IEDDE «CifCAD»

ИСПЭИ (Интегрированная Среда Проектирования Электрических Изделий)  
IEDDE (Integrated Electrical Devices Design Environment)

Contents

[Мотивация 2](#_Toc114190579)

[Идея и Цель 2](#_Toc114190580)

[Наброски Требований и Общие Принципы 2](#_Toc114190581)

[Универсальность Инструментария 2](#_Toc114190582)

[Вложенность Проектов 3](#_Toc114190583)

[Параметрическое Конструирование и Когерентность 3](#_Toc114190584)

[Интерактивность 4](#_Toc114190585)

[Настраиваемость и Расширяемость 4](#_Toc114190586)

[Встроенная Кастомизация 5](#_Toc114190587)

[Интуитивная понятность 5](#_Toc114190588)

[Второстепенные аспекты функциональности 5](#_Toc114190589)

[Предполагаемые Инструменты 5](#_Toc114190590)

[Отдельные Моменты и Фичи 6](#_Toc114190591)

[Переменные 6](#_Toc114190592)

[Инструменты 7](#_Toc114190593)

[Редакторы 7](#_Toc114190594)

[Пользовательский Интерфейс 7](#_Toc114190595)

[Предопределённые профили редакторов 8](#_Toc114190596)

[Срезы проектов 9](#_Toc114190597)

[Методика Сбора Требований 10](#_Toc114190598)

[Рабочий Процесс 10](#_Toc114190599)

[«Фиктивные Заказчики» - Тестовые Проекты 10](#_Toc114190600)

[Процесс Разработки 11](#_Toc114190601)

[Примерный План Работ 11](#_Toc114190602)

[Стратегия Разработки 11](#_Toc114190603)

[Предполагаемые Детали Реализации 11](#_Toc114190604)

# Мотивация

Существующие ECAD по типу Altium Designer не предоставляют возможности полностью провести разработку от идеи до производства внутри одной системы. Детальная разработка на уровне компонентов, соединений и дорожек может быть только малой частью процесса проектирования, причём при должной степени тщательности «высокоуровневой» разработки (системного проектирования) подробная разработка платы и особенно схемы, может быть достаточно «тупой» и прямолинейной задачей, так как все существенные вопросы уже решены, и разработчик не спотыкается постоянно о необходимость подумать о каком-то новом неочевидном моменте. Подобная высокоуровневая разработка у нас производится в программах типа Confluence, инструментарий которого не выходит за рамки простого написания текстов, составления таблиц, по сути являющихся другой формой представления текста, и рисования схем, никак не связанных с остальным контентом проекта. При любом изменении может потребоваться вручную просматривать кучу страниц, таблиц и вычислений и что-то там менять, а заодно и в исходниках проекта в Альтиуме. Круто будет сделать единую среду, совмещающую и тексты, и таблицы, и вычисления, и схемы, как простые блок-схемы в draw.io, так и принципиальные как они делаются в мощных современных ECAD. Если такая среда будет интерактивной, с возможностью пронизать проект кучей перекрёстных ссылок на разные типы содержимого, сделать содержимое зависимым от результатов интерактивных вычислений (хоть математических, хоть элементарных логических), можно будет удобно и быстро вносить изменения в проект (менять принятые решения или менять требования) с возможностью автоматической перегенерации всего зависящего содержимого проекта вниз по дереву процесса проектирования вплоть до компонентов на плате.

Содержательный процесс разработки электронных устройств редко ограничивается непосредственной разработкой Э3 и платы в ECAD. Как минимум, необходимо разработать кабельную сеть и схемы подключения множества устройств извне платы. А ещё проекты бывают иерархическими, под установку в шкаф или в стойку… Всё это стоит на одних и тех же физических принципах, и в индустрии есть общие стандарты представления (как минимум, Э3 и Э4 могут быть сильно похожи), при этом мощные полнофункциональные системы проектирования ЭЛЕКТРИКИ (кабеля, шкафы, Э4) наподобие E3 Zuken существуют в полном отрыве от Альтиума. При том, что в самом Альтиуме можно разрабатывать и Э4, и моделировать сборки, но его собственный инструментарий для этого очень ограниченный, медленный и неудобный. Напрашивается создание единой системы проектирования электрики и электроники от Э3 до сборочного чертежа УСТРОЙСТВА, использующего все достоинства обоих упомянутых программных пакетов. А если уж и задаваться задачей такого укрупнения и интеграции, то лучше заодно заложить и функционал сквозного интерактивного высокоуровневого проектирования в эту единую систему.

