Документация менеджера многократно используемых компонентов ostis-систем

Менеджер многократно используемых компонентов ostis-систем
⇒ *ключевой знак**:
- *Реализация менеджера многократно используемых компонентов ostis-систем*
- *Пример структуры хранилища адресов спецификаций компонентов*

⇒ *ключевое понятие**:
- *менеджер многократно используемых компонентов ostis-систем*

⇒ *ключевое знание**:
- *Минимальные функциональные возможности менеджера компонентов*
- *Расширенные функциональные возможности менеджера компонентов*

менеджер многократно используемых компонентов ostis-систем — подсистема ostis-системы, с помощью которой происходит взаимодействие с библиотекой компонентов ostis-систем.

менеджер многократно используемых компонентов ostis-систем
⊂ *встраиваемая ostis-система*
⊂ *платформенно-зависимый многократно используемый компонент ostis-систем*
:= *часто используемый sc-идентификатор**:
[менеджер многократно используемых компонентов]
:= *часто используемый sc-идентификатор**:
[менеджер компонентов]
⇒ *обобщенная декомпозиция**:
{• *база знаний менеджера многократно используемых компонентов ostis-систем*
• *решатель задач менеджера многократно используемых компонентов ostis-систем*
• *интерфейс менеджера многократно используемых компонентов ostis-систем*
}
∋ *Реализация менеджера многократно используемых компонентов ostis-систем*
⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-component-manager]

база знаний менеджера многократно используемых компонентов ostis-систем содержит все те знания, которые необходимы для установки многократно используемого компонента в дочернюю ostis-систему. К таким знаниям относятся знания о спецификации многократно используемых компонентов, методы установки компонентов, знание о библиотеках ostis-систем, с которыми происходит взаимодействие, *Классификация компонентов* и другие. ***решатель задач менеджера многократно используемых компонентов ostis-систем*** взаимодействует с *библиотекой ostis-систем* и позволяет установить и интегрировать многократно используемые компоненты в дочернюю ostis-систему, также выполнять поиск, обновление, публикацию, удаление компонентов и другие операции с ними. ***интерфейс менеджера многократно используемых компонентов ostis-систем*** обеспечивает удобное для пользователя и других систем использование менеджера компонентов.
решатель задач менеджера многократно используемых компонентов ostis-систем как минимум должен обеспечивать следующие функциональные возможности:

менеджер многократно используемых компонентов ostis-систем
⇒ *минимальные функциональные возможности**:
Минимальные функциональные возможности менеджера компонентов
= {• [**Поиск многократно используемых компонентов ostis-систем.** Множество возможных критериев поиска соответствует спецификации многократно используемых компонентов. Такими критериями могут быть классы компонента, его авторы, идентификатор, фрагмент примечания, назначение, принадлежность какой-либо предметной области, вид знаний компонента и другие.]
• [**Установка многократно используемого компонента ostis-систем.** Установка многократно используемого компонента происходит вне зависимости от типологии, способа установки и местоположения компонента. Необходимое условие для возможности установки многократно используемого компонента — наличие ***спецификации многократно используемого компонента ostis-систем***. Перед установкой многократно используемого компонента в дочернюю систему необходимо установить все зависимые компоненты. Также для платформенно-зависимых компонентов может

быть необходимо установить иные зависимости, которые не являются компонентами какой-либо библиотеки ostis-систем. После успешной установки компонента в базе знаний дочерней системы генерируется информационная конструкция, обозначающая факт установки компонента в систему с помощью отношения *установленные компоненты**.]

- [**Добавление и удаление отслеживаемых менеджером компонентов библиотек.** Менеджер компонентов содержит информацию о множестве источников для установки компонентов, перечень которых можно дополнять. По умолчанию менеджер компонентов отслеживает *Библиотеку Метасистемы OSTIS*, однако можно создавать и добавлять дополнительные библиотеки ostis-систем.]

