

Как было отмечено в п. 1.4 – документация по CoDeSys V3 является крайне скудной, особенно в плане API разработки собственных графических элементов. Но наличие такой документации является необходимым условием для возможности создания таких элементов. Требуется не только описание, как устроен функциональный блок графического элемента, но и документация по используемым им структурам, обработке событий и т. д.

Кроме того, компания CC Systems может быть заинтересована в исследовании подходов к разработке графических интерфейсов для промышленной техники и транспорта. В данном дипломном проекте вообще не рассматривались вопросы эргономики пользовательского интерфейса. В интервью с сотрудниками CC Systems мы обсудили некоторые интересные требования – например, высококонтрастные дисплеи для работы в дневное время. В области эргономики человеко-машинных интерфейсов систем управления проведено достаточно много исследований. Хорошими примерами являются книги «Automation and Human Performance» (Endsley, M.R., 1996) или «Usability Design: A Framework for Designing Usable Interactive Systems in Practice» Йоранссона (Göransson, 2001). CCP XS — это очень продвинутый панельный контроллер; как можно использовать его сенсорный экран с еще большим эффектом? В предыдущих разделах я уже неоднократно упоминал, что графический интерфейс является одним из маркетинговых преимуществ продукта. Соответствует ли маркетинговая привлекательность графического интерфейса реальным потребностям пользователей? Эти вопросы, которые не были сформулированы и рассмотрены в данной дипломной работе, но CC Systems могла бы получить пользу от их изучения.

9. Список литературы

9.1. Библиография

Cronholm, Stefan & Goldkuhl, Göran (2003), Strategies for Information Systems Evaluation – Six Generic Types, *Electronic Journal of Information Systems Evaluation*, vol. 6: 2, ss. 65-74. Also available at http://www.ejise.com/volume6-issue2/issue2-art8-cronholm.pdf

Endsley, M.R.: Automation and situation awareness. In R. Parasuraman & M. Mouloua (Eds), *Automation and Human performance: Theory and applications* (Oxford: Taylor & Francis, Inc, 1996)

Göransson, Bengt: *Usability Design: A Framework for Designing Usable Interactive Systems In Practice* (Uppsala: Department of Informations Technology, Uppsala Universistet, 2001)

Hoepfl, Marie C (1997), Choosing Qualitative Reasearch: A Primer for Technology Education Researchers, *Journal of Technological Education*, vol 9: 1. Also available at http://scholar.lib.vt.edu/ejournals/JTE/v9n1/hoepfl.html

Patton, M. Q: Qualitative Evaluation and Research Methods (2nd ed.). (Newbury Park, CA: Sage Publications, Inc, 1990)

9.2. Другие печатные издания

3S-Software, *User Manual for PLC Programming with CoDeSys 2.3*, user manual (Uppsala: 3S-Software, 2007)

3S-Software, *CoDeSys Visualization*, user manual (Uppsala: 3S-Software, 2008)

CC Systems, *Beskrivning av examanesarbete – Graphical interface framework for control system environments*, document (Uppsala: CC Systems, 2009)

CC Systems, *CrossTalk*, information leaflet (Uppsala: CC Systems, 2007). Also available at http://www.cc-systems.com/Portals/0/PDF/Products/Leaflet/CrossTalk2007.pdf

CC Systems, *CC Pilot XS*, information leaflet (Uppsala: CC Systems, 2008). Also available at http://www.ccsystems.com/Portals/0/PDF/Products/Leaflet/CCPilotXSAll-Integrated-Adress.pdf

Pfob, Michael, *Plugin Interface for the CoDeSys Visualisation*, user manual (Uppsala: CC Systems, 2002)

9.3. Электронные ресурсы

CC Systems website http://www.cc-systems.com, retrieved 2009-05-06

3S-Softwares website http://www.3s-software.com, retrieved 2009-05-06

The laws of simplicity http://lawsofsimplicity.com/, retrieved 2009-05-06

Qt-software http://www.qtsoftware.com/, retrieved 2009-06-05

CoDeSys V3, Online Help Documentation, retrieved 2009-05-05

Wikipedia 2009, article about IEC 61131-3 http://en.wikipedia.org/wiki/IEC_61131-1, retrieved 2009-02-04

9.4. Устное общение и переписка по электронной почте

Andersson, Tobias (a), *Software developer at CC Systems*, Uppsala 2009-02-27

Andersson, Tobias (b), *Software developer at CC Systems*, Uppsala 2009-03-05

Bäckström, Johan, *Software developer at CC Systems*, Uppsala 2009-03-04

Gustafsson, Lars, *Software developer at CC Systems*, Uppsala 2009-03-04

Lans, Fredrik, *Software development manager at CC Systems*, Uppsala 2009-02-27

Lodin, Lars (a), *CoDeSys Expert at Best Engineering Scandinavia*, Järvsö 2009-02-11

Lodin, Lars (b), *CoDeSys Expert at Best Engineering Scandinavia*, Järvsö 2009-05-05 Moon,

Carl-Magnus, *Software developer at CC Systems,* Uppsala 2009-04-05

Pfob, Michael, *Software developer at 3S-Software*, Kempten 2009-03-02

Wahlström, Fredrik, *Software developer at CC Systems*, Uppsala 2009-03-09

Wendebaum, Jochen, *Software developer at CC Systems*, Västerås 2009-04-07

10. Приложение

10.1. Приложение I: Глоссарий

API (Application Programming Interface; интерфейс прикладного программирования) – набор структур данных и классов, используемых для взаимодействия между приложениями.