# Идея и Цель

Создать программный пакет для разработки электрических изделий, больших и маленьких, равноценных по функционалу современным CAD для разработки электрической техники на разных этапах – кабельной сети, распределительных устройств и печатных плат. Навернуть на этот единый CAD полнофункциональный пакет для системного инжиниринга от текстовых набросков до детальной функциональной схемы. Сделать так, чтобы любые данные на самых верхних уровнях (требования, целевые значения по питанию, выбранное абстрактно схемотехническое решение) были взаимозависимы с соответствующими ими данными внизу дерева проекта (футпринты, правила топологии, подписи разъёмов) и могли меняться синхронно. Иными словами, решить своего рода задачу синхронизации потоков, только не выполнения программы, а процесса разработки (при том, что потоки могут быть и на одном горизонтальном уровне – разные платы в одном проекте, так и на разных – абстрактная блок-схема и схемная модель для симуляции. И встроить в этот единый процесс все полезные в электрической индустрии типы математического моделирования.

# Наброски Требований и Общие Принципы

## Универсальность Инструментария

Э4 и Э3 очень похожи. Выполнить визуально и даже функционально похожую схему многоплатного проекта в Альтиуме можно и в обычном редакторе схем. Можно даже что-то наподобие эскиза трассировки кабелей изобразить в редакторе плат, сохраняя зависимость от этой Э4 в схемном редакторе. А структурные и функциональные схемы – это тоже схемы, только рисуем мы их в draw.io, хотя в другой организации могли бы и в Альтиуме.

CifCAD должен предоставить инструментарий для решения самого широкого спектра задач в самых разных отраслях, в конце концов, схемный и платный редакторы – это всего лишь векторная графика. Не должно быть глупостей, когда разные типы схем рисуются в разных редакторах. А широта инструментария открывает хорошие возможности для расширения и кастомизации.

## Вложенность Проектов

Есть проект по автоматизации проектирования электроники, основанный на идее сделать этот процесс похожим на программирование – написание текста на формальном языке и подключение библиотек. Разработка электроники, конечно же никакое не программирование, у нас во главе угла стоят схемы. Но идея обращаться с проектными данными в электронике как с программными исходниками кажется продуктивной. Редактор компонентов в Альтиуме, грубо говоря, подмножество редактора схем. У нас он отдельной сущностью не будет (см. [Универсальность Инструментария](#_Универсальность_Инструментария)), и что отдельный компонент, что дизайн платы будут представлять собой проект. Как большой проект в IDE, подключающий библиотеку (которая тоже сама по себе проект, и используется ещё много где). Проекты плат становятся просто подключаемыми библиотеками для сборок, проекты сборок – для проектов устройств и так далее. Гипотетически даже ничто не мешает сделать проект, где микросхема SOIC-8 будет висеть снаружи шкафа с электрикой и платами в качестве костыля, то есть очень низкоуровневый компонент будет подключаться на самый верх дерева структуры единого изделия.

Не только проекты плат должны быть подключаемыми библиотеками для проекта сборки из этих плат. Платы так или иначе описаны в текстовой документации (у нас она в Confluence), и к проекту одной конкретной платы относится определённый набор высокоуровневых данных (блочки на функциональных диаграммах, параметры, требования, описания выбора решений). У родительского проекта сборки тоже есть такой набор данных. Системные проектные данные плат точно так же подключаются к системным проектным данным сборки, как это происходит с собственно ECAD-ными проектами. Например, вставляются в текст как абзац Ctrl+C Ctrl+V, или подсоединяются одним блочком на блок-схеме, играя таким образом роль компонента на схеме в Альтиуме.

Нет оснований не сделать подобного рода вложенность проектов бесконечной.

## Параметрическое Конструирование и Когерентность

В процессе системной разработки генерируется большое количество проектных данных – параметры (требуемые и фактические, известные из расчётов), структурные и функциональные схемы, являющиеся по сути набросками схем Э3 и Э4. А ещё различные требования, термины и обозначения. В CifCAD при изменении архитектуры не нужно бегать по другим текстовым страницам и везде менять названия составных частей устройства (у них ведь теперь по-новому распределены обязанности и соединения, а значит и названия поменяться могут). Достаточно поменять название один раз. Однажды введённое название можно использовать на других страницах, не набирая его тупо текстом, а создавая активную ссылку на определение. По сути, это объявление и определение переменной и операция Go To Definition. А в нормальных IDE переменные и идентификаторы можно переименовать из любого места. Так же должно быть и с CifCAD.

