

**Стандарт на проектирование ПО**  
**«Модуль OCR (оптического распознавания текста)»**

**Версия 1 от 2019-10-20**

**Москва 2019 г.**

## СОДЕРЖАНИЕ

<b>1</b>	<b>ВВЕДЕНИЕ .....</b>	<b>3</b>
1.1	Назначение и область применения .....	3
1.2	Ссылки .....	3
1.3	Термины, определения и сокращения .....	3
1.4	Соглашения.....	3
<b>2</b>	<b>РЕЗУЛЬТАТЫ ПРОЦЕССА ПРОЕКТИРОВАНИЯ. КЛЮЧЕВЫЕ ПОНЯТИЯ.....</b>	<b>4</b>
2.1	Лица, заинтересованные в описании проекта .....	5
2.2	Архитектура .....	5
2.2.1	Модульные представления	5
2.2.2	Динамические представления	7

# 1 ВВЕДЕНИЕ

## 1.1 Назначение и область применения

Цель данного документа – определить требования и ограничения к результатам процесса проектирования ПО «Библиотека обработки изображений и распознавания символов». А также дать инженерам, участвующим в процессе проектирования, рекомендации по разработке архитектуры ПО и требований низкого уровня.

Положениями настоящего документа необходимо пользоваться при разработке архитектуры и требований низкого уровня ПО «Библиотека обработки изображений и распознавания символов».

## 1.2 Ссылки

- [1] СТАНДАРТ НА ТРЕБОВАНИЯ
- [2] САМИ ТРЕБОВАНИЯ (ВЫСОКОГО УРОВНЯ)
- [3] СЕРТИФИКАЦИОННЫЕ ТРЕБОВАНИЯ
- [4] UML

## 1.3 Термины, определения и сокращения

Таблица 1 Термины и определения

Термин	Определение
Модуль	Элемент архитектуры ПО, объединяющий некоторое множество подпрограмм, типов данных и данных
Активный компонент	Элемент архитектуры ПО, обладающий собственным потоком выполнения.
Средство коммуникации	Средство взаимодействия и синхронизации работы активных компонентов.

Таблица 2 Сокращения

Сокращение	Определение
ЖЦ	Жизненный цикл
ПО	Программное обеспечение
СТПО	Спецификация требований к ПО
УК ПО	Управление конфигурацией ПО

## 1.4 Соглашения

Положения стандарта, содержащие слова «*должен*», «*запрещено*» или их производные формы, являются требованиями к результатам процесса проектирования. Требования обязательны для выполнения.

Положения, содержащие слово «*рекомендуется*» или его производные формы, являются рекомендациями. Рекомендации не обязательны для выполнения, но следование им может повысить качество результатов процесса проектирования.

Для лучшего понимания положений стандарта, правила и рекомендации могут содержать пояснения, раскрывающие детали применения правил и рекомендаций, обоснования правил, а также примеры, иллюстрирующие предложенные правила и рекомендации.

## 2 РЕЗУЛЬТАТЫ ПРОЦЕССА ПРОЕКТИРОВАНИЯ. КЛЮЧЕВЫЕ ПОНЯТИЯ

На рисунке 1 представлены результаты процесса проектирования. В соответствии с [1-3], основной результат процесса проектирования – документ «Описание проекта ПО», в котором даётся описание архитектуры ПО и формулируются требования низкого уровня.