CoDeSys (Controller Development System) – среда разработки приложений для ПЛК, разработанная компанией 3S-Software.

CoDeSys Visu (Controller Development System Visualization) – редактор визуализации среды CoDeSys, используемый для создания графического интерфейса пользователя.

GUI (Graphical User Interface; графический интерфейс пользователя) – набор средств для взаимодействия между пользователем и приложением.

ПЛК (Программируемый логический контроллер) – промышленный компьютер, используемый для автоматизации производственных процессов.

POU (Program Organization Unit; программный объект) – файлы исходного кода, используемые в CoDeSys V2 и CoDeSys V3.

МЭК (Международная электротехническая комиссия) – международная некоммерческая организация по стандартизации в области электрических, электронных и смежных технологий.

МЭК 61131-3 – раздел международного стандарта МЭК 61131 (разработанного Международной электротехнической комиссией), описывающий языки программирования для программируемых логических контроллеров.

Визуализация – синоним термина GUI, часто используемый в документации CoDeSys.

10.2. Приложение II: API для создания собственных графических элементов в CoDeSys V2

Интерфейсный файл библиотеки графических элементов (DLL) используется средой CoDeSys для получения информации об элементах. Он должен иметь расширение .ete и включать в себя следующие функции:

Функция	Описание
int GetElementsCount ();	Возвращает число пользовательских элементов
const char* GetElementName (int iElemNr);	Возвращает имя элемента с номером iElemNr
WORD GetElementIcon(int iElemNr);	Возвращает ID иконки элемента с номером iElemNr
XMLTagNode* CreateElement(int iElemNr, XMLTagNode* pXMLRoot, HWND hParent, void (*SubClassEdit)(HWND));	Формирует диалог добавления и настройки пользовательского элемента с номером iElemNr из переданного XML-дерева pXMLRoot. hParent — это дескриптор окна, в котором находится диалог. SubClassEdit — это указатель на функцию, которая разрешает доступ к уже объявленным переменным через интеллектуальный ввод (Intellisense). Функция возвращает XML-дерево пользовательского элемента. В случае нажатия в диалоге кнопки «Отмена» функция возвращает 0
void DeleteXMLTree (XMLTagNode* pXMLRoot);	«Освобождает» XML-дерево, созданное функцией CreateElement. Вызов этой функции необходим после закрытия диалога. В противном случае могут возникнуть проблемы, если в исполняемом файле освобождается память, которая была зарезервирована в DLL

10.3. Приложение III: Идентификаторы пользовательских элементов

Диапазон идентификаторов пользовательских графических элементов: 46040…46065 – то есть можно создать до 25 элементов. Внутри DLL элементы нумеруются с нуля.

10.3.1. Resource.h

```
#define IDM_INSERTTESTELEM 46040
#define IDS_NAME_TESTELEM  46140
#define IDI_TESTELEM       46040
```

10.3.2. globals.h

```
#define ELEMCOUNT   1
#define TESTELEMID  0
```

10.3.3. ElementDLL.cpp

```
ELEMENTDLL_API int GetElementsCount(void)
{
   return ELEMCOUNT;
}

ELEMENTDLL_API WORD GetElementId (int iNr)
{
   switch (iNr)
   {
      case TESTELEMID:
      {
         return IDM_INSERTTESTELEM;
      }
      /* + 100 означает, что будет запрошен ID имени элемента */
      case TESTELEMID + 100:
      {
         return IDS_NAME_TESTELEM;
      }
     /* + 200 означает, что будет запрошен ID пиктограммы курсора элемента */
      case TESTELEMID + 200:
      {
         return IDC_CUR_TESTELEM;
      }
      /* + 300 означает, что будет запрошен ID иконки элемента */
      case TESTELEMID + 300:
      {
         return IDI_TESTELEM;
      }
      default:
      {
         return 0;
      }
   }
}
```

10.4. Приложение IV: Список методов класса VisualElem (CoDeSys V2)