Если устройству был задан лимит по потреблению, инженер может написать в тексте проектной документации формулы с расчётами и написать в конце предложение, в котором сделает вывод о выполнимости требования по потреблению. Если потом это требование поменяется, ему придётся переписывать все расчёты (ещё даже не зная стоит ли оно того, вдруг такое требование точно никак не выполнить и нужно принципиально иное решение с иным массивом расчётов) и менять резолюцию о выполнении требования в конце документа. В CifCAD достаточно будет создать интерактивный файл расчётов (наподобие MathCAD-овского), сослаться в нём на число в требовании, а в конце этого файла расчётов вывести логическое выражение сравнения двух чисел, в зависимости от которого поставить плашку OK или FAIL.

В Альтиуме есть метки цепей (директивы, формирующие классы цепей), с помощью которого некоторые помечают на схеме потребление по силовым цепям. Такая метка в CifCAD будет прямо ссылаться на нужное число из файла с расчётами и автоматически обновляться. А дальше в редакторе плат можно будет создать правило для минимальной толщины дорожки данного класса, вычислив значение минимальной толщины по заданной инженером формуле IPC, использовав, в конечном счёте, число из файла расчётов. При изменении этого числа автоматически меняется и надпись на принципиальной схеме, и правило топологии платы.

В итоге должна быть возможность автоматически распространить любые данные из любого места в проекте в любом направлении, хоть к предкам по иерархии, хоть к потомкам, хоть по одному уровню. При этом система должна поддерживать консистентность таких данных во всех местах проекта, где на них ссылаются. В Confluence к инструментам, частично реализующим подобное свойство, являются якоря, отрывки и требования Yogi.

## Интерактивность

На этапе системной разработки инженеры часто сравнивают несколько доступных решений или компонентов по ряду параметров. В зависимости от того, какое из сравнивавшихся решений будет принято к реализации, путь дальнейшего проектирования может меняться – могут быть разные схемы, разные подписи и компоненты на плате. Если добавить в CifCAD нативные интерактивные опции для простых примитивов (тех же таблиц сравнения), можно будет, например, переключая радиокнопку в этой таблице сравнения в режиме реального времени смотреть за тем, как меняется функциональная схема в зависимости от принятого решения.

Также в CifCAD должно быть реализовано как можно больше возможностей перенести идеи автоподстановки и автоматического создания бойлерплейт-кода из современных IDE. Например, в тот момент, когда разработчик указал на схеме, что для данного узла будет применено решение в виде схемы дифференциального усилителя, наша среда должна предложить ему поставить туда имеющиеся в библиотеках операционные усилители. А поскольку выбор компонентов предполагается не на этапе выбора решений, а позже, среда должна автоматически создать страницу с описанием выбора компонентов (если таковое предусмотрено пользователем), добавить туда информацию о выбранном инженеров усилителе, и даже сразу расположить его в центре соответствующего листа на принципиальной схеме.

Самый же банальный пример интерактивности, которая должна быть реализована – это автоподстановка терминов (например, названий составных частей изделия) при наборе текста.

Сверхзадача в плане соблюдения принципа интерактивности – освободить инженера от мыслей о том, что конкретно ему нужно делать, какой документ или раздел создавать, какие связи настраивать, на каждом конкретном этапе работ. Конечно, только в случае типового проекта. CifCAD должен предоставлять возможность настроить себя так, чтобы он создавал по максимуму документов и генерировал данных автоматически (минимизировать отношение количества кликов к количеству действий и создаваемой информации) в соответствии с процессом, наиболее подходящим для проекта.

## Настраиваемость и Расширяемость

В первом принципе говорилось, что от системы узкоспециализированных редакторов, как это сделано в нынешних ECAD, мы отказывается, создавая универсально решение. Однако в таком виде имеющаяся среда получится намного более хилой и неудобной, чем Altium, Mentor или Allegro. Есть дихотомия «Универсальность – Эффективность». Для решения этой проблемы необходимо реализовать удобную систему трансляции набора слабых инструментов широкого профиля в набор сильных узкоспециализированных инструментов. Что-то идеологически похожее на микропрограммирование – создание CISC команд из набора RISC инструкций. Достигаться это будет двумя путями:

* Простая кастомизация интерфейса. Ряд самых необходимы для узкого применения инструментов будет вынут из выпадающих списков и панелей в контекстное меню, Active Bar и горячие клавиши. В общем, из труднодоступных мест будет выноситься на поверхность набор самых нужных для данного случая инструментов. Нужно активно использовать горячие клавиши. Заметим, что, здесь, в отличие от большинства случаев, горячие клавиши могут быть настроены даже ВНУТРИ одного редактора, а не только различаться между редакторами схемы и платы, как в Альтиуме.
* Создание собственных инструментов. Это задел на расширяемость. Какими-либо встроенными или внешними программными средствами создаётся новый инструмент. После этого он становится доступным в базовом наборе инструментов редактора, и может быть «вынесен» на поверхность также, как и в предыдущем случае.