}

Исходя из указанных минимальных функциональных возможностей *решатель задач менеджера многократно используемых компонентов ostis-систем* представляет собой следующую иерархию абстрактных sc-агентов:

решатель задач менеджера многократно используемых компонентов ostis-систем
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент поиска многократно используемых компонентов ostis-систем*
⇒ *реализация**:
Агент поиска компонентов
• *Абстрактный sc-агент установки многократно используемых компонентов ostis-систем*
⇒ *реализация**:
Агент установки компонентов
• *Абстрактный sc-агент управления отслеживаемых менеджером компонентов библиотек*
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент добавления отслеживаемой менеджером компонентов библиотеки*
• *Абстрактный sc-агент удаления отслеживаемой менеджером компонентов библиотеки*
}
}
⊃ *Агент установки спецификаций многократно используемых компонентов*

Используя минимальные функциональные возможности менеджер компонентов может установить компоненты, которые будут расширять его же функционал. Компоненты, реализующие расширенный функционал менеджера компонентов являются частью *Библиотеки Метасистемы OSTIS*. К расширенному функционалу относится:

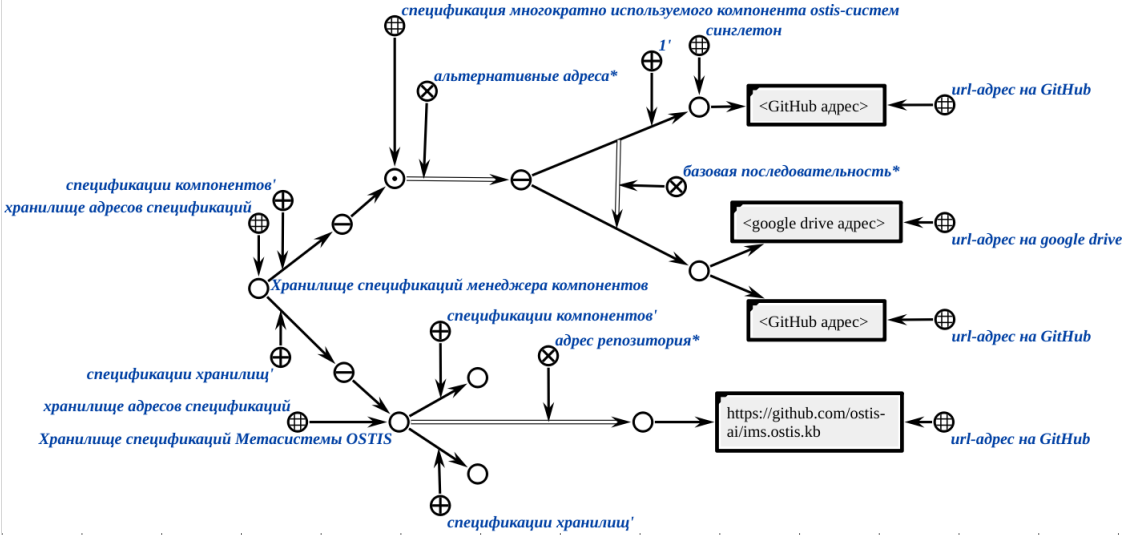
менеджер многократно используемых компонентов ostis-систем
⇒ *расширенные функциональные возможности**:
Расширенные функциональные возможности менеджера компонентов
= {• [**Спецификация** многократно используемого компонента ostis-систем. Менеджер компонентов позволяет специфицировать компоненты, входящие в состав библиотеки ostis-систем, а также специфицировать новые создаваемые компоненты, которые будут публиковаться в библиотеку ostis-систем. При этом спецификация может происходить как автоматически, так и вручную. Например, менеджер компонентов может обновить спецификацию используемого компонента в соответствии с тем, в какие новые ostis-системы его установили, обновить спецификацию авторства компонента при его редактировании в библиотеке ostis-систем, спецификацию ошибок, выявленных в процессе эксплуатации компонента и так далее.]
• [**Формирование** многократно используемого компонента ostis-систем по шаблону с заданными параметрами. При установке шаблона многократно используемого компонента ostis-систем менеджер компонентов позволяет сформировать по нему конкретный компонент. Для этого пользователю предлагается определить значения всех sc-переменных в шаблоне для формирования конкретного компонента из некоторой предметной области. Например, для формирования многократно используемого компонента баз знаний, представляющего собой семантическую окрестность некоторого отношения, нужно определить значения всех переменных, кроме переменной, являющейся ключевым sc-элементом данной структуры.]