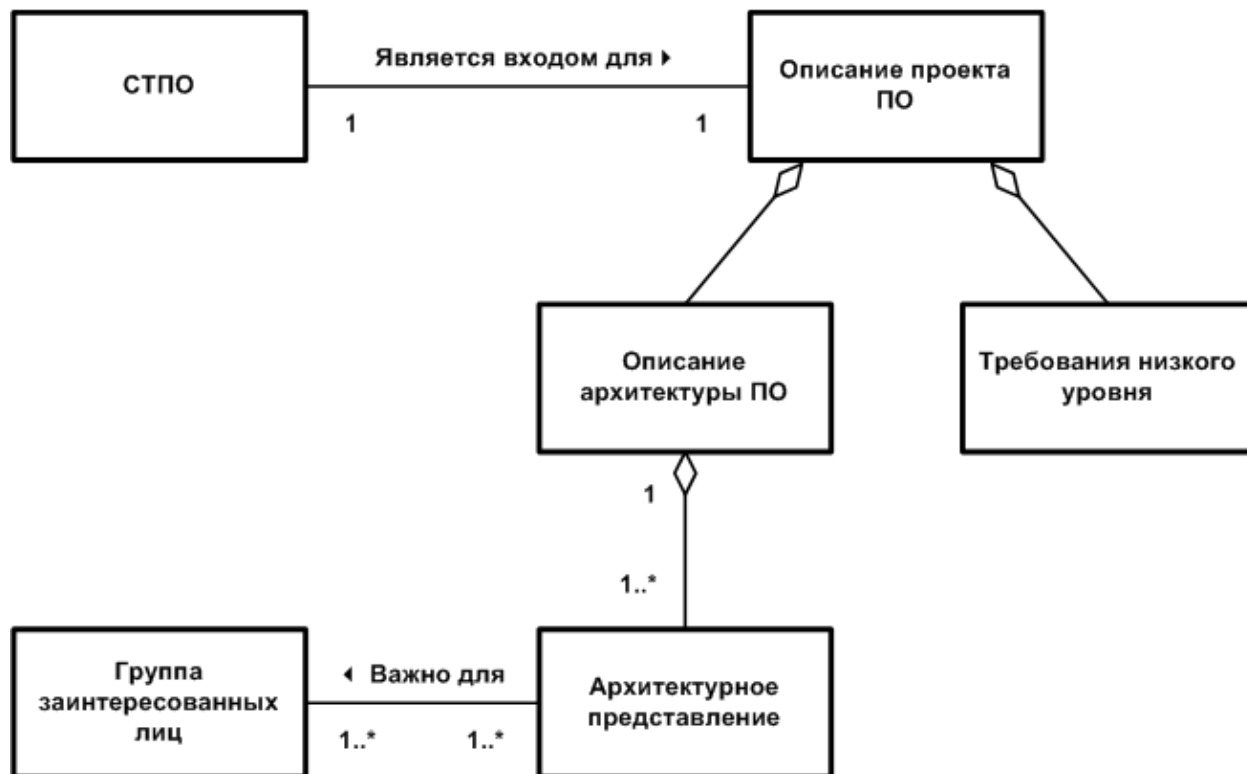


Рисунок 1 Результаты процесса проектирования

В соответствии с [1-3] в описании проекта ПО заинтересованы разные группы лиц, участвующие в процессах ЖЦ ПО. Каждая из этих групп интересуется не всеми аспектами проекта, а лишь теми, что необходимы ей для достижения ее собственных целей. Например, инженеры, разрабатывающие исходный код, используют описание проекта ПО только для получения информации, необходимой для создания исходного кода. А инженеры, участвующие в процессе интеграции ПО, используют описание проекта ПО только для получения информации, необходимой для компиляции и сборки исполняемого кода и загрузки исполняемого кода в память целевого вычислителя.

Для отображения информации, необходимой всем группам заинтересованных лиц, в современной программной инженерии принято рассматривать архитектуру ПО с разных точек зрения - например, поведенческой, структурной, физической. Каждая точка зрения порождает одно или несколько архитектурных представлений - модель архитектуры ПО, в которой отражены лишь существенные аспекты и опущено все, что несущественно при данном взгляде на систему.

В совокупности, архитектурные представления составляют полное описание архитектуры ПО. Архитектурные представления дополняются определением поведения элементов, составляющих архитектурные представления. Некоторые из этих определений становятся требованиями к элементам архитектурных представлений – требованиями низкого уровня.

Каждая группа заинтересованных лиц использует необходимые им архитектурные представления для получения нужной информации.

## 2.1 Лица, заинтересованные в описании проекта

В соответствии с [2-3], следующие группы лиц заинтересованы в описании проекта ПО:

- участники процесса кодирования;
- участники процесса интеграции;
- участники процесса проектирования;
- участники процесса верификации;
- участники процесса управления проектом.

## 2.2 Архитектура

Архитектурное представление состоит из архитектурных элементов определенного типа и отношений между ними.

Различные архитектурные представления, используемые в программной инженерии для описания архитектуры ПО, можно сгруппировать по признаку отображаемой ими информации.

В данном стандарте рассматриваются и предлагаются к использованию при проектировании и документировании описания проекта, следующие типы архитектурных представлений:

- модульные представления;
- динамические представления;

### 2.2.1 Модульные представления

Задача данного представления – показать архитектуру ПО с статической точки зрения, как набор элементов исходного кода.

#### 2.2.1.1 Элементы

Элементы представлений данного типа – *модули*.

Состав модулей определяет состав файлов исходного кода - каждый модуль реализуется в одном или нескольких файлах. Примеры модулей – файл языка Си, пакет языка Ада, класс в объектно-ориентированном языке программирования.

Чаще всего, лишь часть подпрограмм, типов данных и данных, определенных в модуле, доступны другим модулям для использования. Такие подпрограммы, типы данных и данные составляют *предоставляемый интерфейс* модуля.

Подпрограммы, типы данных и данные, используемые модулем, но определенные в другом модуле, составляют *требуемый интерфейс* модуля.

Подпрограммы, типы данных и данные, определенные в модуле, но не вошедшие в предоставляемый интерфейс, образует *скрытую часть* модуля. Скрытая часть обычно защищается от использования другими модулями, с помощью средств языка программирования на котором разрабатывается исходный код модуля.