Пользователю должна быть предоставлена возможность к неограниченному расширению CifCAD с помощью удобных инструментов. Одним из аспектов такого удобства должны быть штатные возможности для интеграции CifCAD со следующими типами продуктов:

* Системы компьютерной алгебры (например Matlab)
* CAE для моделирования электрических схем (например Multisim)
* CAE для моделирования физический полей (например ANSYS или Sigrety)
* MCAD (например Solid Works)
* PLM (например Team Center)
* Системы управления проектами (например Jira)
* Вики-системы (например Confluence)
* Системы управления цепочками поставок
* Системы контроля версий (например GitLab)
* Сервера управляемых данных (например Altium 365)

Тщательное соблюдение этого принципа позволит использовать CifCAD в ряде сторонних индустрий - проектирования гидравлики, пневматики, теплораспределения, слаботочки, электрораспределительных устройств («шкафнины»).

## Непрерывная Валидация

Есть несколько разных способов ответить на вопрос о том, валиден ли разработанный проект - жизнеспособен ли он и выполняет ли поставленную перед ним задачу. К таким способам, распространённым в практике разработки электрических изделий, в том числе, с помощью ECAD, относятся:

* Просмотр входных требований и ручная проверка их выполнения
* Компиляция схемы
* Проверка правил топологии платы
* Моделирование схемы
* Анализ целостности сигналов
* Ручная или автоматизированная проверка синхронности данных между схемой и платой
* Проверка состояний жизненного цикла компонентов, шаблонов, переиспользуемых фрагментов схемы и платы

В CifCAD будет возможность собрать воедино все эти способы. Будет возможность самостоятельно определить, какие именно аспекты валидности должны проверяться и как именно проверяться – автоматически с некоторой периодичностью или по ручному запуску. Также должна быть возможность явно информировать пользователя о валидности проекта, например, подсвечивать красным файлы с ошибками или выдавать предупреждения при генерации выходной технологической документации.

Должно быть два типа валидаторов. Первый – синхронизаторы – инструменты, поверяющие когерентность данных между разными частями проекта (например, все ли компоненты выгружены со схемы на плату) или между проектом и внешними зависимостями (например, актуальная ли или допустимая ли версия компонента используется в проекте). Второй – верификаторы, которые проверяют истинность утверждений, сформулированных на встроенном формальном языке из констант и проектных данных.

## Встроенная Кастомизация

Поскольку наш продукт позиционируется как ECAD, в нём под каждый конкретный аспект работы по разработке электроники и электрики должны быть предустановленные кастомизации с инструментами, расширяющими базовый набор простых средств редактора в соответствии с принципом [Настраиваемости и Расшряемости](#_Настраиваемость_и_Расширяемость).

## Workflow-Driven Development

Набор функций и элементов интерфейса, предоставляемых CifCAD, реализуется в предположении, что разработка электрического изделия следует определённому рабочему процессу (см. [Рабочий Процесс](#_Рабочий_Процесс)). Кастомизация (причём, желательно, встроенная) должна помогать пользователю следовать этому процессу, по возможности подталкивать его к нужным в данный момент действиям и генерировать нужные в данный момент данные, документы и шаблоны.

## Интуитивная Понятность

Помимо всего прочего, данному принципу в том числе служит идея «экстракции» инструментов и конфигурирования редакторов под конкретный частный тип задач, встречающийся при проектировании электрических изделий (см. [Настраиваемость и Расширяемость](#_Настраиваемость_и_Расширяемость)).

## Системные Требования

CifCAD должен работать под Windows 10. Второстепенный приоритет – порт на Linux.

Обобщённо говоря, программа будет требовательная к количеству оперативной памяти.

## Второстепенные Аспекты Функциональности

Невозможно поставить всё во главу угла. Следующие элементы инструментария ECAD считаются в настоящем проекте второстепенными. По возможности их необходимо реализовывать в полноценном и удобном виде, однако в рамках программы минимум они считаются принесёнными в жертву:

* Отслеживание информации на рынках
* Контроль версий (VCS)
* Управление проектными данными (PDM)
* Командная работа

Однако исходя из пожеланий к системе, а именно, её удобства для централизованной разработки больших проектов, часть функционала вышеупомянутых систем должна быть реализована в обязательном порядке и в удобном для пользователя виде. CifCAD должен предоставлять следующие возможности:

* Централизованное хранение данных
* Фиксирование версий
* Подключение подпроектов как независимо управляемых наборов данных
* Разграничение прав доступа к отдельным частям проекта