- **[Публикация** многократно используемого компонента ostis-систем в библиотеку ostis-систем. При публикации компонента в библиотеку ostis-систем происходит верификация на основе спецификации компонента. Публикация компонента может сопровождаться сборкой неатомарного компонента из существующих атомарных. Также существует возможность обновления версии опубликованного компонента сообществом его разработчиков.**]**
- **[Обновление** установленного многократно используемого компонента ostis-систем.**]**
- **[Удаление** установленного многократно используемого компонента. Как и в случае установки после удаления многократно используемого компонента из ostis-системы в базе знаний системы устанавливается факт удаления компонента. Эта информация является важной частью истории эксплуатации ostis-системы.**]**
- **[Редактирование** многократно используемого компонента в библиотеке ostis-систем.**]**
- **[Сравнение** многократно используемых компонентов ostis-систем.**]**

}

Для того, чтобы создать новую ostis-систему "с нуля", используя *ostis-платформу*, необходимо установить некоторый *Программный вариант реализации ostis-платформы* с помощью сторонних средств. Такими средствами могут быть (1) хранилища исходного кода платформы, например, облачные хранилища, такие как GitHub репозиторий, с соответствующим набором инструкций по установке платформы или (2) средства установки заранее скомпилированной программной реализации платформы, например, средство установки программного обеспечения apt. Далее установка многократно используемых компонентов в ostis-систему (независимо от типа компонентов) осуществляется с помощью менеджера компонентов. При установке платформенно-зависимых компонентов менеджер компонентов должен управлять соответствующими средствами сборки таких компонентов (CMake, Ninja, npm, grunt и другие). Компонент находится в некотором хранилище — (1) *библиотеке компонентов ostis-систем* или (2) в виде файлов в некотором облачном хранилище. В случае, когда компонент хранится в библиотеке, для его установки менеджер компонентов копирует все sc-элементы, которые представляют собой компонент, в дочернюю ostis-систему. В случае, когда компонент хранится в виде файлов в облачном хранилище, менеджер компонентов скачивает файлы компонента и устанавливает их в соответствии со спецификацией. Адреса хранилищ спецификаций компонентов должны храниться в базе знаний менеджера компонентов, чтобы иметь доступ к спецификациям компонентов для их последующего использования (поиска, установки и так далее). На рисунке *1* приведен пример фрагмента базы знаний менеджера компонентов, который описывает то, где хранятся спецификации компонентов, доступных для установки. Такое хранилище представляет собой множество, состоящее из двух множеств: (1) множество адресов спецификаций компонентов и (2) множество адресов спецификаций других хранилищ. Таким образом, образуется древовидная структура в соответствии с иерархией материнских ostis-систем и соответствующих им библиотек.

Рис. 1: SCg-текст. Пример структуры хранилища адресов спецификаций компонентов



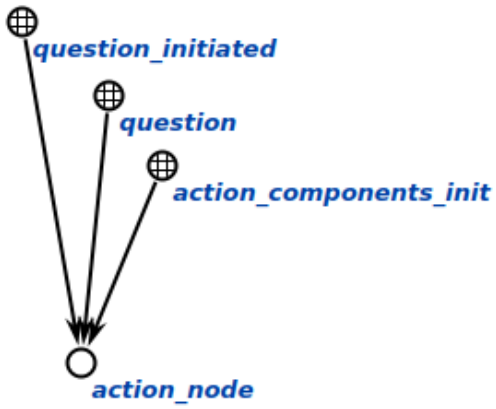
Если указать адрес корня дерева хранилищ адресов спецификаций, то менеджер компонентов получает доступ ко всем спецификациям дочерних хранилищ. При обработке такого дерева спецификаций менеджер компонентов погружает в sc-память спецификации компонентов, доступных для установки, однако не сами эти компоненты.

менеджер многократно используемых компонентов ostis-систем является необязательной подсистемой *ostis-платформы*. Однако система, имеющая менеджер компонентов, может устанавливать компоненты не только сама в себя, но и в другие системы при наличии доступа. Таким образом, одна система может заменить *ostis-платформу* другой системы, оставив при этом *sc-модель кибернетической системы*. Таким же образом некоторая ostis-система может порождать другие ostis-системы, используя компонентный подход.
Включение компонента в *дочернюю ostis-систему* в общем случае состоит из следующих этапов:

- поиск подходящего компонента во множестве доступных библиотек;
- выделение компонента в виде, удобном для транспортировки в дочернюю ostis-систему с указанием версии и модификации при необходимости (например, выбор доступного хранилища компонента, выбор оптимального варианта реализации компонента с учетом состава дочерней системы);
- установка многократно используемого компонента и его зависимостей (с указанием версии и модификации при необходимости);
- интеграция компонента в дочернюю систему;
- поиск и устранение ошибок и противоречий в дочерней системе.