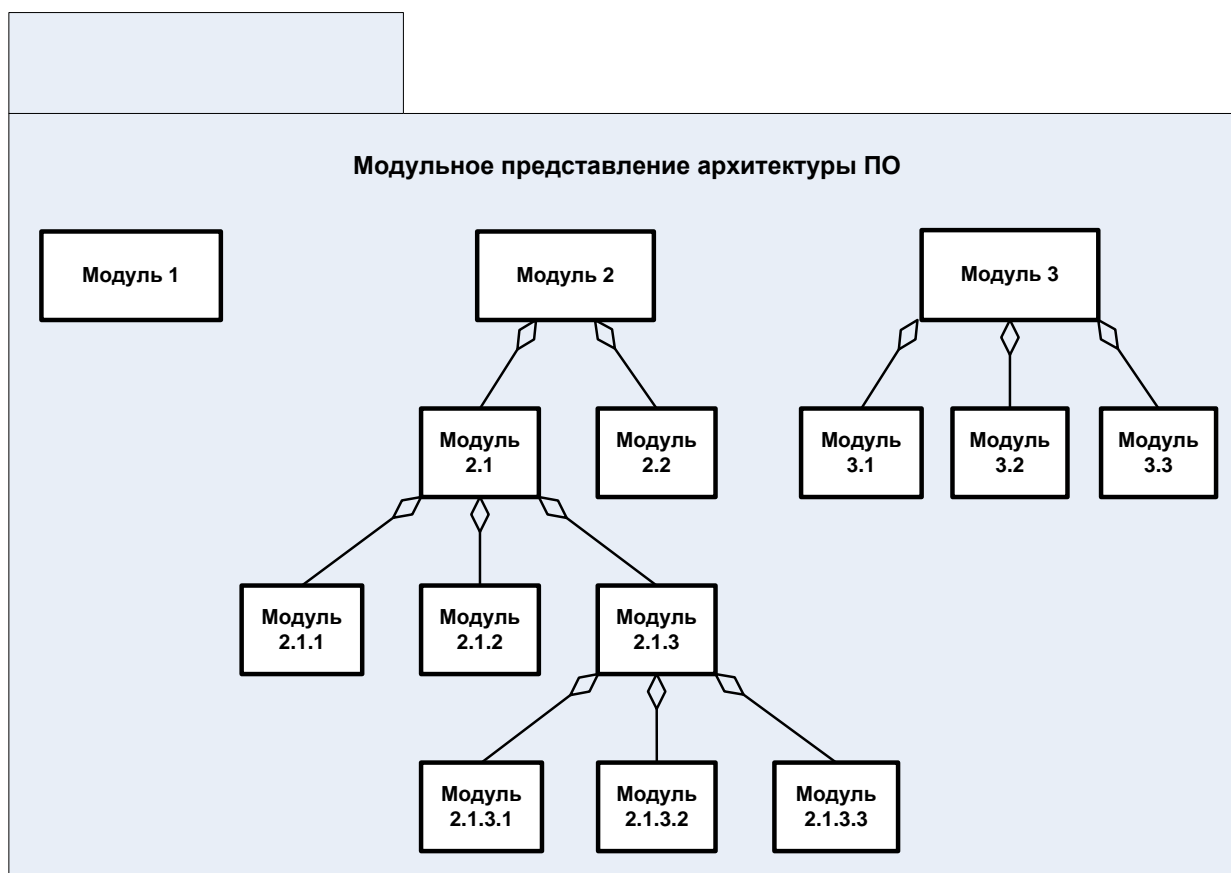
#### 2.2.1.2 Отношения между элементами

Между модулями могут быть следующие отношения:

- *зависимость*. Если корректность работы некоторого модуля зависит от корректности работы другого модуля, то данный модуль зависит от другого модуля. Пример зависимостей между модулями – один модуль вызывает подпрограмму, входящую в предоставляемый интерфейс другого модуля или обращается к данным, входящим в предоставляемый интерфейс другого модуля;

- **часть-целое.** Если для удобства проектирования или анализа принятых проектных решений, некоторый модуль рассматривается как часть другого модуля, то между этими двумя модулями существует отношение часть-целое. Модуль, являющийся частью другого модуля – **подмодуль**;
- **наследование.** Данный вид отношения характерен для объектно-ориентированной технологии. Если класс-потомок использует свойства, определяемые в родительском классе, то класс-потомок наследует классу-родителю;
- **делегирование интерфейса.** Если подпрограмма или данные, входящие в предоставляемый интерфейс некоторого модуля, полностью определены в одном из его подмодулей, то модуль-целое делегирует предоставляемый интерфейс своему подмодулю. Если подпрограмма или данные входящие в требуемый интерфейс некоторого модуля, используются его подмодулем/подмодулями, то модуль-целое делегирует требуемый интерфейс своему подмодулю/подмодулям.

С помощью отношений часть-целое модули образуют иерархическую структуру – лес, корнями деревьев которого являются **модули верхнего уровня**. Вершинами второго уровня в данных деревьях являются подмодули модулей верхнего уровня, вершинами третьего уровня – подмодули модулей-вершин второго уровня и т.д.



**Рисунок 2 Пример модульной иерархии**

На рисунке 2 приведен пример модульной иерархии.

Выделено три модуля верхнего уровня **Модуль 1**, **Модуль 2**, **Модуль 3**. Данные модули являются модулями *одного уровня декомпозиции*.

Модуль **Модуль 2** состоит из подмодулей **Модуль 2.1** и **Модуль 2.2**. Данные подмодули являются модулями *одного уровня декомпозиции*.

Модуль **Модуль 3** состоит из подмодулей **Модуль 3.1**, **Модуль 3.2** и **Модуль 3.3**. Данные подмодули являются модулями *одного уровня декомпозиции*.

Модуль **Модуль 2.1** состоит из подмодулей **Модуль 2.1.1**, **Модуль 2.1.2** и **Модуль 2.1.3**. Данные подмодули являются модулями *одного уровня декомпозиции*.

Модуль **Модуль 2.1.3** состоит из подмодулей **Модуль 2.1.3.1**, **Модуль 2.1.3.2** и **Модуль 2.1.3.3**. Данные подмодули являются модулями *одного уровня декомпозиции*.