# Отдельные Моменты и Фичи

## Проекты

Каждый проект обладает следующими характеристиками:

* **Иерархичность** – каждый проект может быть включён в несколько других проектов, а также может включать в себя несколько других проектов
* **Доменная связанность** – каждый проект может иметь несколько параллельных наборов данных, каждый из которых представляет аспекты проекта в какой-либо конкретной предметной области либо на каком-либо конкретном этапе жизненного цикла продукта (и то и другое называется **домены**). Один такой доменный набор данных может иметь по несколько связанных с ним наборов данных из нескольких других доменов.
* **Нелокальность** – части проекта, относящиеся к нижестоящим иерархическим уровням или к другим доменам могут храниться, верифицироваться, редактироваться и отображатьсяпараллельно с данным проектом в других потоках или на других машинах.

## Переменные

Архитектура и функциональность среды следует идеологеме «Всё переменные». Включая проекты и проектные данные.

**Переменная** – это любое поле данных определённого типа, имеющее заданный идентификатор определённого типа.

Конкретный формат как идентификатора, так и поля данных (значения) может определять пользователь. Доступны текстовый тип, числовой тип (может потребоваться тщательный рефакторинг при интеграции с CAE-системами), числовой тип с единицей измерения (с предопределёнными префиксами по системе СИ, можно добавлять новые единицы измерения), ссылка на другую переменную, упорядоченная комбинация всех допустимых типов. В самом общем случае тип любого значения можно рассматривать как массив ячеек.

Переменные можно объявлять один раз в проекте. **Объявление** – создание нового идентификатора.

Переменные можно инициализировать один раз в проекте. **Инициализация** – присвоение значения существующему идентификатору.

Одномоментное и одноместное объявление с определением переменной называется **Определением.**

На значение переменной можно ссылаться бесконечное количество раз в любом месте проекта.

Следовательно, предполагается несквозная компиляция проекта.

Все переменные **глобальны** по умолчанию – доступны из любого места в текущем проекте, но не доступны извне проекта. Например, переменная, объявленная в файле с требованиями, будет видна даже из .txt файла программы для сверления. Однако переменная, объявленная в библиотечном проекте, непосредственно из другого проекта недоступна. Чтобы сделать её доступной, нужно подключить проект в проект как библиотеку. У нас как раз тот случай, когда глобальные переменные являются благом, а не злом, так как нужно не устранить, а наоборот максимизировать возможность конфликта имён – в нормальном проекте вообще не должно быть неидентичных сущностей с одинаковыми именами.

Все переменные по умолчанию **константны**, они описываются в одном конкретном месте и дальше просто переиспользуются. Проект – это не исполнение программы, а её исходный код, поэтому, если содержимое переменной меняется, то всё зависящее от неё перевычисляется. В проекте больше ничего не опирается на её старое значение, а всё зависящее от неё определяется исключительно её нынешним значением.

Любая переменная, объявленная не среди уникальных для данного проекта документов, а подключенная вместе с подпроектом (например параметр компонента или распиновка разъёма в универсальном модуле), является «**вдвойне константной**» ей не просто нельзя присваивать новые значения, ей даже нельзя поменять определение, так как из проекта нельзя поменять библиотечные данные. По крайней мере, с той же простотой, что и собственные данные проекта. Такую возможность можно предоставить, но с предупреждением; также следует помнить, что такое изменение может повлечь за собой целый каскад действий – валидация данных переменной, если для данного типа она предусмотрена, настроена и включена, генерация и релиз на сервер новой ревизии переменной (или включающего её пакета), обновление проектных данных и перевалидация всех проектов, которые используют данную «библиотеку». Всё прямо как в большом IT.

В любом месте проекта переменные можно **разыменовывать** – получать значение по ссылке. Это может быть как использование значения в интерактивной формуле (файле интерактивных расчётов или правиле топологии), так и просто вставка значения в текст, ячейку таблицы или поле параметра какого-либо примитива схемного (или топологического) редактора.

Нужно обеспечить удобные, быстрые, интерактивные и интуитивно понятные возможности для инициализации и разыменовывания переменных в документах проекта. Это должно выглядеть как обращение к полям сложных типов, если нужно конкретное числовое значение одного из полей подставить в формулу. В общем случае это может потребовать дополнительного исследования UX.

Заметка на полях: на продвинутом уровне в системах интерактивных вычислений может потребоваться исполнение алгоритмов вместо вычисления формул, поэтому можно допустить, что одной из дополнительных фичей будет возможность **присваивания** значения переменной – принудительного задания её значения. Тогда это потребует введения на плоской шкале проектных данных дополнительного измерения «времени выполнения», или полного упорядочивания присваиваний и разыменований для того, чтобы соблюсти когерентность проектных данных.