Публичные методы

virtual XMLTagNode * **GetXMLNode** ()
virtual void **SetFrameColor** (COLORREF col)
virtual void **SetInsideColor** (COLORREF col)
virtual void **SetFrameColorAlarm** (COLORREF col)
virtual void **SetInsideColorArarm** (COLORREF col)
virtual void **SetHasFrameColor** (bool val)
virtual void **SetHasInsideColor** (bool val)
virtual void **SetLineWidth** (int width)
virtual void **SetExprXOffset** (string expr
virtual void **SetExprYOffset** (string expr)
virtual void **SetExprScale** (string expr)
virtual void **SetExprAngle** (string expr)
virtual void **SetExprInvisible** (string expr)
virtual void **SetExprAlarm** (string expr)
virtual void **SetCenter** (Point center)
virtual void **SetToggleVariable** (string var)
virtual void **SetZoomVisu** (string var)
virtual void **SetExec** (string var)
virtual void **SetTapVariable** (string var)
virtual void **SetCalculatedCenter** ()
virtual COLORREF **GetFrameColor** ()
virtual COLORREF **GetInsideColor** ()
virtual COLORREF **GetFrameColorAlarm** ()
virtual COLORREF **GetInsideColorArarm** ()
virtual int **GetLineWidth** ()
virtual string **GetExprXOffset** ()
virtual string **GetExprYOffset** ()
virtual string **GetExprScale** ()
virtual string **GetExprAngle** ()
virtual string **GetExprInvisible** ()
virtual string **GetExprAlarm** ()
virtual Point **GetCenter** ()
virtual string **GetToggleVariable** ()
virtual string **GetZoomVisu** ()
virtual string **GetExec** ()
virtual string **GetTapVariable** ()
virtual bool **HasFrameColor** ()
virtual bool **HasInsideColor** ()
virtual int **GetFontSize** ()
virtual string **GetHorzAlignment** ()
virtual string **GetVertAlignment** ()
virtual void **SetItalic** (BOOL val)
virtual void **SetStrikedOut** (BOOL val)
virtual void **SetUnderlined** (BOOL val)
virtual void **SetFontWeight** (int weight)
virtual void **SetCharSet** (unsigned char charset)

virtual void **SetFontColor** (COLORREF col)
virtual void **SetAlignHorz** (HorzAlignment horz)
virtual void **SetAlignVert** (VertAlignment vert)
virtual void **SetFontSize** (int size)
virtual void **SetLabelText** (string text)
virtual void **SetFontName** (string name)
virtual void **SetTextVariable** (string var)
virtual void **SetFont** (const CHOOSEFONT &chFont)
virtual bool **CalculateTextSize** (SIZE *sz)
virtual void **PaintElement** (HWND hwnd, HDC hdc, RECT rect, int iSizeX, int iSizeY)

Защищенные (protected) методы

VisualElem (const VisualElem &VElem)
virtual void **PaintText** (HWND hwnd, HDC hdc, RECT rect, Point OutTopLeft, Point OutBottomRight, int iSizeX, int iSizeY)
XMLTagNode * **GetExprNode** (const string &strMainTag, const string &strVar, bool isVar=true)

Защищенные (protected) атрибуты

bool **m_bHasFrameColor**
bool **m_bHasInsideColor**
COLORREF **m_FrameColor**
COLORREF **m_InsideColor**
COLORREF **m_FrameColorAlarm**
COLORREF **m_InsideColorAlarm**
int **m_iLineWidth**
string **m_strExprXOffset**
string **m_strExprYOffset**
string **m_strExprScale**
string **m_strExprAngle**
string **m_strExprInvisible**
string **m_strExprAlarm**
string **m_strToggleVariable**
string **m_strZoomVisu**
string **m_strExec**
string **m_strTapVariable**
Point **m_CenterPoint**
HorzAlignment **m_iAlignHorz**
VertAlignment **m_iAlignVert**
CHOOSEFONT **m_ChooseFont**
LOGFONT **m_LogFont**
string **m_strText**
string **m_strTextVariable**

10.5. Приложение V: Список методов интерфейса IVisualElement (CoDeSys V3)

BOOL **ContainPoint** (VisuStructPoint pt)
BOOL **Destruct** ()
INT **ElementInfo** (POINTER TO Visu_StructElementInfo pData)
POINTER TO VisuStructClientData **GetClientData** ()
POINTER TO BYTE **GetClientSpecificData** (POINTER TO STRING pst)
VisuStructSimpleRectangle **GetSurroundingRect** ()
BOOL **GetTextProperties** (POINTER TO VisuStructFont pFont, POINTER TO DWORD pOptAlignment, POINTER TO STRING pText, POINTER TO POINTER TO STRING ppTooltip, POINTER TO DWORD pFlags)
BOOL **GetUpdateRects** (INT elemIndex)
BOOL **HandleInput** (POINTER TO VisuStructEvent pEvent)
BOOL **Initialize** (IVisualization parentVisu)
BOOL **Point** (POINTER TO VisuTbCommandBuffer pPaintBuffer, BOOL pStatic, BOOL bDynamic)
BOOL **SetClientData** (POINTER TO VisuStructClientData pClientData)
BOOL **SetStaticState** ()
BOOL **Update** ()