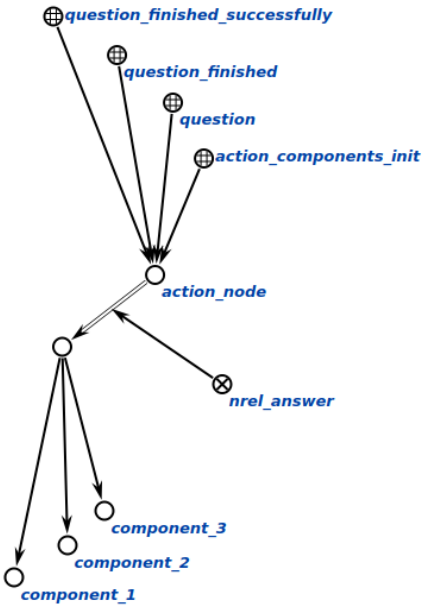
Данный процесс с точки зрения пользователя не зависит от типа компонента и особенностей его реализации.

Агент установки спецификаций многократно используемых компонентов
:= [sc-агент установки спецификаций многократно используемых компонентов]
⇒ *команда, вызывающая агент**:
components init
⇒ *задача**:
⟨• [Отыскать все спецификаци спецификаций многократно используемых компонентов в базе знаний.]
• [Скачать спецификаций на устройство.]
⟩
⇒ *пример входной конструкции**:
[



]
⇒ *аргументы агента**:
{• *пустое множество*
}
⇒ *ответ агента**:
ответ агента установки спецификаций многократно используемых компонентов
⇒ *примечание**:
[В результате выполнения агентом поискового действия на устройстве появятся scs-файлы со спецификациями, скаченными по сети. Данные файлы сразу транслируются в sc-память.]
⇒ *пример выходной конструкции**:

[



]

Агент установки компонентов

:= [sc-агент установки компонентов]

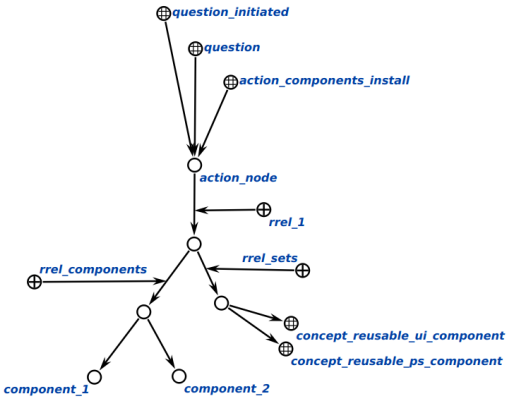
⇒ *команда, вызывающая агент**:
components install

⇒ *задача**:

⟨• [Поиск спецификации компонента в базе знаний по заданному критерию, если он есть, иначе поиск всех спецификаций компонентов.]
• [Скачивание и установка (опционально) на устройство.]
⟩

⇒ *пример входной конструкции**:

[



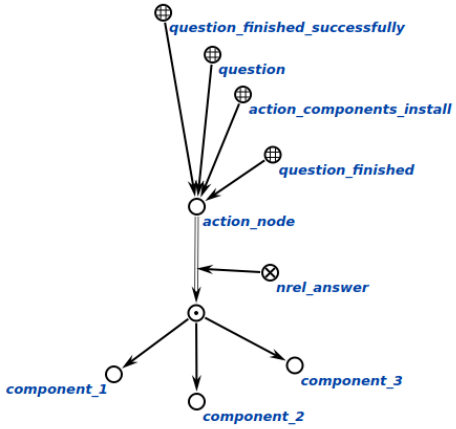
]

⇒ *аргументы агента**:

{• *множество компонентов*
⇒ *примечание**:
[В этом случае отыскиваются все компоненты, принадлежащие указанному множеству, и скачиваются.]
• *множество множеств компонентов*
⇒ *примечание**:
[В этом случае отыскиваются все компоненты, принадлежащие указанным множествам множества, и скачиваются.]
}

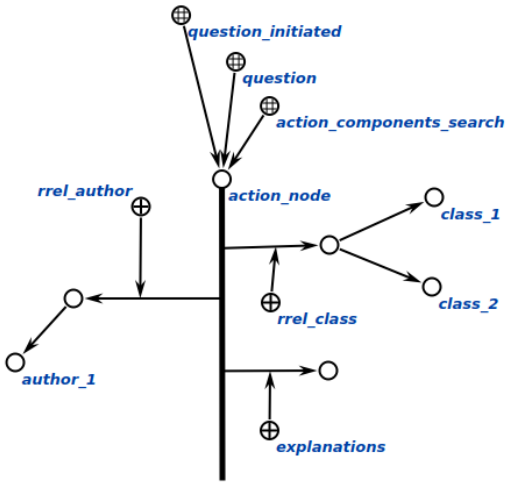
⇒ *ответ агента**:

ответ агента установки компонентов
⇒ *примечание**:
[В результате выполнения агентом действия по установке компонентов на устройстве появятся компоненты в соответствующих папках со спецификациями, и данные компоненты будут установлены, если это необходимо.]
⇒ *пример выходной конструкции**:
[



]

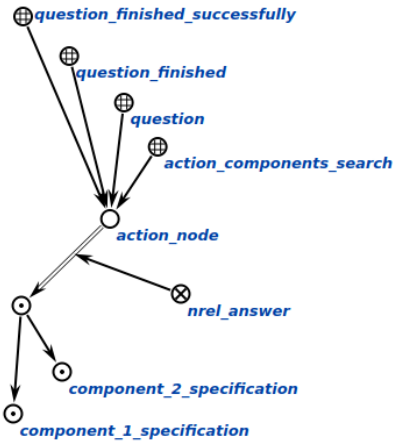
Агент поиска спецификаций компонентов
:= [sc-агент поиска спецификаций компонентов]
⇒ *команда, вызывающая агент**:
components search
⇒ *задача**:
⟨• [Отыскать спецификации компонентов в базе знаний по заданным критериям.]
⟩
⇒ *пример входной конструкции**:
[



]
⇒ *аргументы агента**:
{• *authors*
⇒ *примечание**:
[В этом случае отыскивается спецификация компонента по автору, который создал компонент.]
• *classes*
⇒ *примечание**:
[В этом случае отыскивается спецификация компонента по классу, которому принадлежит компонент.]

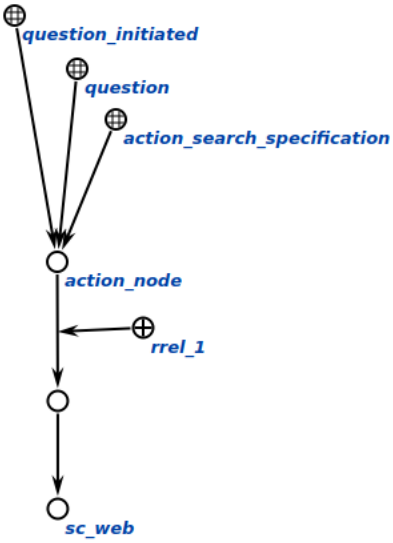
- *explanations*
  ⇒ *примечание**:
  [В этом случае отыскивается спецификация компонента по его описанию (примечанию), которое в общем случае может являться подстрокой реального примечания компонента, хранящегося в базе знаний.]
- *пустое множество*
  ⇒ *примечание**:
  [В этом случае ничего не будет найдено.]

}
⇒ *ответ агента**:
ответ агента поиска компонентов
⇒ *примечание**:
[В результате выполнения агентом поискового действия сформируется ответ со множеством найденных компонентов.]
⇒ *пример выходной конструкции**:
[



]

Агент поиска спецификации компонента по компоненту
:= [sc-агент поиска спецификации компонента по компоненту]
⇒ *задача**:
⟨• [Отыскать спецификацию компонента в базе знаний по заданному компоненту.]
⟩
⇒ *пример входной конструкции**:
[



]

⇒ *аргументы агента**:

⟨• *component*

⇒ *примечание**:

[Узел компонента, для которого необходимо найти спецификацию.]

⟩

⇒ *ответ агента**:

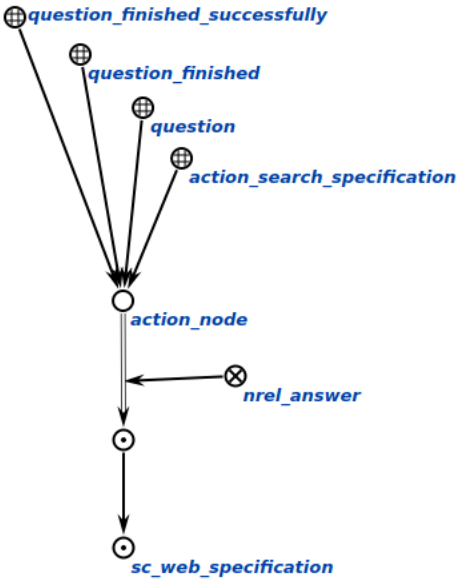
ответ агента поиска спецификации компонента по компоненту

⇒ *примечание**:

[В результате выполнения агентом поискового действия сформируется ответ с найденной спецификацией.]

⇒ *пример выходной конструкции**:

[



]