Для реализации описанных выше [принципов](#_Наброски_Требований_и) и исходя из описанных в этой главе механизмов все переменные должны быть **статическими**. Однако памятуя о том, что в системах компьютерной алгебры типа MathCAD реализована возможность программирования «среди» формул, стоит задуматься о том, чтобы всё-таки сделать возможность определять **динамические** переменные там, где это необходимо. **Автоматическим** же переменным в проектах без времени выполнения явно не место.

## Подсистемы

4 **редактора** «над капотом» - непосредственно доступные пользователю с удобным интерфейсом:

* **Текстовый редактор** (Text Editor, TE)
* **Редактор схем** (Diagram Editor, DE)
* **Топологический векторный редактор** (Topological Vector Editor, TVE)
* **Система трёхмерного моделирования** (3-Dimensional Modeling System, 3DMS)

3 **подсистемы «под капотом»** - если и доступны пользователю, то не с таким удобством как редакторы. Предполагается, что редакторы будут использовать эти инструменты в своих целях, потому к ним всё же будет удобный доступ, но не непосредственный, а через редакторы:

* **Система компьютерной алгебры** (Computer Algebra System, CAS)
* **Система моделирования динамических систем** (Dynamic Systems Simulation System, DSSS)
* **Система конечноэлементного моделирования физических полей** (Physical Fields Finite Element Simulation System, PFFESS)

Система компьютерной алгебры нужна на случай больших массивов вычислений, которые нельзя просто так быстро посчитать в потоке редактора. DSSS – штука наподобие Simulink или Multisim, используется для моделирования ЦЕПЕЙ – электрических, тепловых или радиотехничечских (с S-параметрами). PFFESS – вещь наподобие CAE инструментов Solid Works, нужна для моделирования тепловых полей, полей растекания токов и прочего анализа целостности сигналов на топологии платы, в продвинутом случае – для моделирования электромагнитных полей платы и нагружённого состояния.

## Редакторы

* Могут определять, использовать и демонстрировать [переменные](#_Переменные)
* Могут [на лету](#_Параметрическое_Конструирование_и) вычислять значения
* Есть возможность [создавать профиль](#_Настраиваемость_и_Расширяемость) редактора. **Профиль** – в том числе ограниченный набор из всех доступных инструментов, который делается доступным из панелей инструментов быстрого и удобного доступа и из горячих клавиш. Возможно, профиль, в том числе – определённая схема поведения подсказок и автодополнения
* У каждого редактора есть [предопределённые профили](#_Предопределённые_профили_редакторов)
* Можно задавать **ограничение рабочей области**. Например:
  + В текстовом редакторе:
    - Основной режим – сверху и слева (для арабо-израильской локализации – справа) ограничено (как в Confluence или электронных таблицах)
    - Ограничено по бокам (бесконечная лента)
    - Ограничено по 4-м измерениям (лист)
  + В редакторе схем:
    - Неограниченно (для редактора компонентов или диаграмм)
    - Ограничено по 4-м измерениям (лист)
  + В топологическом векторном редакторе:
    - Неограниченно (плата, жгут)
    - Ограничено по 4-м измерениям (лист)
  + В системе трёхмерного моделирования:
    - Неограниченно в 6-ти измерениях (сборка)

## Пользовательский Интерфейс

Есть три **общих вида пользовательского интерфейса**:

* **Панели инструментов общего вида**
* **Скриптовый язык запросов** *(низкий приоритет)*
* **Низкоуровневое API**

Есть три вида **быстрого пользовательского интерфейса**:

* **Панели инструментов быстрого доступа**
* Контекстные меню
* Горячие клавиши

## Предопределённые Профили Редакторов

*(по максимуму, возможны сокращения)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Редактор** | **Профиль** | **Аналоги** | **Расширение файла** |
| **Текстовый редактор** | **Текстовый документ** | Word, Confluence, NotePad++ | .txtCif |
| **Электронная таблица** | Excel | .csvCif |
| **Интерактивный файл расчётов** | MathCAD | .clcCif |
| **Редактор схем** | **Граф** | ??? | .gphCif |
| **Эскиз** | CorelDraw, Qt Designer | .dftCif |
| **Структурная схема** | draw.io, Visio | .blcCif |
| **Функциональная схема** | draw.io, Visio | .fncCif |
| **Принципиальная схема** | AD Schematic Editor | .schCif |
| **Схема соединений** | AD Multiboard Schematic Editor | .conCif |
| **Схема жгута** |  | .hrnCif |
| **УГО** | AD SCHLIB Editor | .symCif |
| **Топологический векторный редактор** | **Рисунок** | CorelDraw, Qt Designer | .skcCif |
| **Печатная плата** | AD PCB Editor | .pcbCif |
| ***Геометрическое представление жгута*** |  | *.kntCif* |
| **Чертёж** | AD Draftsman, КОМПАС 3D, AutoCAD | .drwCif |
| **Компонент** | AD PCBLIB Editor | .ftpCif |
| **Система трёхмерного моделирования** | **Сборка** | SolidWorks, Fusion 360, Autodesk Inventor | .asmCif |
| **Трассировка проводов** |  | *.wirCif* |
| **Деталь** | SolidWorks, Fusion 360, Autodesk Inventor | .bdyCif |

