«Модельно-ориентированная системная и программная инженерия (MBSE)» как базовый курс подготовки ИТ-специалистов (презентация одноименного учебника)

В.А. Сухомлин

Проф. МГУ им. Ломоносова sukhomlin@mail.ru

О системной инженерии (SE)

- Системы естественной и искусственной природы служат строительными блоками современного мира и бытия человека
- Поэтому изучение научных и прикладных аспектов, связанных с системами, а также процессами, методами и средствами, необходимыми для их создания и использования, имеют решающее значение для функционирования современного мира
- Значительную часть систем составляют системы, частично или полностью созданные человеком. Такие системы называются **инженерными**
- Научно-прикладная дисциплина, предназначенная для изучения инженерных систем, а также процессов, методов, средств их создания и использования называется **системной инженерией** (**system engineering – SE**)
- Область SE является многодисциплинарной и характеризуется быстрыми темпами развития, интеграцией с другими технологиями, в частности, такими, как: онтологический инжиниринг и искусственный интеллект
- Международный совет по системной инженерии (INCOSE) определяет **системную инженерию** (SE) как междисциплинарный подход и средства, позволяющие создавать успешные системы

О системной инженерии (SE)

- Как самостоятельная дисциплина системная инженерия сложилась в 1960-х годах в результате опыта реализации программ по созданию сложных космических и ракетных систем, строительству энергетических комплексов, крупных промышленных объектов
- На основе этого опыта сформировались основные принципы и методы, а затем и стандарты, системной инженерии, такие как анализ требований, анализ функциональности систем, архитектурное проектирование, верификация, испытания, анализ качества, обеспечение информационной безопасности систем и т. д.
- Также были разработаны первые математические модели и инструменты для анализа и оптимизации систем
- В SE основное внимание уделяется целостному и одновременному пониманию потребностей заинтересованных сторон (в частности, заказчиков), изучению возможностей систем и требований к системам, их синтезу, проверке, испытаниям и разработке решений при рассмотрении проблемы в целом, от исследования концепции систем до их утилизации

О системной инженерии (SE)

Системную инженерию, которая сложилась до конца XX столетия, часто называют традиционной или **системной инженерией, основанной на документах (Document-Based Systems Engineering, или DBSE),** ввиду того, что основными артефактами, сопровождающими систему на протяжении ее жизни, являлись документы на бумажных и/или машинных носителях.

С конца 1990-х специалисты по системному проектированию стали широко применять методы, основанные на моделях, используемых для облегчения коммуникации, управления сложностью дизайна, улучшения качества продукта, улучшения сбора и повторного использования знаний. Такой подход получил название **модельно-ориентированной системной инженерии (Model-Based Systems Engineering, или MBSE).**

MBSE определяется как формализованное приложение графического моделирования с точными семантическими определениями для анализа операций, определения требований, разработки системного дизайна, верификации и испытаний, начиная с концептуальной фазы и продолжая на более поздних стадиях/этапах жизненного цикла [3].

О системной инженерии (SE)

Суть MBSE заключается в создании машиночитаемых моделей, представляющих все аспекты системы и поддерживающих все действия по проектированию, разработке, производству и эксплуатации системы на протяжении всего ее жизненного цикла

- Эти цифровые модели образуют **цифровой двойник** (Digital Twin — DT) целевой системы, основанный на общих схемах данных, формирующих цифровой поток, который объединял бы все заинтересованные стороны, участвующие в создании или приобретении новых систем
- Стратегия цифрового инжиниринга направлена на повышение эффективности и улучшение качества всей деятельности по созданию и приобретению новых сложных систем
- MBSE предоставляет возможность консолидировать информацию в доступном централизованном источнике, обеспечивая частичную или полную автоматизацию многих процессов системного проектирования и облегчая интерактивное представление компонентов и поведения систем

Область применения системной инженерии

- Область деятельности, связанная с созданием и использованием инженерных систем, чрезвычайно многообразна и сложна
- В связи с чем необходимо уточнить сферу применения именно SE и прежде всего определить взаимосвязи SE с такими дисциплинами как:

- **System Implementation -** внедрение/реализация систем и
- **Project/systems Management -** управление проектами/системами ()

- В SEBoK область применения системной инженерии в жизненном цикле систем иллюстрируется с помощью диаграммы Венна (Рис. 1), показывающей взаимосвязь между системной инженерией, реализацией систем и управлением проектами/системами
- В последние годы наметилась тенденция более тесной интеграции системной инженерии с программной инженерией, что отражено в международном стандарте **ISO/IEC/IEEE 24641:2023-2023 — «Системная и программная инженерия. Методы и инструменты для модельно-ориентированной системной и программной инженерии»**, в котором представлены фундаментальные методологические решения в этой области

Область применения методов системной инженерии

Рисунок 1. Взаимосвязь трех дисциплин - **Системной инженерии**, **реализации систем** (system implementation), и **Управления проектами/системами** (project/systems management): https://sebokwiki.org/d/images/sebokwiki-farm!d/1/1a/Scope_BoundariesSE_PM_SM.png .

Книга содержит 13 глав и 5 приложений

В.А. СУХОМЛИН, В.Ю. РОМАНОВ, Д.А. ГАПАНОВИЧ «ВВЕДЕНИЕ В МОДЕЛЬНО-ОРИЕНТИРОВАННУЮ СИСТЕМНУЮ И ПРОГРАММНУЮ ИНЖЕНЕРИЮ (MBSE)»

Содержание глав по темам следующее:

- **Глава 1. Введение в SE. Основные понятия**
- **Глава 2. Модели жизненного цикла систем**
- **Глава 3. Модели итеративного процесса разработки программного обеспечения**
- **Глава 4. Характерные черты и возможности языка UML**
- **Глава 5. Язык моделирования SysML и его применение для разработки моделей систем**
- **Глава 6. Инженерия требований**
- **Глава 7. MBSE — системная инженерия на основе моделей**
- **Глава 8. Описание архитектуры системы**
- **Глава 9. Цифровые двойники**
- **Глава 10. Система стандартов SE. Процессные стандарты**
- **Глава 11. Эталонная модель модельно-ориентированной системной и программной инженерии (MBSSE) и ее связь с процессными стандартами системной инженерии**
- **Глава 12. Интеграция системы и программного обеспечения. Методические аспекты**
- **Глава 13. Математические основы системной инженерии**

Глава 1. Введение в SE. Основные понятия

1.1. История и настоящее системной инженерии

1.2. Определение и область применения

1.3. Пользователи системной инженерии, область деятельности системных инженеров

1.4. Методологические основы системной инженерии. Системный подход и системное мышление

1.5. Определение и классификация систем: концептуальная модель понятия «система»

1.6. Жизненный цикл систем и процессы жизненного цикла

1.7. Итеративность стадий и процессов жизненного цикла

Методологические основы SE

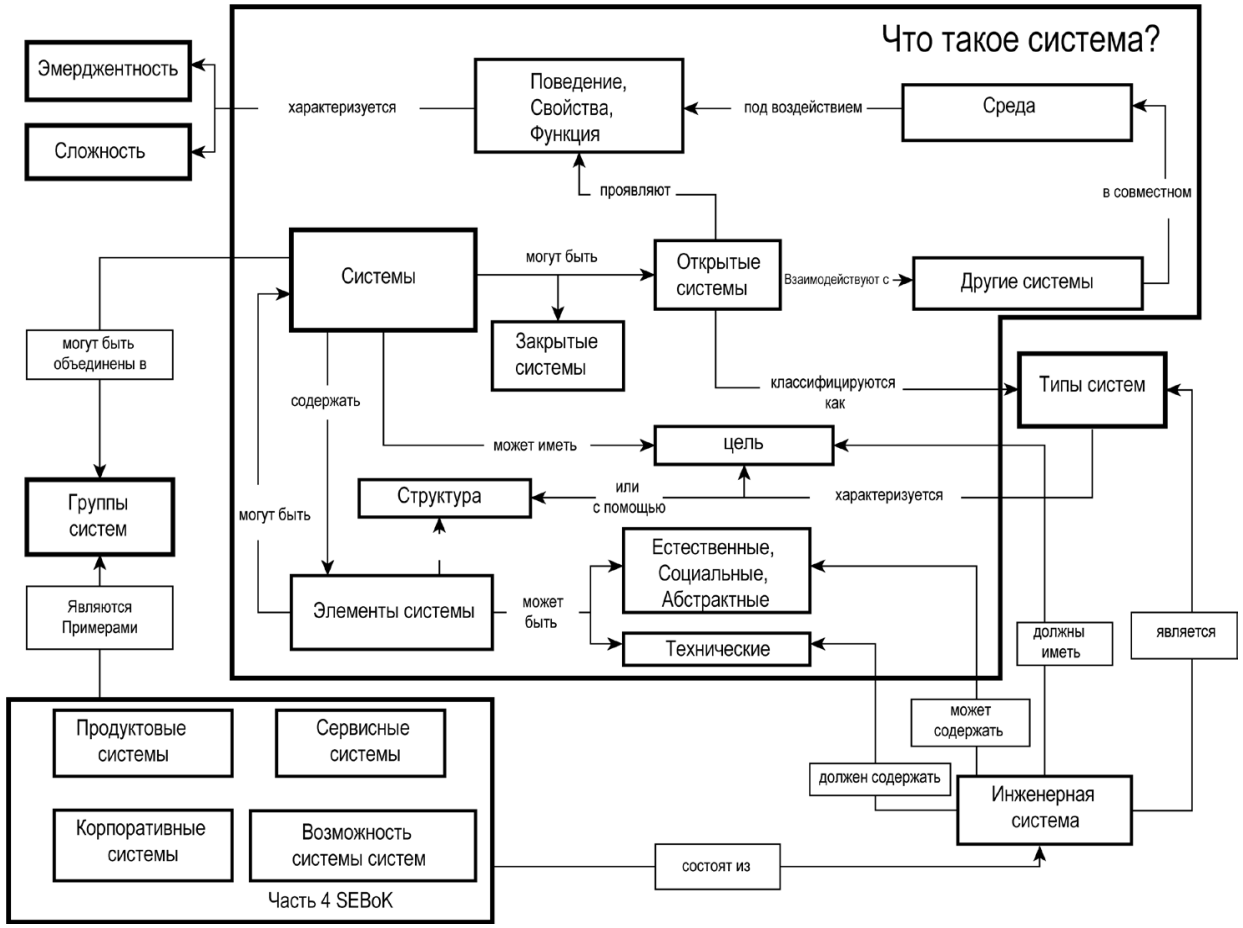
- **системный подход и главные принципами системного подхода**

- **системное мышление и принципы системного мышления** - мыслительная деятельность, построенная на использовании принципов системного подхода,

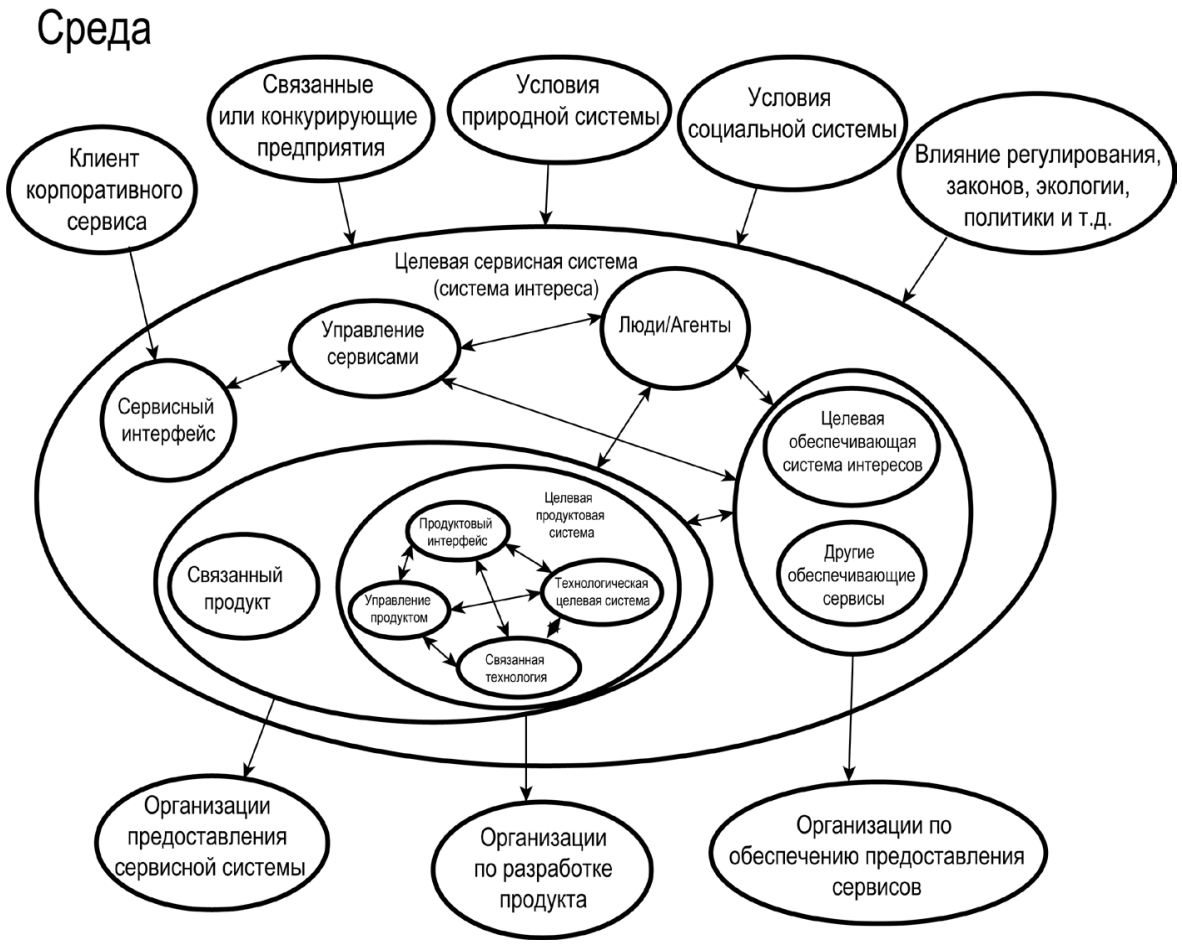
- **принципы системной инженерии** (SEPAT (The INCOSE Systems Engineering Principles Action Team)