### **2.2.1.3 Информация для заинтересованных лиц**

Модульные представления дают следующую информацию об архитектуре ПО:

- состав файлов исходного кода, подпрограммы, данные и типы данных, определенные в файлах исходного кода. Используется для разработки исходного кода участниками процесса кодирования. Используется при компиляции ПО участниками процесса интеграции;
- зависимости между модулями. Используется при рассмотрении изменений участниками процесса УКПО и участниками процесса проектирования;
- распределение требований высокого уровня между модулями. Используется при рассмотрении изменений участниками процесса УКПО. Используется при проверке соответствия архитектуры ПО требованиям высокого уровня участниками процесса верификации. Используется участниками процесса управления проектом для оценки трудоемкости каждого модуля и распределения дальнейшей работы над каждым модулем между участниками процессов ЖЦ ПО;
- интерфейсы модулей, их программное окружение. Используется при проведении испытаний низкого уровня и испытаний интеграции ПО участниками процесса верификации.

## **2.2.2 Динамические представления**

Задача данного представления – показать архитектуру ПО с динамической точки зрения – как набор взаимодействующих элементов, каждый из которых обладает собственным потоком выполнения.

### **2.2.2.1 Элементы**

Элементы представлений данного типа – *активные компоненты* и *средства коммуникации*.

Примеры активных компонент – задачи и потоки в многозадачной операционной системе, обработчики прерываний. Примеры средств коммуникации – очереди сообщений, семафоры.

Подобно модулям, элементы представлений динамического типа - активные компоненты и средства коммуникации, обладают предоставляемым и требуемым интерфейсами.

Операции элемента динамического представления, доступные для использования другим элементом и данные выдаваемые элементом составляют предоставляемый интерфейс элемента динамического представления.

Операции и данные, требуемые элементом динамического представления, но предоставляемые другими элементами, составляют требуемый интерфейс элемента динамического представления.

#### 2.2.2.2 Отношения между элементами

Между элементами представления могут быть следующие отношения:

- **соединение.** Одно средство коммуникации может быть соединено с двумя или более активными компонентами;
- **часть-целое.** Любой активный компонент или средство коммуникации может иметь сложную структуру и, в свою очередь, состоять из других компонентов и средств коммуникации. Элемент, входящий в состав другого элемента – **подэлемент**;
- **делегирование интерфейса.** Если операция или данные, входящие в предоставляемый интерфейс некоторого элемента, предоставляются его подэлементом, то элемент-целое делегирует предоставляемый интерфейс своему подэлементу. Если операция или данные входящие в требуемый интерфейс некоторого элемента, используются его подэлементом/подэлементами, то элемент-целое делегирует требуемый интерфейс своему подэлементу/подэлементам.

#### 2.2.2.3 Информация для заинтересованных лиц

Динамические представления дают следующую информацию об архитектуре ПО:

- существующие в архитектуре ПО активные компоненты, их свойства, методы обмена данными между ними и используемые ими общие ресурсы. Используется при проверке соответствия архитектуры ПО требованиям высокого уровня участниками процесса верификации, и в первую очередь, при проверке соответствия архитектуры временным требованиям высокого уровня. Наличие динамических представлений дает возможность оценить, присутствуют ли в архитектуре проблемы, свойственные многозадачному ПО: состояние гонок, взаимоблокировки, инверсия приоритетов;
- распределение требований высокого уровня между активными компонентами. Используется при рассмотрении изменений участниками процесса УКПО. Используется при проверке соответствия архитектуры ПО требованиям высокого уровня участниками процесса верификации. Используется участниками процесса управления проектом для оценки трудоемкости каждого компонента и распределения дальнейшей работы над каждым компонентом между участниками процессов ЖЦ ПО;
- интерфейсы активных компонентов, их окружение. Используется при проведении испытаний интеграции ПО участниками процесса верификации.



### 3 АРХИТЕКТУРА. ПРАВИЛА

#### 3.1 Общие правила

##### 3.1.1 Модульные представления

###### [DES-ARCH-GENERAL-01]

Правило	Модуль <i>должен</i> иметь единственную и четко определенную роль в архитектуре.
Обоснование	«Сфокусированность» модуля на одной цели упрощает архитектуру ПО, что улучшает свойства модифицируемости и верифицируемости архитектуры.
Пояснения	Данное правило требует, чтобы подпрограммы, данные и типы данных модуля, были «сфокусированы» на одной цели, определяемой ролью модуля в архитектуре.
Пример	<p>Примером модуля, соответствующего данному правилу, является модуль, ответственный за работу с некоторым внешним устройством. Все подпрограммы данного модуля «сфокусированы» на одной единственной цели – предоставить остальным модулям сервисы для работы с данным устройством. Модуль предоставляет другим модулям подпрограммы для чтения и записи данных во внешнее устройство и подпрограммы проверки состояния внешнего устройства.</p> <p>Другим примером модуля, соответствующего данному правилу, является модуль, ответственный за работу с некоторым контейнером данных (списком, массивом и т.п.). Все подпрограммы данного модуля «сфокусированы» на одной единственной цели – предоставить остальным модулям сервисы для работы с данным контейнером. Модуль предоставляет другим модулям подпрограммы чтения элемента данных из контейнера, вставки элемента данных в контейнер, удаления элемента данных из контейнера, функцию и подпрограммы проверки состояния контейнера.</p> <p>Примерами модулей, соответствующих данному правилу, могут быть модули, каждый из которых ответственен за вычисление одного скоростно-высотного параметра самолета: первый модуль вычисляет барометрическую высоту, второй – истинную скорость, третий – число Маха. Эти модули могут рассматриваться как подмодули другого модуля. Такой объединяющий модуль также соответствует данному правилу, так как имеет единственную и четко определенную роль – вычисление скоростно-высотных параметров самолета.</p>