## Базовые Примитивы

### Для Текстового Редактора

* **Текстовое поле** (Text Field)
* **Электронная таблица** (Spreadsheet)
* **Блок вычисления** (Calculation Block)
* **Блок выбора** (Selection Block)
* **Рисунок** (Picture)

Имеет смысл подумать об унификации синтаксиса формул в таблицах и вычислениях.

Блоки выбора имеют самое разное визуальное представление – иногда удобно делать их в виде радиокнопок, иногда в виде выпадающих списков.

### Для Редактора Схем

* **Узел** (Node)
* **Точка подключения** (Connection Point)
* **Соединение** (Connection)

Примитивы здесь не равноправны, так как точки подключения принадлежат узлам. Также здесь есть **вспомогательные примитивы**:

* **Передатчик соединения** (Connection Transmitter)
* **Зона** (Zone)

Зона – это просто фиксированное объединение блоков, отображаемое в виде «легковесного» надблока. Передатчики соединения в общем виде бывают двух видов:

* **Шина** (Bus)
* **Порт** (Port)

Порты подразумевают также и Power Port, и Off Sheet Connector. Шина в общем случае с кастомным индексом, поэтому жгут здесь - разновидность шины.

### Для Топологического Векторного Редактора

* **Дорожка** (Track)
* **Полигон** (Polygon)
* **Надпись** (Inscription)

Контактные площадки реализуются через полигоны.

### Для Системы Трёхмерного Моделирования

* **Твёрдое тело** (Solid Body)
* **Упругое тело** (Elastic Body)
* **Тонкостенная пластина** (Thin-Walled Plate)
* **Нитевидное тело** (Threadlike Body)

Упругое тело служит для представления пружин и «язычков» в защёлкивающихся соединениях. Тонкостенная пластина служит для представления гибких участков печатных плат. С помощью нитевидных тел реализуются различные провода, жгуты и кабели.

## Срезы Проектов

**Срез** – это набор проектных данных определённого типа. В панелях навигации по проекту есть возможность отображать только те данные (документы, файлы), которые относятся к данному срезу.

|  |  |
| --- | --- |
| **Срез** | **Примеры Содержания** |
| **Замысел** | * Спецификация проекта (TE) * Входные требования (TE) * Функциональные требования (TE) * Технические требования (TE) * Дерево трассировки требований (DE) |
| **Проектный облик** | * Описание процедуры синтеза архитектуры (TE) * Пошаговое описание процедуры принятия проектных решений (TE) * Описание выбора решений (TE) * Описание выбора компонентов (TE) * Проектировочный расчёт (TE) * Проверочный расчёт (TE) * Структурная схема (DE) * Функциональная схема (DE) * Схема подсистемы (DE) * Диаграмма состояний (DE) * Эскиз интерфейса (DE) |
| **Исходные данные** | * Схема соединений (DE) * Принципиальная схема (DE) * Файл платы (TVE) * Схема жгута или кабеля (DE) * Файл геометрического представления жгута или кабеля (TVE) * Сборка (3DMS) |
| **Подготовка производства:**  **Изготовление** | * Чертёж-спецификация платы (TVE) * Файлы производства платы (TVE) * Чертёж мультизаготовки (TVE) * Файлы производства мультизаготовки (TVE) * Чертёж жгута или кабеля (TVE) * Перечень элементов и расходных материалов жгута или кабеля (TE) |
| **Подготовка производства:**  **Сборка** | * Перечень элементов платы (TE) * Чертёж трафарета (TVE) * Файл топологии трафарета (TVE) * Монтажная схема платы (TVE) * Файлы автоматизированной сборки платы (TVE) * Чертёж финишных защитных покрытий платы (TVE) * Файлы топологии финишных покрытий платы (TVE) * Сборочный чертёж узла (TVE) * Перечень компонентов и расходных материалов узла (TE) * Схема трассировки проводных соединений узла (TVE) * Описание процедуры приёмочного тестирования изделия (TE) |
| **Отладка** | * Описание процедуры проверки требований к изделию (TE) * Дефектная ведомость (TE) |
| **Внедрение** | * Техническое описание (TE) * Руководство по эксплуатации (TE) * Входные данные для программирования (TE, DE) |