Глава 1. Введение в SE. Основные понятия

Концептуальная модель. Определения и классификация систем



Определения системного контекста



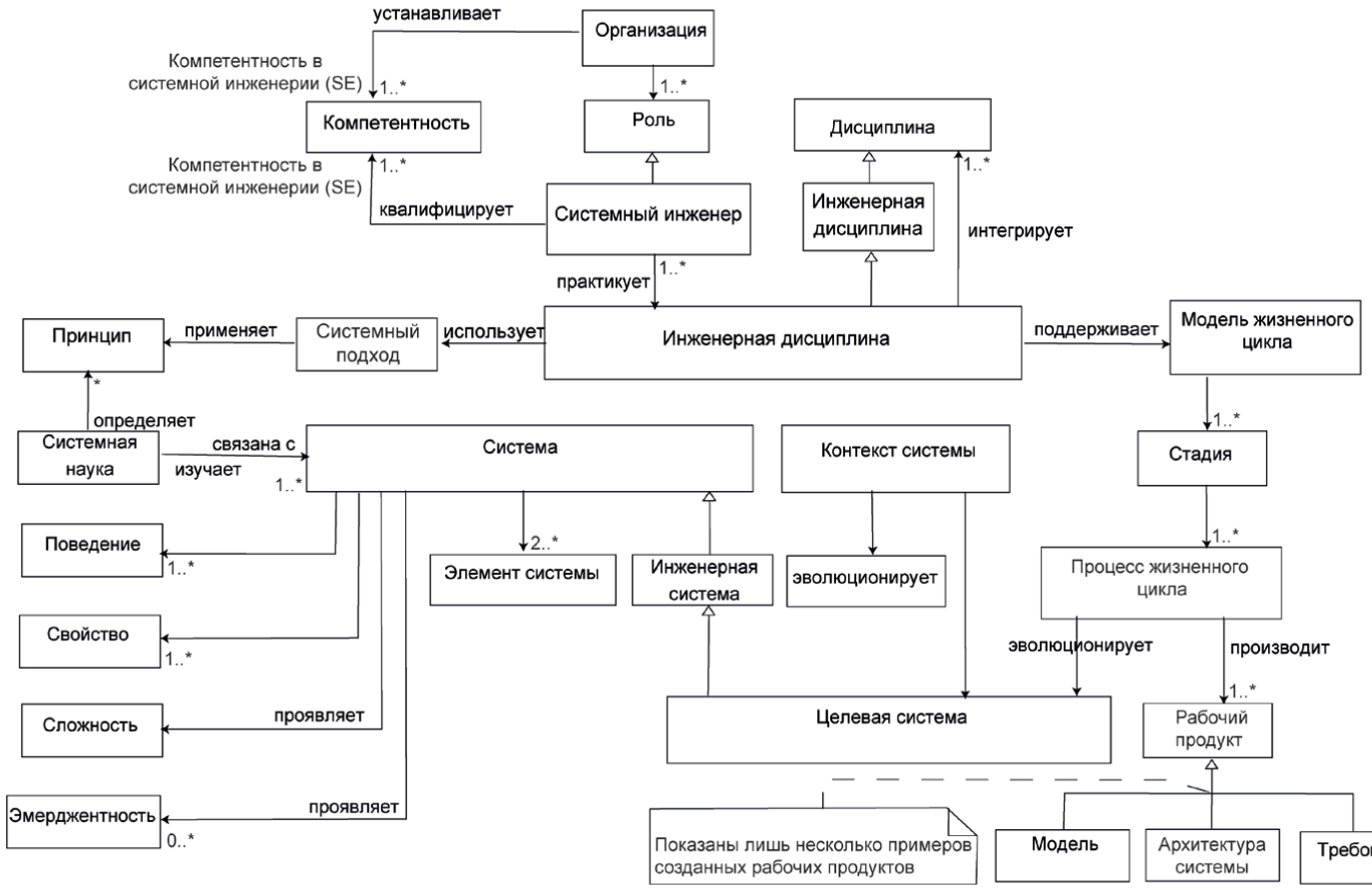
Классификация SOI

Рассмотрен общий случая целевой системы (SoI), которая на протяжении жизненного цикла взаимодействует с системами следующих видов:

- **Технологически ориентированная продуктовая система** SoI, встроенная в один или несколько интегрированных продуктов
- **Интегрированная мультитехнологическая продуктовая система** SoI, используемая непосредственно для предоставления услуги
- **Обеспечивающая сервисная система** SoI, поддерживающая несколько систем обслуживания
- **Сервисная система SoI**, созданная и поддерживаемая для непосредственного предоставления возможностей
- **Система предприятия** (Enterprise Systems)
- **Система систем (System of systems - SOS)** - термин SoS является дополнением к общей концепции системной иерархии, применимой ко всем системам. Поскольку технология, позволяющая интегрировать независимые системы, концепция SOS становится общим аспектом многих жизненных циклов SE

Глава 1. Введение в SE. Основные понятия

Концептуальная модель. Определения жизненного цикла систем



Итерации через обратную связь

Итеративность жизненного цикла систем

Понятия концепции жизненного цикла

Системный инженер — это роль в Организации, которая практикует инженерную дисциплину системной инженерии (SE) и имеет квалификацию набора компетенций SE

Системная инженерия объединяет другие дисциплины для поддержки модели жизненного цикла

Модель жизненного цикла состоит из **этапов жизненного цикла**, которые обычно включают стадии концепции, реализации, производства, поддержки, использования и вывода из эксплуатации (не показаны).

Каждая стадия жизненного цикла относится к **процессам жизненного цикла**, которые производят различные виды **рабочих продуктов**

Процессы жизненного цикла развивают **систему интереса** (SoI)

Существует множество **видов систем**, включая **природные системы**, **социальные системы** и **технологические системы** (не показаны)

Системы, созданные людьми и для людей, называются **инженерными системами**

Инженерная система, чей жизненный цикл находится в стадии рассмотрения, называется **целевой системой** или **система интереса** (SoI)

Общая модель жизненного цикла системы

Рисунок 10. **Обобщенная модель жизненного цикла SoI. -** https://sebokwiki.org/d/index.php?title=File%3AFig_1_A_generic_life_cycle_KF.png [SEBoK Original]

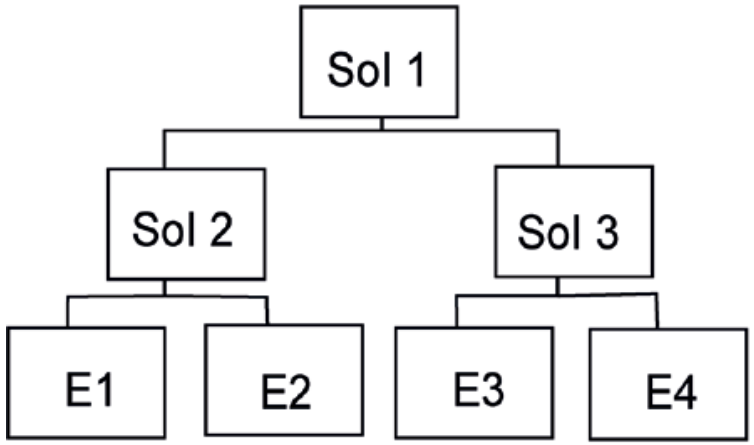
Стадия определения концепции начинается с решения вопроса об инвестировании создания некоторой системы (SoI). В частности, исследуется: целесообразность, возможность разработки, преимущество создания SoI, стоимость затрат на жизненный цикл SoI

Стадия определения системы включает разработку системных архитектур; определение и согласование уровней системных требований; разработка планов жизненного цикла на системном уровне и выполнение системного анализа, чтобы проиллюстрировать совместимость и осуществимость итогового определения системы

Стадия реализации системы начинается есть уверенность, что создание SoI оправдает выделение ресурсов, необходимых для разработки и поддержания начальной операционной способности (**IOC**) или разработки полной эксплуатационной способности (**FOC**). Активности этапа включают: создание элементов системы, проверку и испытание элементов, их интеграцию и подготовку к производству

Последующими этапами жизненного цикла являются **стадии: Производства системы** (**Production**), **Поддержки** (**Support**), **Использования** (**Utilization**) с последующим **выводом из эксплуатации** (**Retirement**)

Общая парадигма SE



SOI - Целевая система
E - Элемент системы

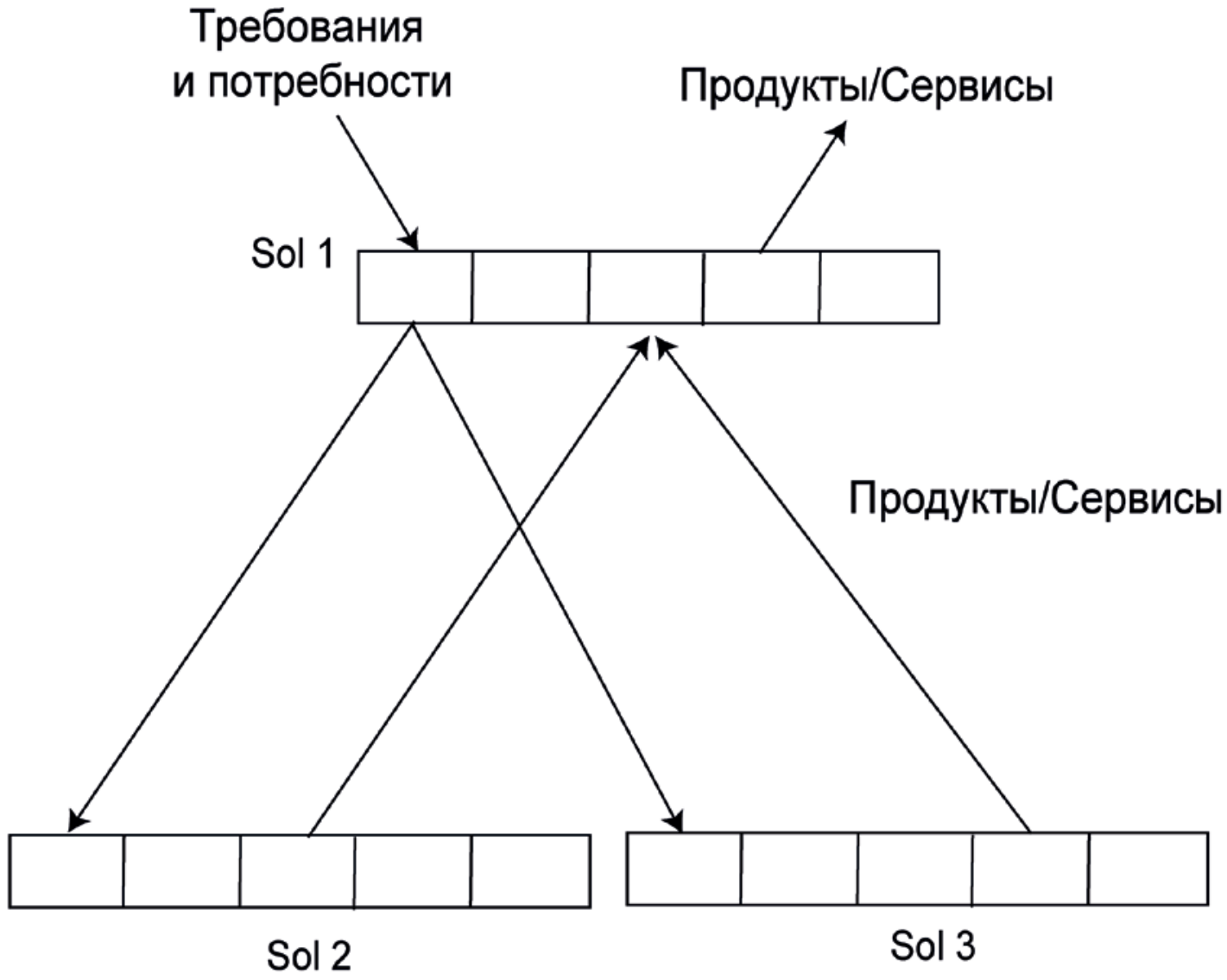
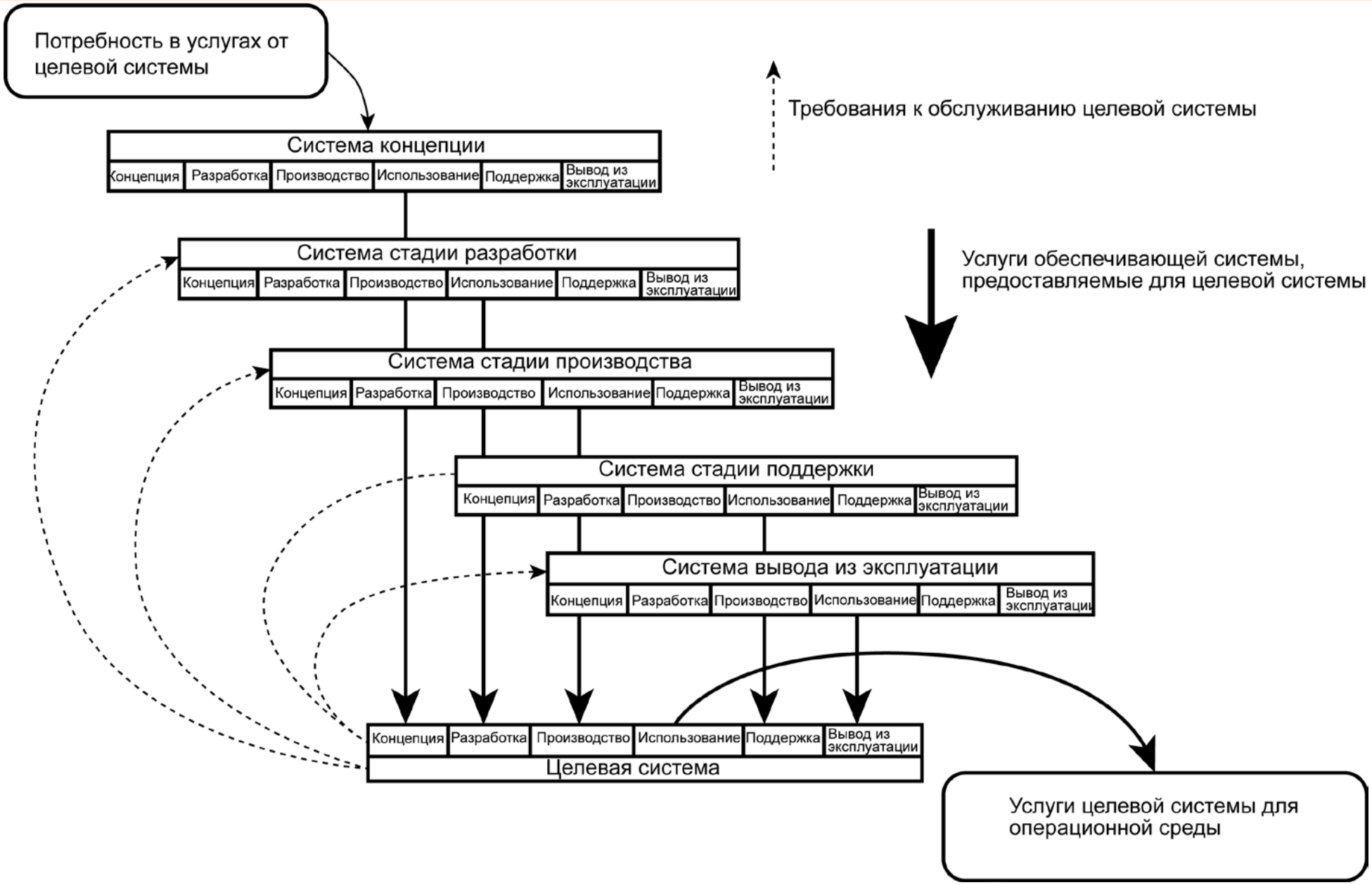