###### [DES-ARCH-GENERAL-02]

Правило	Предоставляемый интерфейс модуля <i>не должен</i> содержать детали реализации этого модуля, знания о которых не требуются другим модулям.
Обоснование	Модули зависят от предоставляемых интерфейсов друг друга, так что внесение изменений в предоставляемый интерфейс одного модуля повлечет за собой изменение одного или нескольких других модулей. В процессе проектирования, проектировщик может неоднократно принимать решения об изменении каких-либо деталей реализации модуля. Поэтому, для уменьшения изменений, которые потребуется вносить в спроектированную архитектуру, в предоставляемый интерфейс модуля необходимо помещать подпрограммы, данные и типы данных,

	<p>скрывающие от других модулей детали реализации данного модуля.</p> <p>При правильно спроектированном интерфейсе, изменение деталей реализации модуля не приведет к изменению его предоставляемого интерфейса и таким образом, изменение деталей реализации модуля не потребует внесения изменений и в другие модули.</p>
<b>Пояснения</b>	n/a.
<b>Пример</b>	<p>Примером модуля, соответствующего данному правилу, является модуль, ответственный за работу с некоторым контейнером данных и скрывающий от других модулей способ хранения данных, порядок их хранения и т.п. Другими словами, такой модуль предоставляет абстрактный интерфейс доступа к данному устройству. Если бы такой модуль, открыл контейнер для прямого чтения или записи другим модулям, то другие модули стали бы зависимы от способа хранения данных в этом контейнере и структуры контейнера. При необходимости изменить структуру контейнера, изменения бы пришлось вносить не только в модуль, ответственный за контейнер, но и во все модули его использующие. Все что нужно знать другим модулям – интерфейсные функции записи данных в контейнер и чтения данных из него.</p> <p>Другим примером модуля, соответствующего данному правилу, является модуль, ответственный за чтение/запись данных во внешнее устройство и скрывающий от других модулей протокол доступа к данному устройству, адреса его регистров и т.п. Другими словами, такой модуль предоставляет абстрактный интерфейс доступа к данному устройству. При изменении протокола доступа к внешнему устройству изменения затронут лишь данный модуль и не распространятся на остальные.</p>

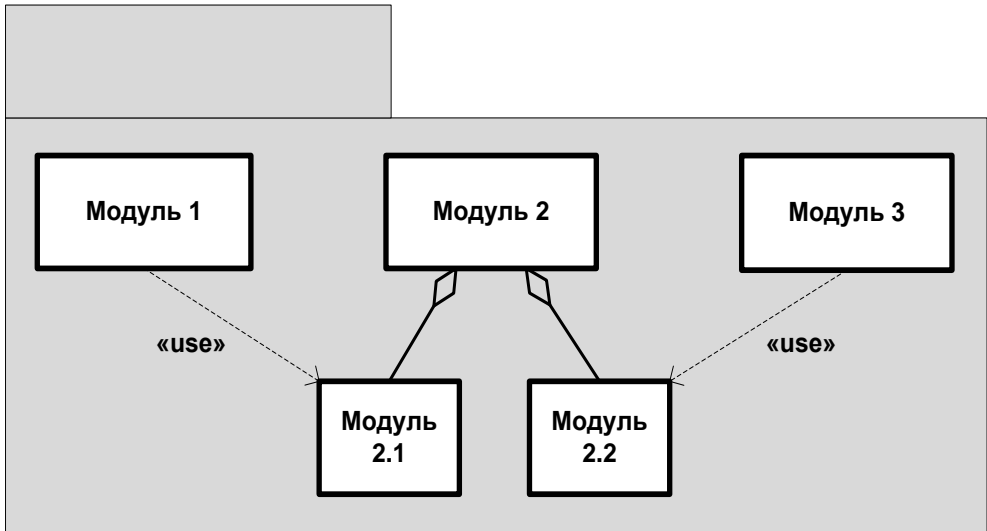
#### [DES-ARCH-GENERAL-03]