## Релизы

В разрабатываемом продукте будет такая сущность как релиз проекта – момент, строго определяемый пользователем, когда вся проектная информация фиксируется как завершённая. В этот момент должны произойти следующие действия:

* Сохранение всех текущих правок в открытых документах проекта
* Генерация всех полностью автоматически генерируемых файлов (таких как Gerber)
* Полная валидация проекта (запуск всех настроенных пользователем валидаторов)
* Загрузка актуальной версии всех проектных данных в централизованное хранилище

Если хотя бы на одном из этих этапов произошёл сбой (в случае с валидацией – если проект её не прошёл), система должна активно сигнализировать об этом пользователю, а релиз будет считаться несостоявшимся.

# Методика Сбора Требований

## Рабочий Процесс

Методика сбора требований основана на следующем воркфлоу разработки:

1. Описание идеи
2. Формулирование требований
   1. Сбор требований
   2. Анализ требований
3. Проектирование
   1. Синтез структуры
   2. Выбор решений
   3. Выбор компонентов
4. Разработка
   1. Разработка принципиальной схемы
   2. Разработка геометрии и топологии
5. Подготовка производства
   1. Разработка документации для изготовления деталей
   2. Разработка документации для сборки
   3. Разработка документации для функционального тестирования и настройки
6. Отладка
7. Внедрение
   1. Разработка документации для смежных систем
   2. Разработка пользовательской документации

## «Фиктивные Заказчики» - Тестовые Проекты

Собираем требования, делая вид, как будто у нас есть несколько заказчиков, каждый их которых хочет нашу систему для своего проекта, и у каждого из проектов свои особенности.

Прогнать по всем релевантным этапам несколько типов проектов:

* Подсистема питания печатной платы с системного до схемного уровня
* Простой проект платы от структурной схемы до CAM-файлов
* Многоплатное устройство в корпусе
* Многоуровневая система вплоть до принципиальных схем
* Одноплатное устройство в корпусе с DFMEA от требований до сборочной документации
* Электрика в квартире

# Процесс Разработки

## Примерный План Работ

1. Описать замысел (написать Design Specification)
2. Собрать входные требования, прогоняя тестовые проекты по этапам воркфлоу
3. Добыть какой-либо проект домовой электрики, чтобы также его прогнать (связаться с Максом Захаровым)
4. Агрегировать входные требования в единый перечень
5. Узнать и изучить какую-нибудь программу для прототипирования GUI
6. Добавить в единый перечень требований картинки с прототипами интерфейса
7. Проектирование, в том числе:
   1. Синтез архитектуры со схемами
   2. Выбор моделей данных
   3. API заготовки редактора
8. Разработать заготовку редактора

*(следующие два пункта могут меняться местами)*

1. Создать прототип ECAD с Э4
   1. Экстрактировать требования
   2. Разработать
   3. Проверить на тестовом проекте
2. Создать прототип CAD для системного проектирования
   1. Экстрактировать требования
   2. Разработать
   3. Проверить на тестовом проекте

*(после начала разработки на первом из двух шагов выше, осуществить следующие два шага)*

1. Попробовать маркетинг
2. Пересмотреть требования и план
3. Ревизия плана и требований
4. Мне исполняется 30 лет

## Стратегия Разработки

В трилемме «Скорость исполнения – Требования к памяти – Расширяемость» жертвовать памятью.

## Предполагаемые Инструменты

### Кодовая База

Open source – Libre Office с их текстовым и табличным редакторами, свободный текстовый редактор для программистов с автоподстановкой (видимо Vim), TeX, открытые ECAD для плат (KiCAD?), открытые системы 3-мерного моделирования, Wolfram Alpha, какая-то свободная среда компьютерной алгебры (урезанные версии Matlab или его более-менее вызывающие доверие аналоги – *подумать про Matlab Runtime Library и GNU Octave*) свободные системы моделирования в электронике (разнообразные SPICE, MicroCap, RFsim99)

### Языки Программирования

C++, Python, вероятно C#, вероятно F#

### Система Хранения Версий:

Git

## Предполагаемые Детали Реализации

* Проект = Дерево ссылок на подпроекты (включая компоненты) и содержимое, уникальное для данного Проекта
* Модель – Представление – Контроллер
* POSIX-совместимое Windows API?
* Нужно обеспечивать когерентность данных между:
  + Моделями и их представлениями
  + Файлами одного проекта
  + Проектом и централизованным хранилищем