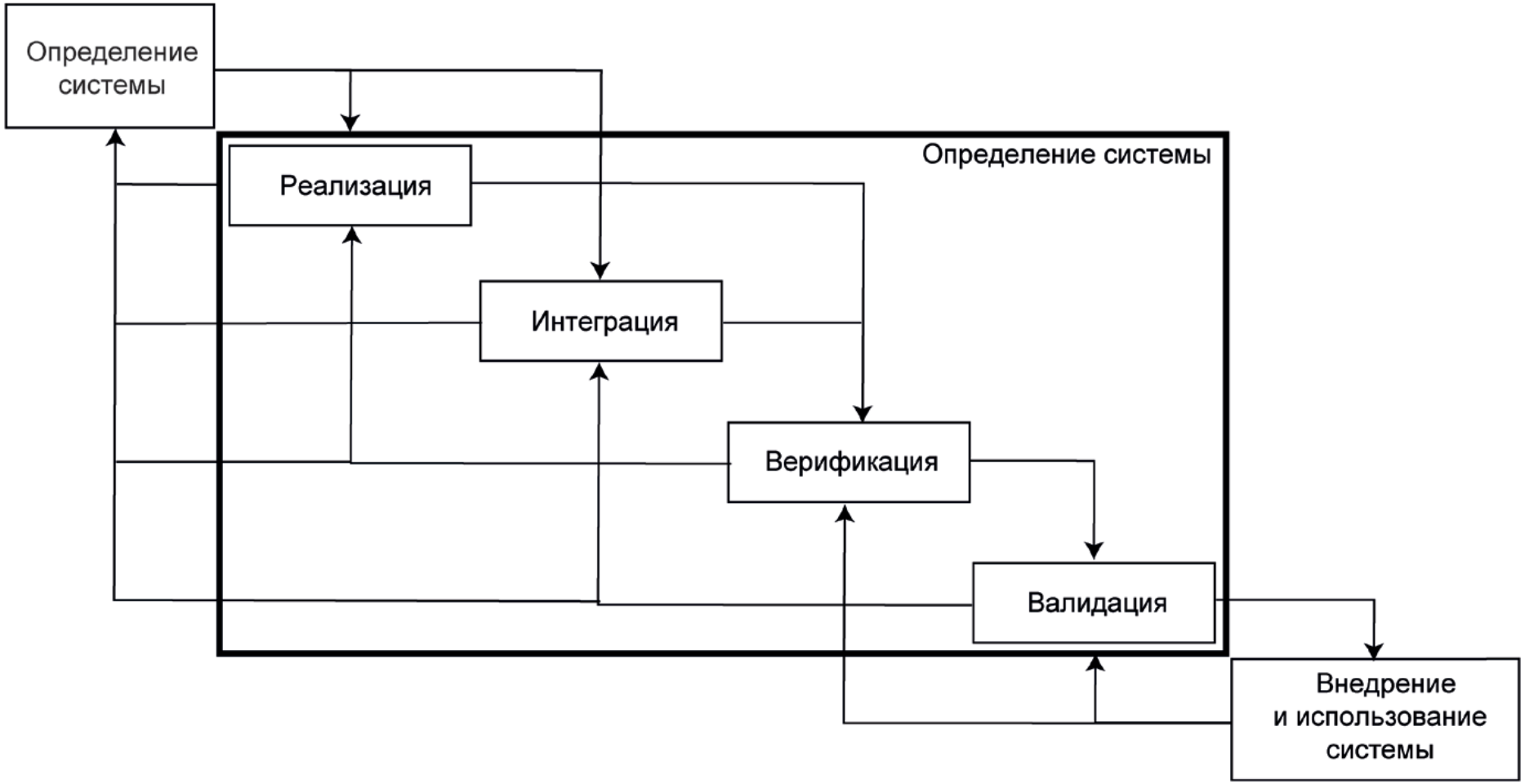
Рисунок 9. Общая парадигма SE - https://sebokwiki.org/d/images/sebokwiki-farm!d/6/6d/062211_BL_Paradigm.png

Глава 1. Введение в SE. Основные понятия



Глава 1. Введение в SE. Основные понятия

Модель жизненного цикла систем



Модель роли системного инженера

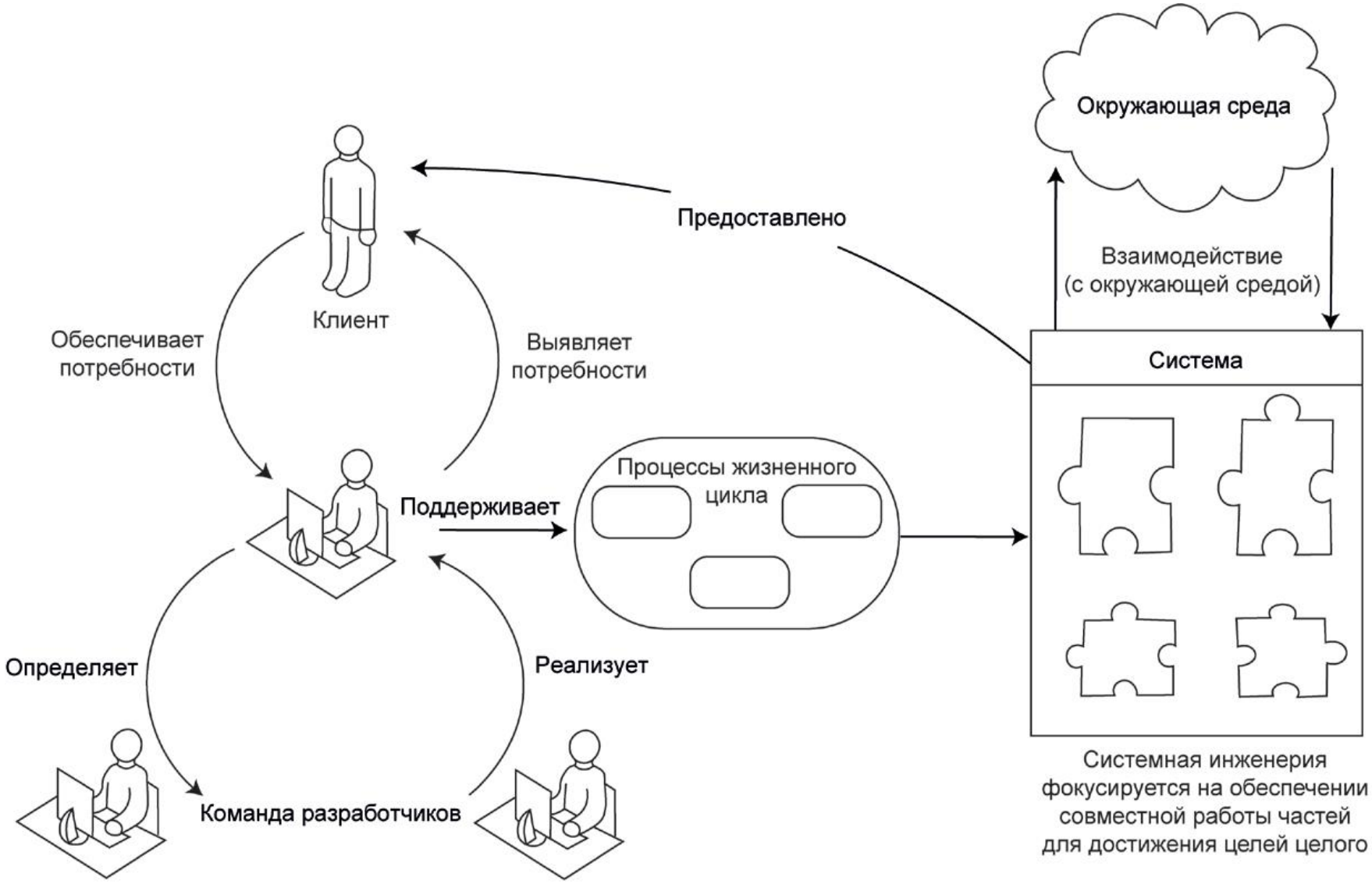


Рисунок 2. Роль системного инженера [https://sebokwiki.org/d/images/sebokwiki-farm!d/b/bd/SE_Key_Concepts.jpeg].

Роль системной инженерии (SE)

- Роль системной инженерии [SE] заключается в определении:

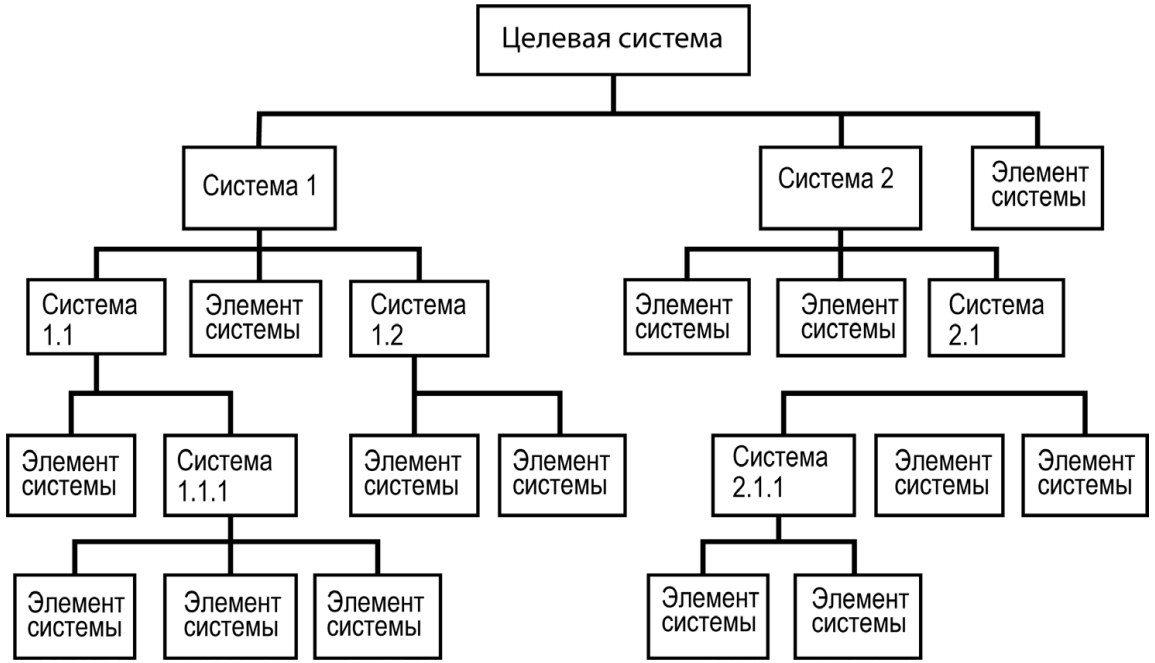
- Системных требований и ограничений
- Распределений требований
- Поведения системы и
- Структурных характеристик систем

- Система определяется с точки зрения иерархических структурных элементов и их поведенческих взаимодействий
- Взаимодействия включают обмен данными, энергией, силой или массой, который изменяет состояние взаимодействующих элементов, что приводит к возникающему, дискретному или непрерывному поведению
- Поведения находятся на последовательных уровнях агрегации [снизу вверх] или декомпозиции [сверху вниз] для удовлетворения требований, ограничений и распределения