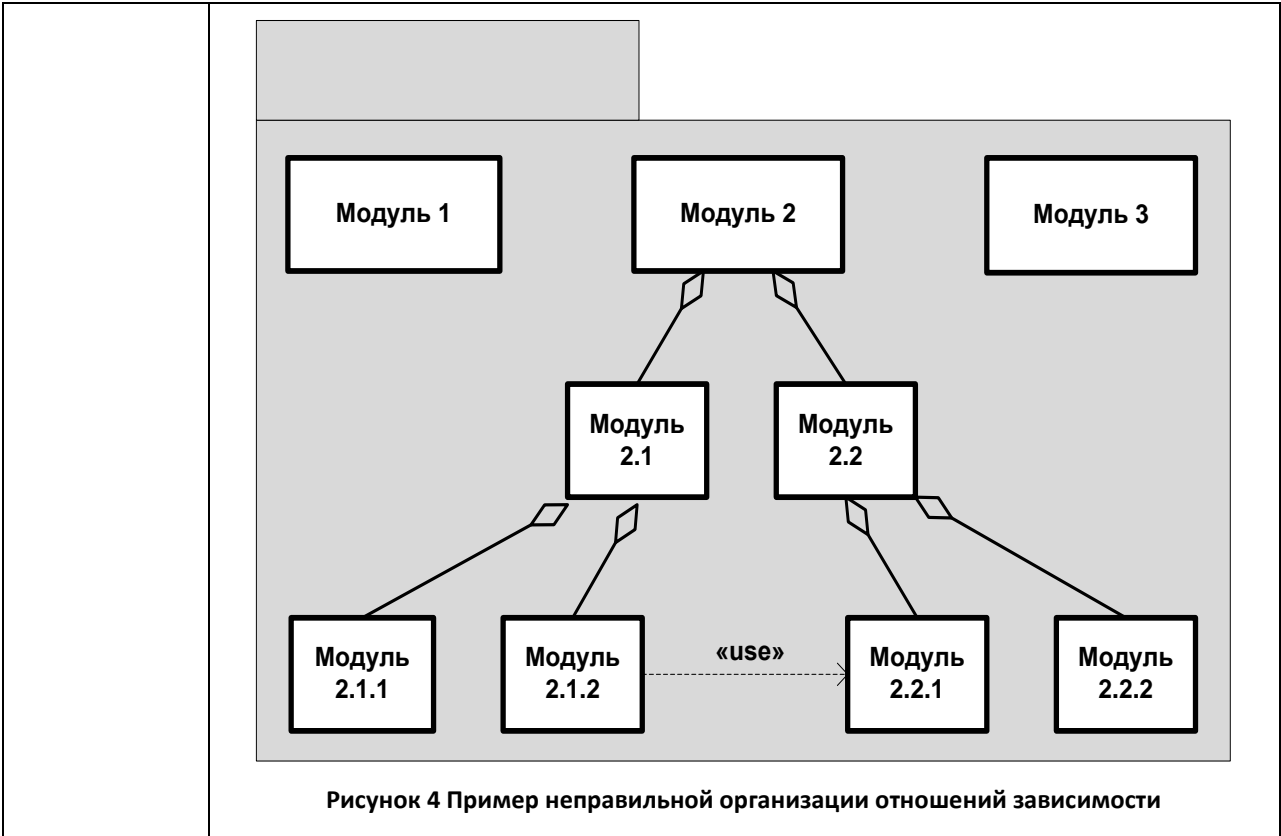
<b>Правило</b>	Модуль <i>должен</i> использовать только документированный интерфейс другого модуля.
<b>Обоснование</b>	Использование недокументированных возможностей других модулей ставит под угрозу надежность работы системы, в случае изменения этого поведения, либо наличия неочевидных побочных эффектов.
<b>Пояснения</b>	n/a.
<b>Пример</b>	n/a.

#### [DES-ARCH-GENERAL-05]

<b>Правило</b>	Параметры функции, входящей в программный интерфейс модуля, изменение которых в данной функции не предусматривается, <i>должны</i> быть определены как константные параметры.
<b>Обоснование</b>	n/a.
<b>Пояснения</b>	n/a.
<b>Пример</b>	n/a.

[DES-ARCH-GENERAL-07]

<b>Правило</b>	<p>Модуль <i>должен</i> зависеть только от модулей, находящиеся с ним на одном уровне декомпозиции и от своих подмодулей.</p> <p>Подмодуль <i>не должен</i> зависеть от модулей, зависимость от которых не разрешена модулю его содержащему.</p>
<b>Обоснование</b>	<p>Ограничение на использование подмодулей улучшает свойство модифицируемости архитектуры: если модуль <b>А</b> зависит только от интерфейса модуля <b>В</b>, но не зависит от его подмодулей, то при изменении состава подмодулей модуля <b>В</b>, их интерфейсов и поведения, не затрагивающем интерфейс самого модуля <b>В</b>, модуль <b>А</b> менять не потребуется.</p>
<b>Пояснения</b>	п/а.
<b>Исключения</b>	п/а.
<b>Пример</b>	<p>На рисунке 3 показан пример неправильной организации отношений зависимости между модулями: модули <b>Модуль 1</b> и <b>Модуль 3</b> используют предоставляемые интерфейсы подмодулей модуля <b>Модуль 2</b> вместо того чтобы использовать предоставляемый интерфейс собственно модуля <b>Модуль 2</b>.</p>  <p>The diagram illustrates a hierarchical structure. At the top level, there are three boxes labeled 'Модуль 1', 'Модуль 2', and 'Модуль 3'. Below 'Модуль 2' are two submodules, 'Модуль 2.1' and 'Модуль 2.2'. Dashed arrows labeled '«use»' point from 'Модуль 1' to 'Модуль 2.1' and from 'Модуль 3' to 'Модуль 2.2'. Solid lines with open diamond heads point from 'Модуль 2' to both 'Модуль 2.1' and 'Модуль 2.2', indicating that 'Модуль 2' contains these submodules. This setup is incorrect because 'Модуль 1' and 'Модуль 3' should depend on 'Модуль 2' directly rather than on its internal submodules.</p> <p><b>Рисунок 3 Пример неправильной организации отношений зависимости</b></p> <p>На рисунке 4 показан еще один пример неправильной организации отношений зависимости между модулями: подмодуль <b>Модуль 2.1.2</b> использует предоставляемые интерфейс подмодуля <b>Модуль 2.2.1</b>, однако использование подмодуля <b>Модуль 2.2.1</b> не разрешено модулю содержащему подмодуль <b>Модуль 2.1.2</b> – модулю <b>Модуль 2.1</b>. Модулю <b>Модуль 2.1.2</b> разрешено использовать лишь интерфейс модуля <b>Модуль 2.2</b>.</p>