Глава 2. Модели жизненного цикла систем

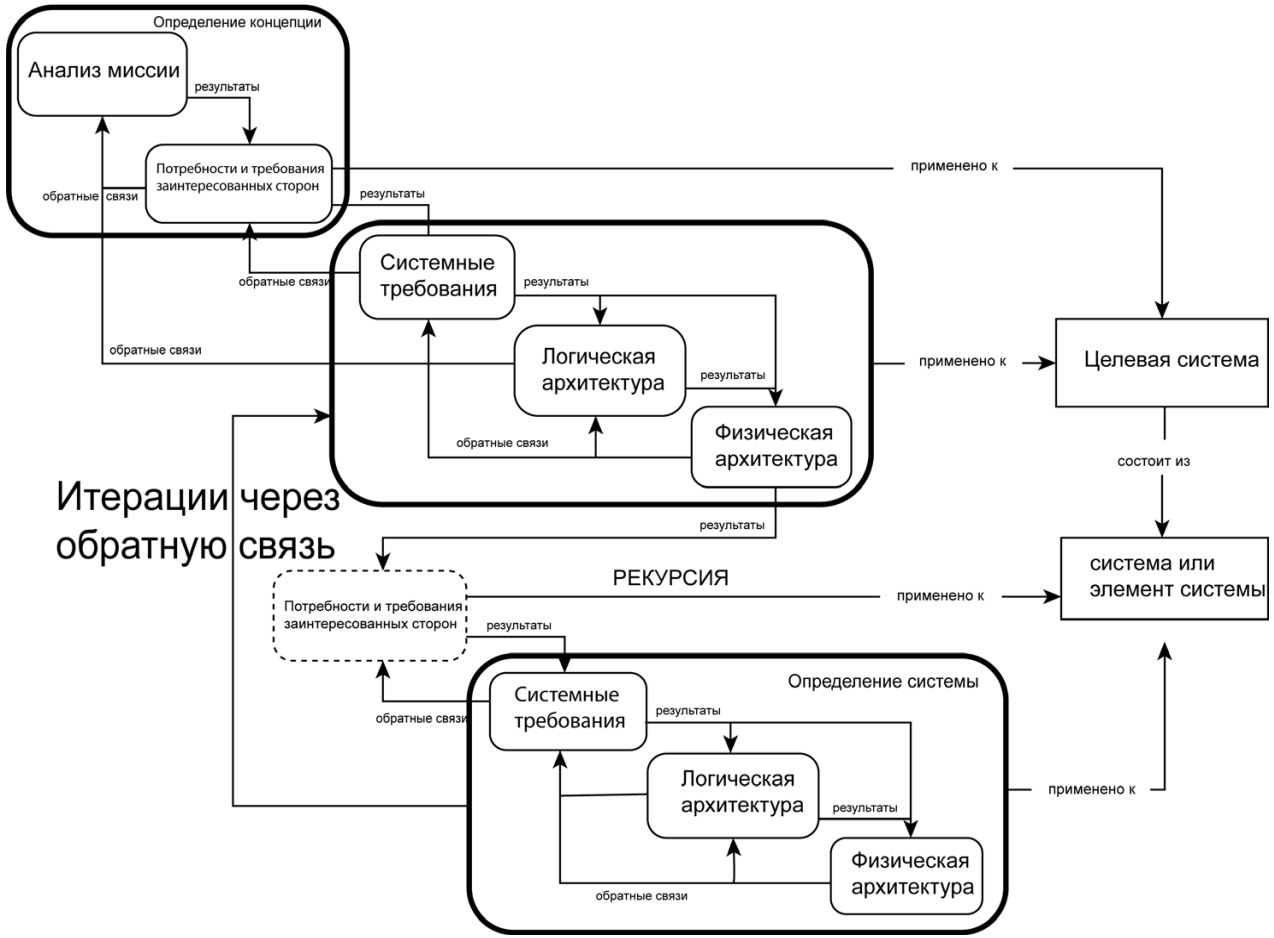
2.1. ЖИЗНЕННЫЕ ЦИКЛЫ СИСТЕМ И ИХ РЕКУРСИВНЫЙ ХАРАКТЕР
2.2. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ И ИХ КЛАССИФИКАЦИЯ
2.3. ФАЗЫ И СТАДИИ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ
2.4. ПРЕДВАРИТЕЛЬНО ЗАДАННАЯ ОДНОШАГОВАЯ МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА SOI, МОДЕЛИ ТИПА VEE
2.5. ИНКРЕМЕНТАЛЬНЫЕ МОДЕЛИ ПРОЦЕССОВ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ
2.6. ЭВОЛЮЦИОННЫЕ МОДЕЛИ ПРОЦЕССОВ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ
2.7. ГИБКИЕ МОДЕЛИ ПРОЦЕССОВ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ (AGILE SYSTEMS ENGINEERING)
2.8. ВОПРОСЫ ЗАЩИТЫ ИНФОРМАЦИИ В ПРОЦЕССЕ УПРАВЛЕНИЯ МОДЕЛЬЮ ЖИЗНЕННОГО ЦИКЛА СИСТЕМЫ

Глава 2. Модели жизненного цикла систем



Иерархическая декомпозиция SoI

Рекурсивность процессов жизненного цикла в соответствии с уровнями конструктивной иерархии SoI

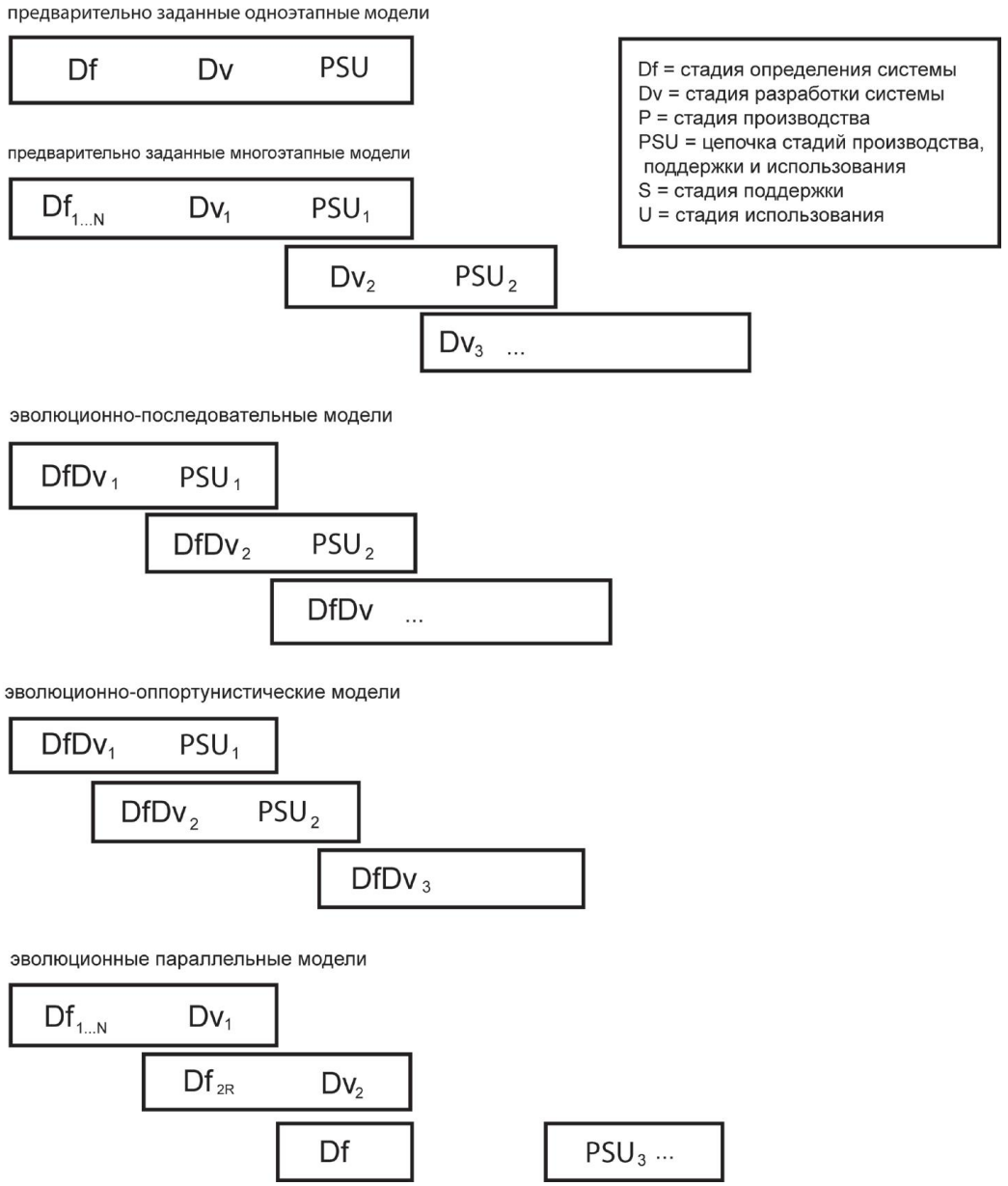


2.2. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ И ИХ КЛАССИФИКАЦИЯ

Модели делятся на следующие основные категории:

- 1) предварительно заданные одноэтапные (pre-specified, single-ste and sequential processes (e.g. the single-step waterfall model)), также известные как традиционные или последовательные процессы;
- 2) предварительно заданные многоэтапные (pre-specified, multi-step processes (e.g. the multi-step waterfall model));
- 3) эволюционно-последовательные;
- 4) эволюционно-оппортунистические;
- 5) эволюционные параллельные (или инкрементно-гибкие — incremental agile).

2.2. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ И ИХ КЛАССИФИКАЦИЯ



Pre-specified, Single-Step (PS) — предварительно заданные одноэтапные модели;
Pre-specified, Multi-Step (PM) — предварительно заданные многоэтапные модели;
Evolutionary Sequential (ES) — эволюционно-последовательные модели;
Evolutionary Opportunistic — эволюционно-оппортунистические модели;
Evolutionary Concurrent (EC) — эволюционные параллельные модели.
Отметим характерные особенности определенных выше категорий моделей жизненного цикла систем.

Процессы жизненного цикла системы

1. Предварительно заданная одношаговая модель (Pre-specified Single-step), примером которой является традиционная каскадная модель или последовательная модель Vee, применяется, если требования к SoI предварительно определены и имеют низкую вероятность значительных изменений, а также если нет возможности или смысла предоставлять частичные возможности продукта.

2. Предварительно заданная многоэтапная модель (Pre-specified multi-step models) разделяет разработку на части, чтобы задействовать ранние начальные эксплуатационные возможности (pre-planned product improvements — P3I) на протяжении жизненного цикла. Для данного подхода желательно, чтобы все возможности продукта были бы указаны заранее и вероятность значительных изменений мала. Он применяется, когда ожидание разработки полной системы влечет за собой потерю важных и полезных дополнительных возможностей конечной цели создания системы.

3. Эволюционно-последовательная модель (Evolutionary Sequential model) развивает первоначальные операционные возможности (IOC) и совершенствует их на основе опыта эксплуатации. Чистая гибкая (Agile) разработка программного обеспечения соответствует этой модели (если обнаружены дефекты в ПО и их нужно изменить, то исправления появятся через 30 дней в следующем релизе. Быстрое развертывание также подходит для этой модели для больших или программно-аппаратных систем. Его главное достоинство заключается в том, чтобы обеспечить возможности быстрого реагирования в процессе использования.

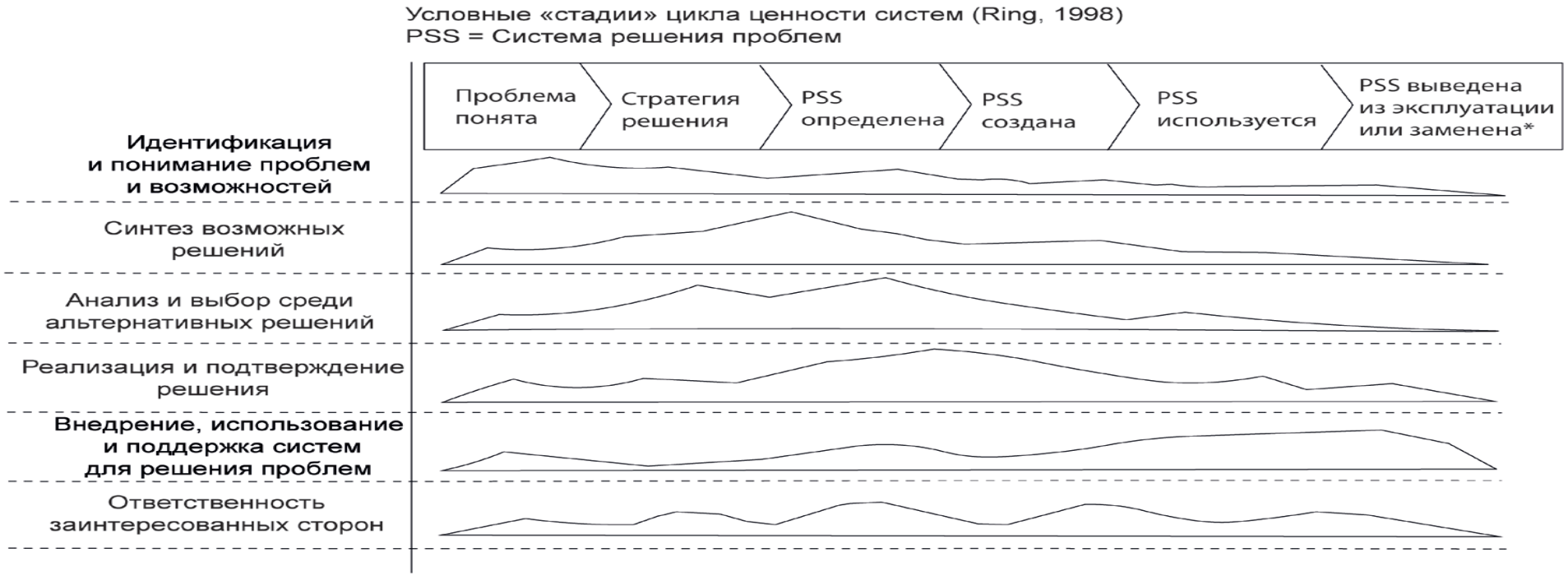
4. Эволюционно-оппортунистическая модель (Evolutionary Opportunistic model) может быть принята в случаях, когда следующее приращение откладывается до тех пор, пока: появляется достаточно привлекательная возможность, желаемая новая технология или до тех пор, пока не станут доступными другие факторы, такие как дефицитные компоненты или ключевой персонал. Такой подход также подходит для синхронизации обновлений нескольких готовых коммерческих продуктов (multiple commercial-off-the-shelf — COTS). Этот подход лучше всего использовать, когда для инкремента системы не требуется ждать оперативной обратной связи о ее использовании, а может потребоваться ожидание таких факторов для реализации инкремента, как технологическая зрелость, возможности внешней системы, необходимые ресурсы или новые возможности добавления ценности.

5. Эволюционно-параллельная модель (Evolutionary Concurrent model) реализуется на основе спиральной модели возрастающих обязательств [1]. Для нее характерным является работа постоянной команды системных инженеров, обрабатывающих трафик изменений, планы и спецификации для следующего шага проекта, а также стабильной команды разработчиков для своевременной и надежной реализации текущего шага и использование одновременной проверки и валидации (V&V) для обеспечения более высокого уровня надежности

Процессы жизненного цикла системы

Процесс – это последовательность действий или шагов, предпринятых для достижения определенной цели, преобразующие входные данные в выходные

Процессы жизненного цикла системной инженерии определяют технические и управленческие действия, выполняемые на одном или нескольких этапах для предоставления информации, необходимой для принятия решений в течение жизненного цикла; и обеспечить реализацию, использование и поддержку интересующей системы (SoI)



Показывает одновременную деятельность в рамках «одного цикла» системного подхода * PSS выведена из эксплуатации или заменена может означать переоценку проблемной ситуации и формулировку нового понимания проблемы.

Предварительно заданная одношаговая модель жизненного цикла SoI, модели Vee

Типичный вариант традиционного многоэтапного предварительно заданного жизненного цикла модели SoI [2)]

Стадии предварительно заданной одношаговой модели жизненного цикла Sol

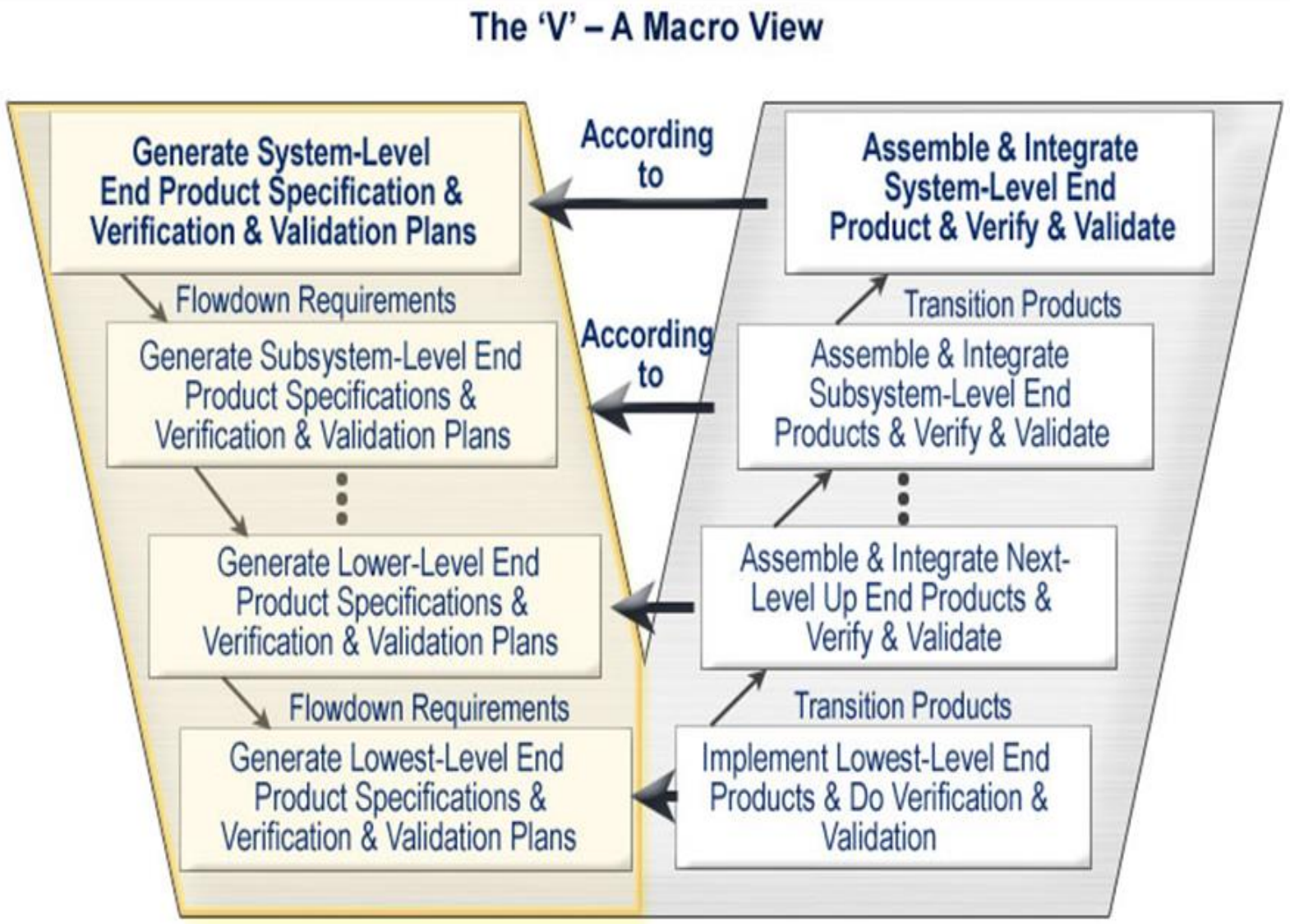
Данная модель включает следующие стадии (описанные в стандарте ISO ISO/IEC 15288):

- **Исследовательский (Exploratory)**
- **Концепция (Concept)**
- **Разработка (Development)**
- **Производство (Production)**
- **Использование (Utilization)**
- **Поддержка (Suppot)**
- **Вывод из эксплуатации (Retirement)**

После завершения каждой стадии выполняется процедура **Решатель (Decision Gate**), которая принимает одно из следующих альтернативных решений:

- **Прейти к следующей стадии**
- **Прейти к следующей стадии, но открыть активности по элементам, которые должны быть перепроектированы**
- **Нет готовности: повторить предыдущую стадию**
- **Завершить проект**

Предварительно заданная одношаговая модель жизненного цикла SoI, модель Vee



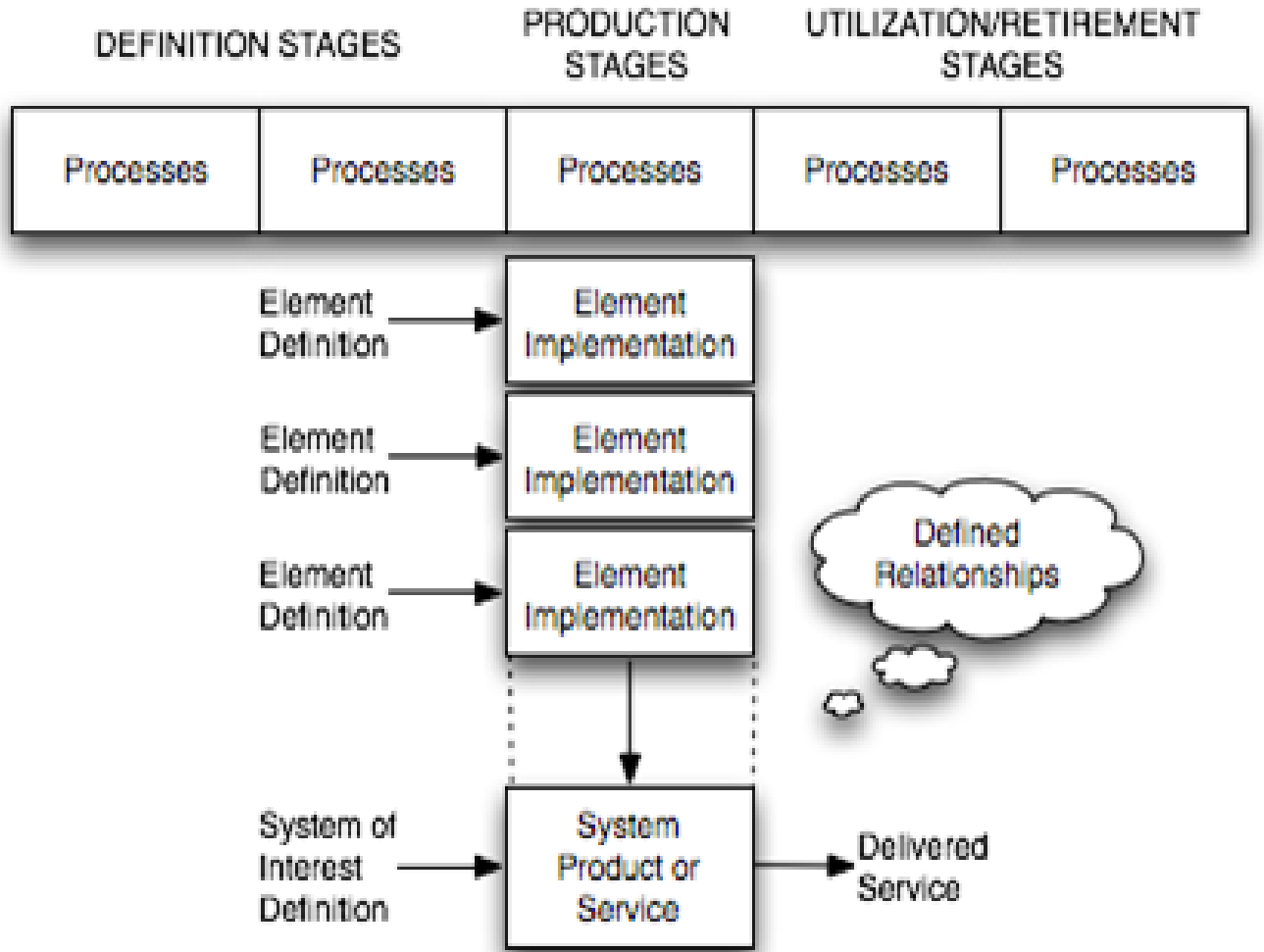
Важным свойством модели Vee являются горизонтальные соответствия между стадиями одного уровня, по существу показывающие связь между процессами проверки и валидации с соответствующими им спецификациям, по которым эти процессы будут тестировать соответствующие компоненты систем.

Стадии левой нисходящей и правой восходящей ветви модели Veeветви модели Vee

- **Левая нисходящая ветвь модели Vee включает стадии:**
- **General System-Level End Product Specification & Verification & Validation Plans** (Общая спецификация конечного продукта на уровне систем и планы проверки и валидации)
- **General Subsystem-Level End Product Specification & Verification & Validation Plans** (Общая спецификация конечного продукта на уровне подсистем и планы проверки и валидации)
- **General Lower-Level End Product Specification & Verification & Validation Plans** (Общая спецификация конечного продукта нижнего уровня и планы проверки и валидации)
- **General Lowest-Level End Product Specification & Verification & Validation Plans** (Общая спецификация конечного продукта самого низкого уровня и планы проверки и валидации)
- Стадии нисходящей ветви связаны между собой передачей потоков требований по мере продвижения вниз в виде все более формализованных спецификациях
- **Правая восходящая ветвь модели Vee включает стадии:**
- **Assemble & Integrate System-Level End Product & Verify & Validate** (Сборка и интеграция конечного продукта системного уровня, проверка и валидация)
- **Assemble & Integrate Subsystem-Level End Product & Verify & Validate** (Сборка и интеграция конечного продукта уровня подсистем, проверка и валидация)
- **Assemble & Integrate Next-Level Up End Product & Verify & Validate** (Сборка и интеграция конечного продукта следующего уровня, а также проверить и испытать (подтвердить))
- **Assemble & Integrate Lowest-Level Up End Product & Do Verification & Validate** (Сборка и интеграция конечного продукта самого низкого уровня, а также выполнить его проверку и валидацию)
- Стадии восходящей ветви связаны между собой передачей снизу-вверх так называемого **переходного продукта** (**Transition Product**).