3.1.2 Динамические представления

[DES-ARCH-GENERAL-08]

Правило	Каждый компонент <i>должен</i> иметь единственную и четко определенную роль в архитектуре.
Обоснование	«Сфокусированность» активного компонента на одной цели упрощает архитектуру ПО, что улучшает свойства модифицируемости и верифицируемости архитектуры.
Пояснения	п/а.
Исключения	п/а.
Пример	Следующие активные компоненты могут являться примерами, соответствующими данному правилу: <ul style="list-style-type: none"><li>• задача, обрабатывающая одно единственное событие;</li><li>• задача, обрабатывающая все события от одного источника;</li><li>• задача, работающая (прием или выдача данных) с некоторым внешним устройством.</li></ul>

3.2 Правила именования

[DES-ARCH-NAME-01]

<b>Правило</b>	Имя модуля или компонента <i>должно</i> удовлетворять следующим условиям: <ul style="list-style-type: none"> <li>• имя отражает назначение элемента;</li> <li>• имя состоит из одного или нескольких английских слов или их сокращений, чисел, английских аббревиатур;</li> <li>• в имени отсутствуют такие сокращения английских слов, которые могут привести к неоднозначному толкованию смыслового значения имени;</li> <li>• имя начинается со слова или его сокращения;</li> <li>• имя начинается со строчной буквы;</li> <li>• части имени (слова, числа, аббревиатуры) разделяются знаками «_».</li> </ul>
<b>Обоснование</b>	n/a.
<b>Пояснения</b>	n/a.
<b>Исключения</b>	Если при создании ПО используется объектно-ориентированная технология и некоторому модулю соответствует класс, то имя данного модуля формируется по правилам, определенным для имен классов в стандарте на кодирование.
<b>Пример</b>	Примеры имен, удовлетворяющих данному правилу: «built_in_control»

#### [DES-ARCH-NAME-04]

<b>Правило</b>	Имя файла исходного кода <i>должно</i> удовлетворять следующим условиям: <ul style="list-style-type: none"> <li>• имя отражает назначение файла;</li> <li>• длина имени не более 31 символа;</li> <li>• имя состоит из одного или нескольких английских слов или их сокращений, чисел, английских аббревиатур;</li> <li>• в имени отсутствуют такие сокращения английских слов, которые могут привести к неоднозначному толкованию смыслового значения имени;</li> <li>• имя начинается со слова или его сокращения;</li> <li>• имя начинается со строчной буквы;</li> <li>• части имени (слова, числа, аббревиатуры) разделяются знаками «_».</li> </ul>
<b>Обоснование</b>	n/a.
<b>Пояснения</b>	n/a.
<b>Исключения</b>	n/a.
<b>Пример</b>	Примеры имен файлов, удовлетворяющих данному правилу: «fault_log», «nearest_data», «control_fms».

## 4 ТРЕБОВАНИЯ НИЗКОГО УРОВНЯ. ПРАВИЛА

### [DES-REQ-GENERAL-05]

<b>Правило</b>	Каждое требование к модулю <i>должно</i> быть однозначным.
<b>Обоснование</b>	п/а.
<b>Пояснения</b>	Требование является однозначным тогда и только тогда, когда оно допускает единственное толкование.
<b>Исключения</b>	п/а.
<b>Пример</b>	п/а.

### [DES-REQ-GENERAL-06]

<b>Правило</b>	Каждое требование к модулю <i>должно</i> быть неделимым.
<b>Обоснование</b>	п/а.
<b>Пояснения</b>	Требование не является неделимым, если оно может быть разделено очевидным образом на две или более независимых составляющих. Если требование фактически содержит два или несколько независимых утверждений, то его следует представлять в документе соответствующим числом неделимых требований.
<b>Исключения</b>	п/а.
<b>Пример</b>	п/а.

### [DES-REQ-GENERAL-07]

<b>Правило</b>	Каждое требование к модулю <i>должно</i> быть верифицируемым.
<b>Обоснование</b>	п/а.
<b>Пояснения</b>	Требование является верифицируемым тогда и только тогда, когда оно выражено в форме, для которой существует конечная процедура проверки реализации требования в исходном коде. Подразумевается, что такая процедура обязательно включает испытания.
<b>Исключения</b>	п/а.
<b>Пример</b>	п/а.

### [DES-REQ-GENERAL-08]

<b>Правило</b>	Каждое требование к модулю <i>должно</i> иметь ссылки на требования, из которых оно получено.
<b>Обоснование</b>	п/а.
<b>Пояснения</b>	п/а.
<b>Исключения</b>	Производные требования к модулю.

## **5 ПРАВИЛА ОФОРМЛЕНИЯ ОПИСАНИЯ ПРОЕКТА ПО**

Описание проекта ПО оформляется как описание взаимосвязанных архитектурных представлений.

Архитектурные представления описываются «сверху-вниз». Т.е. сначала описываются модули/компоненты верхнего уровня, затем их подмодули/подкомпоненты и т.д. Такой подход позволяет заинтересованному лицу, работающему с описанием проекта, в один момент времени иметь дело с ограниченным количеством модулей/компонентов: сначала с небольшим количеством модулей/компонентов верхнего уровня, потом, для каждого модуля/компонента, с небольшим количеством его подмодулей/подкомпонентов и т.д. Таким образом, сложные задачи анализа и описания архитектуры разбиваются на ряд более простых подзадач.