Рекурсивная и параллельная Vee модели жизненного цикла систем

В том случае, когда целевая система представляет собой композицию, состоящую из многих элементов, которые сами могут рассматриваться как SoI, применяется **рекурсивная декомпозиция исходной системы**, в которой реализация каждого системного элемента осуществляется повторным использованием модели жизненного цикла системы, но на следующем более низком уровне конструктивной иерархии, рассматривая системный элемент как самостоятельный SoI



Общая структура Т-стадий жизненного цикла систем

Инкрементальные модели процессов жизненного цикла системы

- Инкрементальные или пошаговые модели жизненного цикла систем широко используются тогда, когда с помощью рассмотренного выше традиционного или водопадного (traditional or waterfall) подхода создана SoI с первоначальными операционными возможностями (**initial operational capability - IOP**), а затем планируется провести улучшение созданной базовой версии SoI
- Методы инкрементной разработки позволяют быстро создать действующую системы с минимальными возможностями, а затем с помощью инкрементального проекта поэтапно (пошагово) развивать базовую версию SoI до конечной SoI с заданными требованиями.
- Инкрементальный подход используется, когда:

- желательно осуществить быстрое исследование и внедрение части SoI

- требования изначально неясны

- финансирование ограничено

- заказчик желает оставить SoI открытой для возможности внедрения новой технологии в более позднее время и/или - требуется экспериментирование для разработки последовательных версий прототипа

- В этой модели применяются три **точки обзора проекта** (решатели):
- предварительного анализа проекта (**preliminary design review** - **PDR**) для проверки и утверждения набора системных требований, артефактов проекта и элементов обоснования в конце очередного цикла проектирования (также известного как решатель или ворота «проектирование-до»);
- критического обзора проекта (**critical design review - CDR**) для проверки и испытаний набора системных требований, артефактов проекта и элементов обоснования в конце цикла проектирования;
- точки обзора типа TRR (**test readiness review**) - многопрофильной проверки, предназначенной для обеспечения того, чтобы проверяемая подсистема или система была готова перейти к формальному тестированию, т.е. испытаниям -

Инкрементальные модели процессов жизненного цикла системы

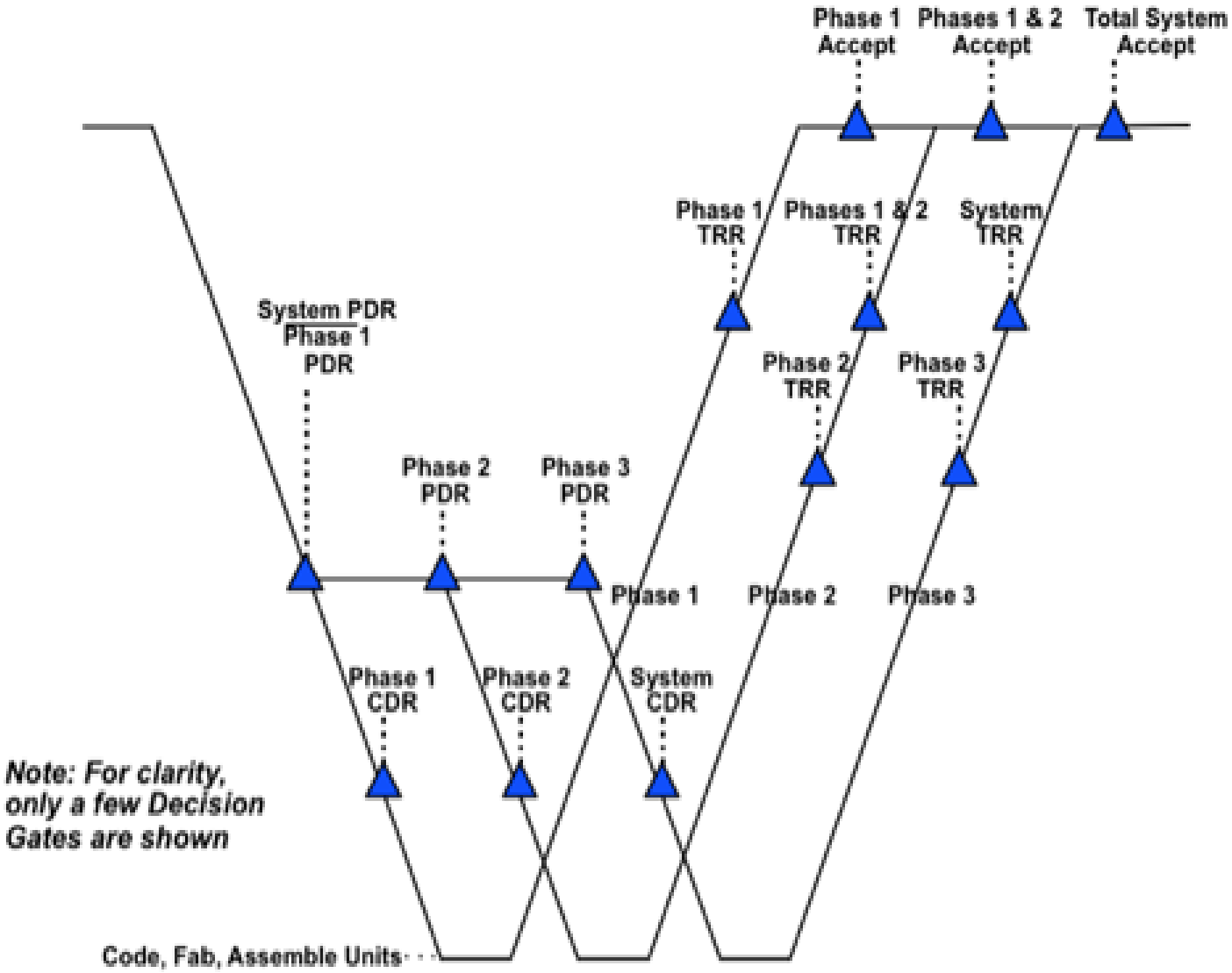
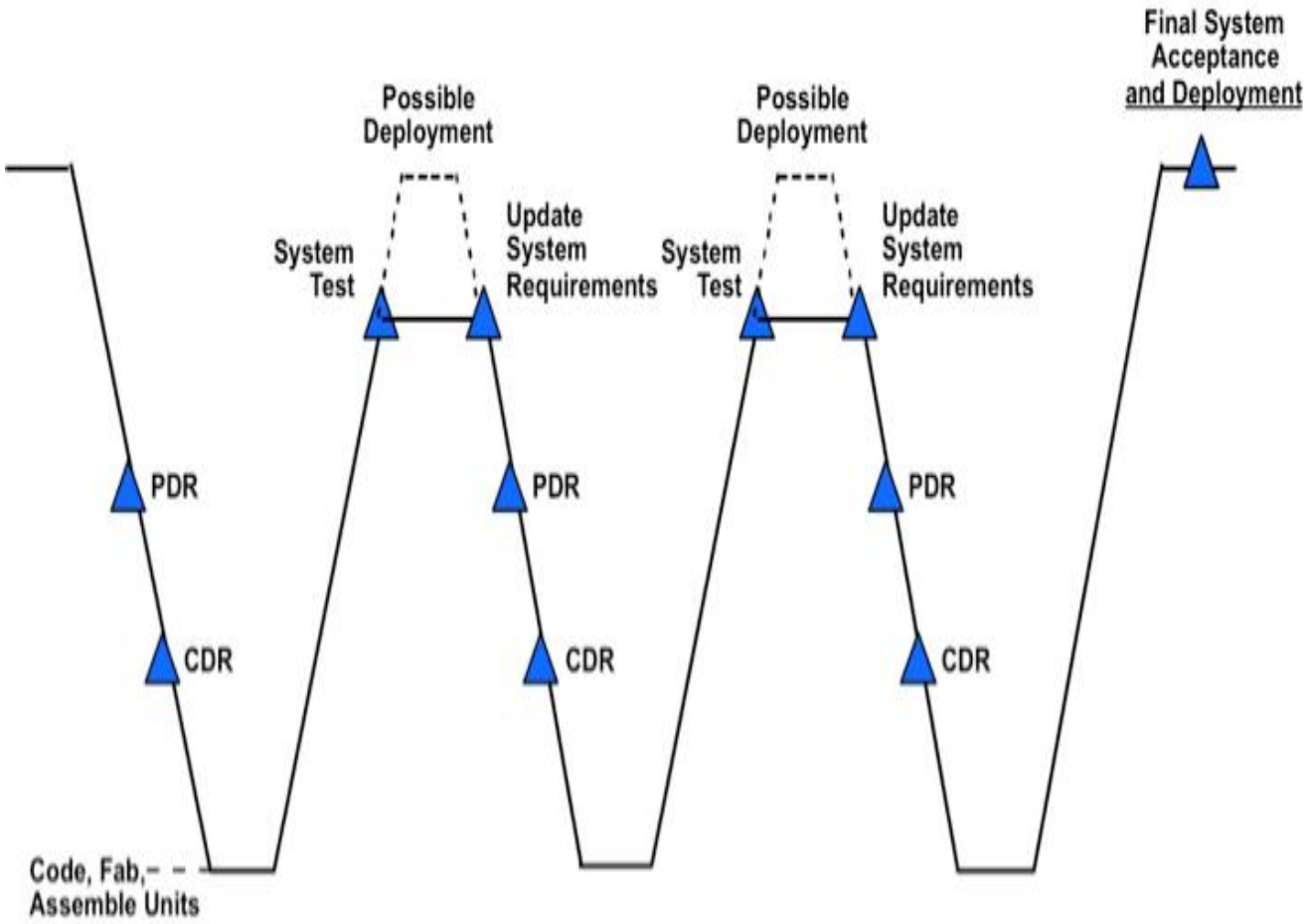


Рисунок 9. Инкрементальный подход с множеством шагов/фаз развития базовой версии системы и поставками (deliveries) продукта конечному пользователю.

Модель процессов жизненного цикла системы эволюционного подхода

- Обзор эволюционного подхода
- Особая методология, называемая эволюционным развитием, распространена в сфере исследований и разработок (НИОКР) как в государственном, так и в коммерческом секторе.
- Рисунок 10 иллюстрирует этот подход, который использовался при разработке высокотемпературных плиток для космического корабля НАСА (Форсберг, 1995).
- При эволюционном подходе конечная стадия каждой фазы развития неизвестна, хотя цель каждой фазы состоит в том, чтобы привести к созданию какого-то полезного продукта.
- Реальная среда разработки сложна, и ее трудно отобразить, поскольку одновременно реализуется множество различных проектных циклов.
- Конкретная методология, называемая эволюционным развитием, распространена в среде исследований и разработок (НИОКР) как в государственном, так и в коммерческом секторе. Такой подход в организации жизненного цикла систем иллюстрируется на Рис. 10.
- В эволюционном подходе, конечное состояние каждой фазы разработки неизвестно, хотя цель каждой фазы состоит в том, чтобы привести к какому-то полезному продукту.
- На практике применяются модели жизненного цикла инкрементального и эволюционного развития системы. Такие модели сочетают в себе пошаговый принцип наращивания функциональности продукта с возможностью его качественного совершенствования с каждым циклом проекта. Они представляют собой комбинации рассмотренных выше подходов.

Эволюционные модели процессов жизненного цикла системы



На представленной на Рис. 10 модели жизненного цикла каждый проектный шаг начинается с создания компонент и сборки очередной версии конечного продукта (Code, Fab, -, Assemble Units), затем полученная версия проходит тестирование (System Test).
Если тестирование прошло успешно, то возможно развертывание системы, далее выполняется обновление системных требований (**Update System Requirements**), после чего начинается нисходящая ветвь разработки новой версии системы с двумя точками обзора (решателями) проекта: предварительного анализа проекта PDR и критического обзора проекта CDR.

Рис. 10. Эволюционная общая модель - https://sebokwiki.org/d/images/sebokwiki-farm!d/b/b2/Evolutionary_Generic_Model.PNG [1].

Гибкие модели процессов жизненного цикла системы: Agile Systems Engineering

- В современном мире сложные и взаимосвязанные инженерные системы сталкиваются с быстрым устареванием под давлением технологических изменений, изменений окружающей среды и быстро меняющихся потребностей. Чтобы эти системы оставались устойчивыми к изменениям, они должны быть спроектированы гибкими, адаптируемыми, способными к расширению своей функциональности
- Разработка новых методологических решений сформировало новое направление гибких технологий проектирования инженерных систем, получившее название процессов **Agile SE**, которое базируется на особой культуре работы системных инженеров
- Такая культура в свою очередь основывается на так называемом **Agile-мышлении** или **Agile Mindset** - наборе убеждений и действий, основанных на **agile-ценностях**, которые делают акцентированное внимание на максимальном использовании возможностей исполнителей проектов, доверие работникам умственного труда в поиске наилучшего решения, улучшение продукта и процесса посредством регулярных демонстраций и ретроспектив, частое планирование для реализации извлеченных уроков.
- В процессе **Agile SE** системный инженер работает итеративно, поэтапно, постоянно моделируя, анализируя, разрабатывая и торгуя вариантами, чтобы сосредоточить внимание на определении системных решений, руководствуясь **Agile-принципами**, основные из которых являются:

Принципы гибкой разработки

1. Удовлетворить клиента за счет раннего и непрерывного предоставления ценных возможностей
2. Планируйте меняющиеся требования и сохраняйте столько гибкости, сколько необходимо на протяжении всей разработки; гибкие процессы используют изменения для клиента, особенно когда изменения приводят к конкурентному преимуществу
3. Предоставляйте рабочие возможности часто, от пары недель до пары месяцев, отдавая предпочтение более коротким временным рамкам
4. Бизнес-персонал, клиенты или их сторонники и исполнители должны ежедневно работать вместе на протяжении всего проекта
5. Стройте проекты вокруг мотивированных людей. Обеспечьте им необходимые условия и поддержку и доверьте им выполнение работы
6. Самый оперативный и действенный способ передачи информации — личный разговор
7. Рабочие способности являются основным мерилом прогресса
8. Гибкие процессы способствуют устойчивому развитию. Спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределенный срок
9. Постоянное внимание к техническому совершенству и хорошему дизайну повышает гибкость
10. Простота «искусство максимизировать объем невыполненной работы» имеет важное значение, особенно в группе внедрения. По-настоящему гибкий проект разработки не навязывает искусственную отчетность и технологические требования группе внедрения
11. Лучшие архитектуры, требования и проекты создаются самоорганизующимися командами, основанными на минимальном наборе руководящих принципов
12. Через регулярные промежутки времени команда размышляет о том, как стать более эффективной, затем соответствующим образом настраивает и корректирует свое поведение

Принципы гибкой разработки

- Согласно стандарта ISO/IES/IEEE 24748-1 «Системная и программная инженерия. Управление жизненным циклом. Часть 1. Руководство по управлению жизненным циклом» существует шесть асинхронных и параллельных стадий жизненного цикла системы: **Concept**, **Development**, **Production**, **Utilization**, **Support**, **Retierement**.
- В представленной на Рис. 11 модели добавлена седьмая стадия жизненного цикла - **ситуационная осведомленность** (***Situational Awareness***) [Dove 2019].
- Используя эту модель жизненного цикла разработки гибких систем (ASELCM), проект может начинаться на стадии «Концепция», переходить в фазу ситуационной осведомленности, затем переходить в фазу разработки, обратно в фазу ситуационной осведомленности и так далее по кругу.
- Эта фаза ситуационной осведомленности позволяет акцентировать внимание на проактивном осознании ситуационных возможностей и рисков как для процессов, так и для продуктов на протяжении всего процесса системной инженерии и жизненного цикла целевой системы.
- Эта упреждающая осведомленность и последующая деятельность по смягчению последствий вдыхают жизнь в процесс разработки гибких систем, выводя его за рамки повторяющегося выполнения традиционных приращений разработки, выполняющих невыполненные запланированные функции.
- Структура гибкой модели жизненного цикла допускает постоянную эволюцию после первоначальной доставки и требует, чтобы на стадии разработки создавалась гибкая целевая система для поддержки эволюции. Стадия ситуационной осведомленности является отправной точкой для всей остальной деятельности стадии и включает в себя исходное осознание того, что необходима новая система.

Гибкие модели процессов жизненного цикла системы: Agile Systems Engineering

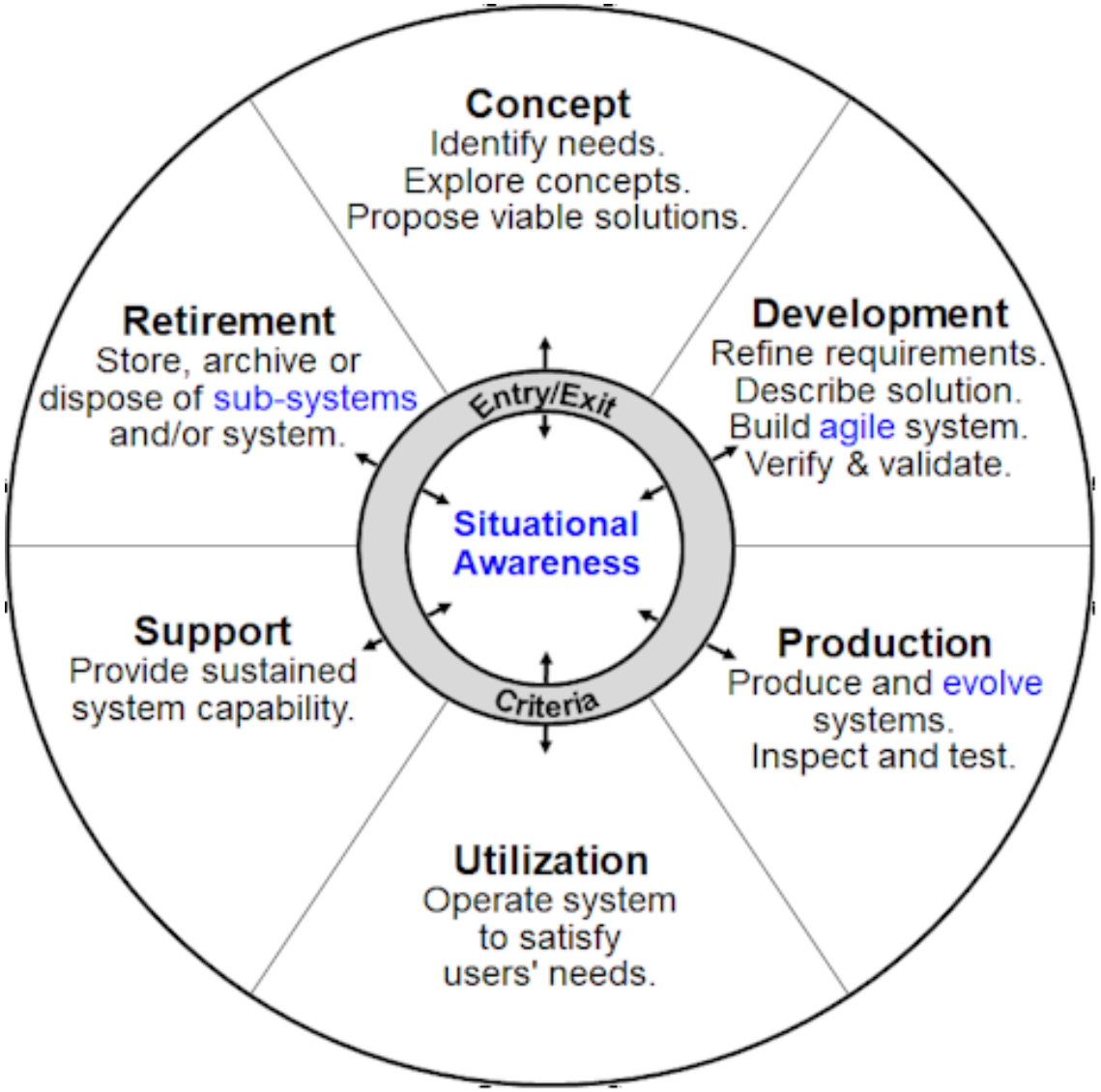


Рис. 11. Цели для каждой стадии модели жизненного цикла, адаптированные из (ISO/IEC/IEEE 2018, стр. 17), с добавлением стадии «ситуационной осведомленности» - []

Глава 3. Модели итеративного процесса разработки программного обеспечения

- **3.1.** КЛАССИФИКАЦИЯ СИСТЕМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И МОДЕЛЕЙ ИХ ЖИЗНЕННОГО ЦИКЛА
- **3.2.** ИНКРЕМЕНТАЛЬНАЯ СБОРКА (INCREMENTAL-BUILD)
- **3.3.** ПРОТОТИПИРОВАНИЕ (PROTOTYPING) В РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
- **3.4.** СПИРАЛЬНЫЕ МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА
- **3.5.** ГИБКАЯ МОДЕЛЬ РАЗРАБОТКИ ПО (AGILE) — ИТЕРАТИВНАЯ ЭВОЛЮЦИЯ ТРЕБОВАНИЙ И КОДА
- **3.6.** ЭВОЛЮЦИОННЫЙ ПОСЛЕДОВАТЕЛЬНЫЙ ПРОЦЕСС СКРАМ (SCRUM)
- **3.7.** ПРИНЦИПЫ БЕРЕЖЛИВОГО ПРОИЗВОДСТВА И РАЗРАБОТКИ ПО
- **3.8.** РУКОВОДСТВО ПО СВОДУ ЗНАНИЙ В ОБЛАСТИ ПРОГРАММНОЙ ИНЖЕНЕРИИ SWEBOK

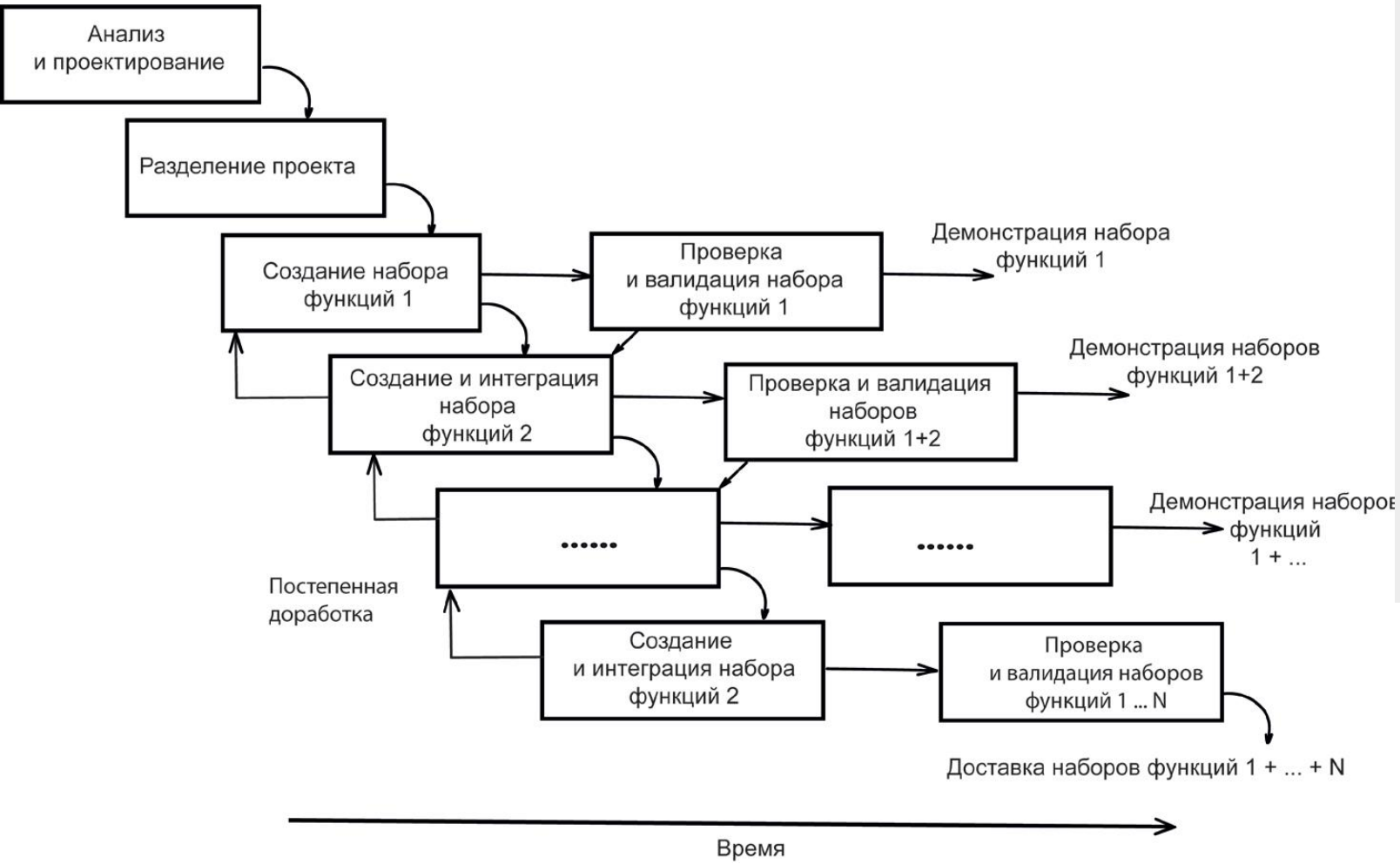
Модели итеративного процесса разработки программного обеспечения

- Итеративный подход подразумевает, что реализация проекта организована в виде последовательности итераций
- Каждая итерация модели жизненного цикла разработки программной системы добавляет код к растущей базе ПО, завершаясь получением согласованных результатов выполнения итерации (артефактов системы), включающих работающий код программного релиза системы, соответствующий данной итерации, модели проекта и документацию
- Расширенная кодовая база тестируется, при необходимости перерабатывается и демонстрируется, что она удовлетворяет пользовательским требованиям
- Каждый инкремент добавляет функциональность к основному коду создаваемой программной системы, однако при этом, возможно, не привнося качественных изменений свойств системы. В этом случае подход можно назвать чисто итеративным
- В случае, когда итерация приводит к качественным изменениям свойств проекта итерационный подход становится итерационным и эволюционным, так как под эволюцией понимается качественное изменение свойств проекта.

Рассматриваются следующие модели процессов для разработки ПО:

- **Инкрементальная сборка (Incremental-build) - итеративные циклы внедрения-проверки-валидации-демонстрации;**
- **Прототипирование** (**Prototyping**) – первоначальная разработка упрощенного макета целевого продукта;
- **Спиральный** (**Spiral**) - итеративный риск-ориентированный анализ альтернативных подходов и оценка результатов;
- **Гибкая модель** (**Agile**) - итеративная эволюция требований и кода

Инкрементальная сборка (Incremental-build)



Процесс инкрементной сборки обычно хорошо работает с небольшими командами, но его можно масштабировать для более крупных проектов.
Достоинствами данного подхода являются:
- непрерывная интеграция, проверка и проверка развивающегося продукта;
- частые демонстрации прогресса; раннее выявление дефектов;
- раннее предупреждение о технологических проблемах; и
- систематическая отработка переделок, вызванных устранением обнаруженных ошибок

Прототипирование (prototyping) в разработке программного обеспечения

- В программной инженерии (Software Engineering - SwE) прототип — это макет желаемой функциональности некоторой части системы, что отличается от физических систем, где прототип обычно представляет собой первую полнофункциональную версию системы, которую передают на серийное производство.
- Одной из основных задач прототипирования и является вовлечение заказчика в отработку требований к создаваемой системе. Это достигается посредством техники быстрых релизов упрощенной экспериментальной или прототипной системы, которая легко модифицируется и позволяет итеративно и быстро отрабатывать требования к проектируемому продукту и с помощью оперативной обратной связи с заказчиком, демонстрировать ему полученные результаты, благодаря чему заказчик становится активным участников процесса создания системы. В результате такого подхода заказчики (конечные пользователи) системы принимают активное участие в проектировании системы, анализируя результаты работы быстрого прототипа они выдают свои замечания до тех пор, пока прототип не будет удовлетворять всем их требованиям.
- Такой подход часто называют структурной эволюционной технологией быстрого прототипирования. При создании прототипа ПО, полученные в ходе его разработки программные решения, как правило, не используется в готовой версии программного продукта. Во многих случаях эффективнее создавать рабочий код конечного продукта с нуля, используя знания и опыт, полученные при прототипировании, чем перепроектировать существующий код прототипа.

Спиральные модели жизненного цикла

Спиральная модель процесса разработки ПО предложена Барри Боэмом в 1986 году [5].