Описание проекта ПО представляет собой последовательность разделов, каждый из которых включает в себя последовательность текстовых фрагментов и (или) подразделов. Каждый подраздел, как и раздел, также включает в себя последовательность текстовых фрагментов и (или) подразделов. Таким образом, описание проекта ПО имеет иерархическую структуру.

## ПРИЛОЖЕНИЕ А. ШАБЛОН ОПИСАНИЯ МОДУЛЯ

### Общая структура

*СОДЕРЖАНИЕ РАЗДЕЛА: описание отношений (зависимости, часть-целое, наследования) между подмодулями описываемого модуля/модулями верхнего уровня. Описание связей между подмодулями описываемого модуля/модулей верхнего уровня и их окружением (другими модулями)*

### Распределение требований

*СОДЕРЖАНИЕ РАЗДЕЛА: описание распределения требований к описываемому модулю/ПО между его подмодулями/модулями верхнего уровня*

### Описание подмодулей/модулей верхнего уровня

#### Подмодуль/модуль верхнего уровня <имя модуля>

##### Назначение

*СОДЕРЖАНИЕ РАЗДЕЛА: описание роли подмодуля/модуля верхнего уровня в архитектуре*

##### Интерфейс

*СОДЕРЖАНИЕ РАЗДЕЛА: см. Приложение Б*

##### Требования низкого уровня

*СОДЕРЖАНИЕ РАЗДЕЛА: информация о том, какие требования отнесенные к данному подмодулю/модулю верхнего уровня переработаны в требования к подмодулю/модулю верхнего уровня, а какие стали требованиями к данному подмодулю/модулю верхнего уровня без переработки. Формулировки переработанных требований к подмодулю/модулю верхнего уровня*

##### Файлы исходного кода

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление файлов исходного кода, реализующих модуль*

##### Отключенный код

*СОДЕРЖАНИЕ РАЗДЕЛА: описание отключенного кода и описание средств, гарантирующих, что он не будет активизирован в целевом вычислителе*

...

#### Подмодуль/модуль верхнего уровня <имя модуля>

##### Назначение

*СОДЕРЖАНИЕ РАЗДЕЛА: описание роли подмодуля/модуля верхнего уровня в архитектуре*

##### Интерфейс

*СОДЕРЖАНИЕ РАЗДЕЛА: см. Приложение Б*

##### Требования низкого уровня

*СОДЕРЖАНИЕ РАЗДЕЛА: информация о том, какие требования отнесенные к данному подмодулю/модулю верхнего уровня переработаны в требования к подмодулю/модулю верхнего уровня, а какие стали требованиями к данному подмодулю/модулю верхнего уровня без переработки. Формулировки переработанных требований к подмодулю/модулю верхнего уровня*

##### Файлы исходного кода

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление файлов исходного кода, реализующих модуль*

##### Отключенный код

*СОДЕРЖАНИЕ РАЗДЕЛА: описание отключенного кода и описание средств, гарантирующих, что он не будет активизирован в целевом вычислителе*

### Потоки данных и управления

*СОДЕРЖАНИЕ РАЗДЕЛА: описание взаимодействия между подмодулями описываемого модуля/модулями верхнего уровня*



## ПРИЛОЖЕНИЕ Б. ШАБЛОН ОПИСАНИЯ ИНТЕРФЕЙСА МОДУЛЯ

### Собственный интерфейс

#### Предоставляемый интерфейс

##### Подпрограммы

*СОДЕРЖАНИЕ РАЗДЕЛА: краткое описание подпрограмм, входящих в предоставляемый интерфейс модуля, назначения этих подпрограмм, их аргументов, возвращаемых значений, генерируемых этими подпрограммами исключений.*

##### Типы данных

*СОДЕРЖАНИЕ РАЗДЕЛА: краткое описание типов данных, входящих в предоставляемый интерфейс модуля, назначения этих типов*

##### Данные

*СОДЕРЖАНИЕ РАЗДЕЛА: краткое описание данных, входящих в предоставляемый интерфейс модуля, назначения этих данных*

#### Требуемый интерфейс

##### Подпрограммы

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление подпрограмм, требуемых данным модулем, но входящих в предоставляемый интерфейс других модулей*

##### Типы данных

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление типов данных, требуемых данным модулем, но входящих в предоставляемый интерфейс других модулей*

##### Данные

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление данных, требуемых данным модулем, но входящих в предоставляемый интерфейс других модулей*

### Делегированный интерфейс

#### Предоставляемый интерфейс

##### Подпрограммы

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление подпрограмм, определенных в данном модуле, но входящих в предоставляемый интерфейс другого модуля(модуля, содержащего данный модуль)*

##### Данные

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление данных, определенных в данном модуле, но входящих в предоставляемый интерфейс другого модуля(модуля, содержащего данный модуль)*

#### Требуемый интерфейс

##### Подпрограммы

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление подпрограмм, используемых данным модулем, но входящих в требуемый интерфейс другого модуля(модуля, содержащего данный модуль)*

##### Данные

*СОДЕРЖАНИЕ РАЗДЕЛА: перечисление данных, используемых данным модулем, но входящих в требуемый интерфейс другого модуля(модуля, содержащего данный модуль)*

•