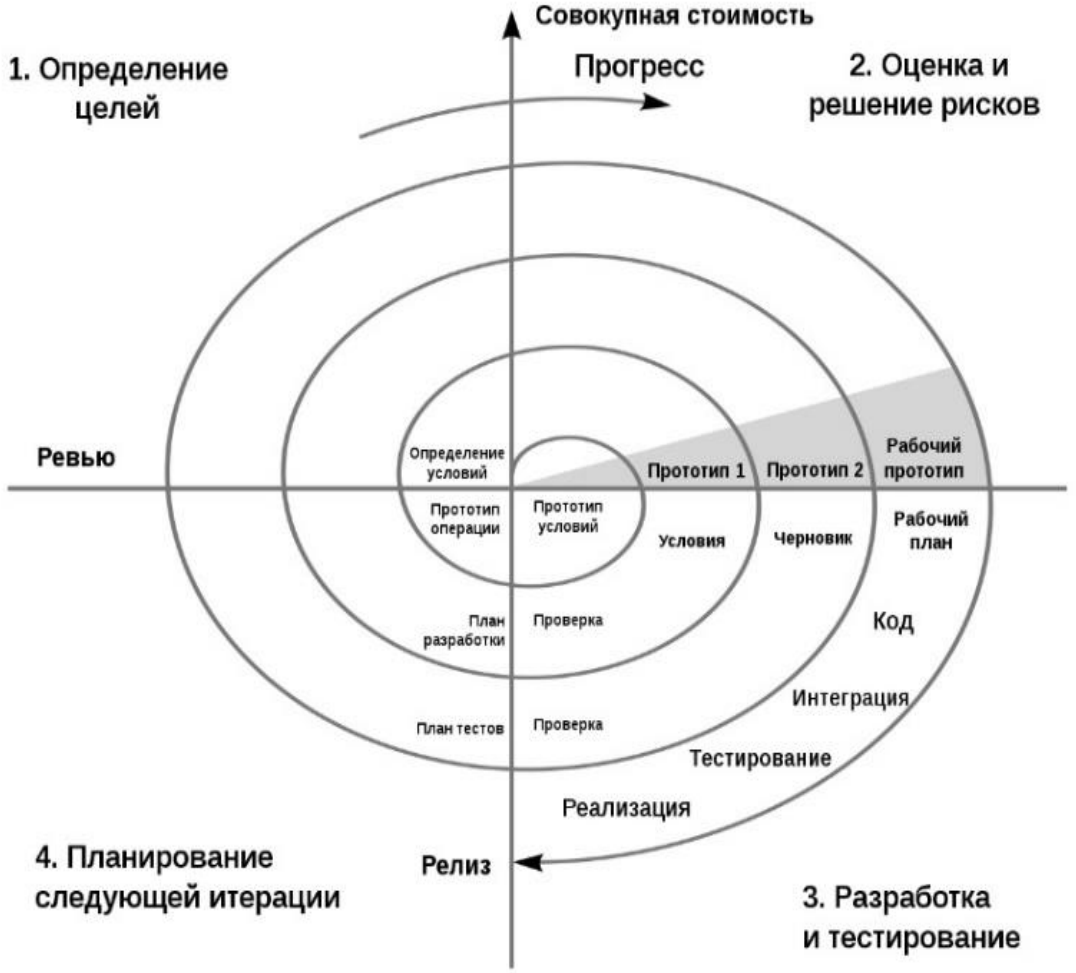
Она представляет собой модель процесса разработки ПО, сочетающий в себе как инкрементальность (пошаговость), так и итеративность.

Особое внимание в этой модели уделено рискам, влияющим на результат реализации жизненного цикла ПО.

Боэм формулирует десять наиболее распространённых (по приоритетам) рисков:

- Дефицит специалистов
- Нереалистичные сроки и бюджет
- Реализация несоответствующей функциональности
- Разработка неправильного пользовательского интерфейса
- «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей
- Непрекращающийся поток изменений
- Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию.

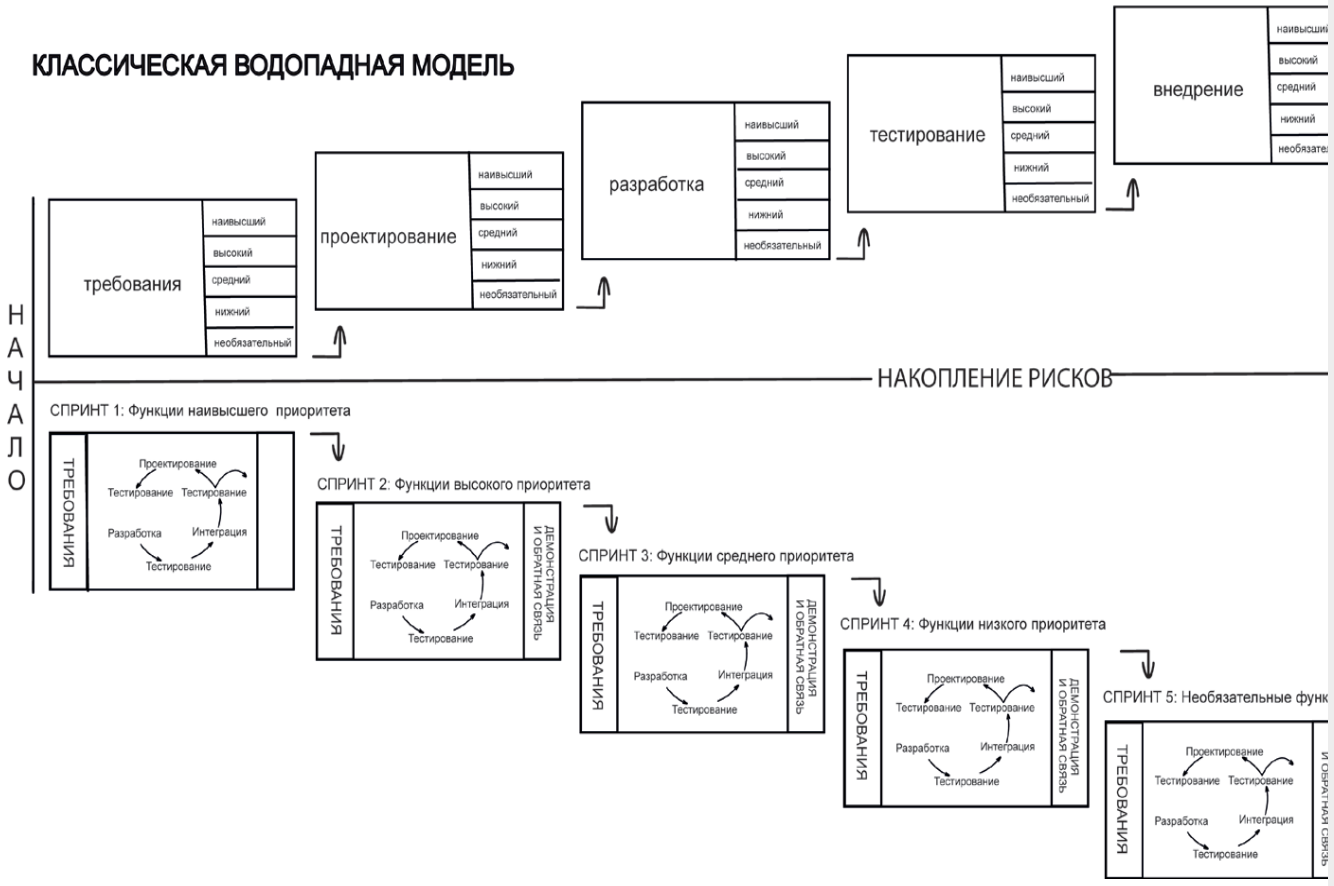
Спиральные модели жизненного цикла



Спиральная модель процесса разработки ПО предложена Барри Боэмом в 1986 году [5].
Она представляет собой модель процесса разработки ПО, сочетающий в себе как инкрементальность (пошаговость), так и итеративность.
Особое внимание в этой модели уделено рискам, влияющим на результат реализации жизненного цикла ПО.
Боэм формулирует десять наиболее распространённых (по приоритетам) рисков:
- Дефицит специалистов
- Нереалистичные сроки и бюджет
- Реализация несоответствующей функциональности
- Разработка неправильного пользовательского интерфейса «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей
- Непрекращающийся поток изменений
- Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию.

Рисунок 10. Спиральная модель Барри Боэма - https://commons.wikimedia.org/wiki/File:%D0%A1%D0%BF%D0%B8%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%91%D0%B0%D1%80%D1%80%D0%B8_%D0%91%D0%BE%D1%8D%D0%BC%D0%B0.svg [6].

Гибкая модель разработки ПО (Agile) - итеративная эволюция требований и кода

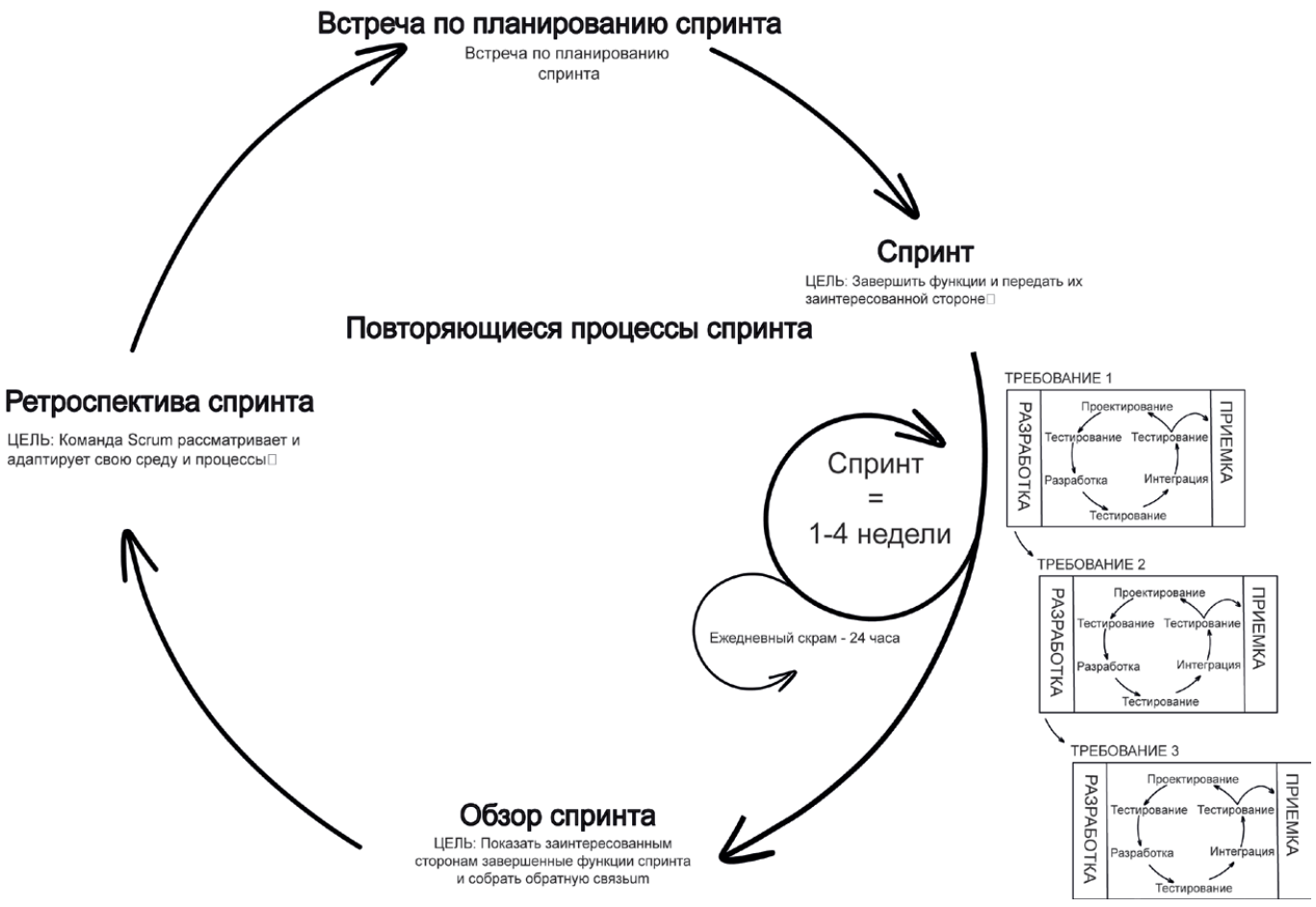


Agile - гибкий неформализованный подход в управлении проектами [6]. В отличие от классических моделей жизненного цикла данный подход не требует подробного задания на разработку точных спецификаций требований, также не применяется поэтапное планирование проекта и управление жизненным циклом.

Реализация гибкого процесса строится на выполнении работ по проекту на коротких временных отрезках, так называемых **спринтах**, длиной в одну–две недели, по окончании которых происходят совещания членов команды, на которых делается оценка выполнения задач, поставленных на предыдущей встрече, и ставятся задачи для следующего спринта.

Различие между классической водопадной моделью жизненного цикла ПО и моделью гибкого подхода иллюстрируется на Рис. 12.

Гибкая модель разработки ПО (Agile) - итеративная эволюция требований и кода



Работа спринта состоит в следующем:
Во время спринта проводятся постоянные проверки, чтобы оценить прогресс в достижении цели спринта и, следовательно, в достижении цели релиза.
Проводятся ежедневные собрания по схватке для организации работы на день, анализа того, что команда сделала вчера, и согласования того, над чем она будет работать сегодня.
По сути, скрам-команда проверяет свой прогресс в достижении цели спринта.
В конце спринта проводится ретроспективное совещание по спринту, чтобы оценить производительность и спланировать необходимые адаптации.

Рисунок 3.4. Работа спринта в Agile-методологии [5]. [5] Mark, C. Layton and Steven J Ostermiller / C. Mark. Agile Project Management For Dummies®, 2nd Edition Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com. Copyright © 2017 by John Wiley & Sons, Inc., Hoboken, New Jersey.

Эволюционный последовательный процесс Скрам (Scrum)

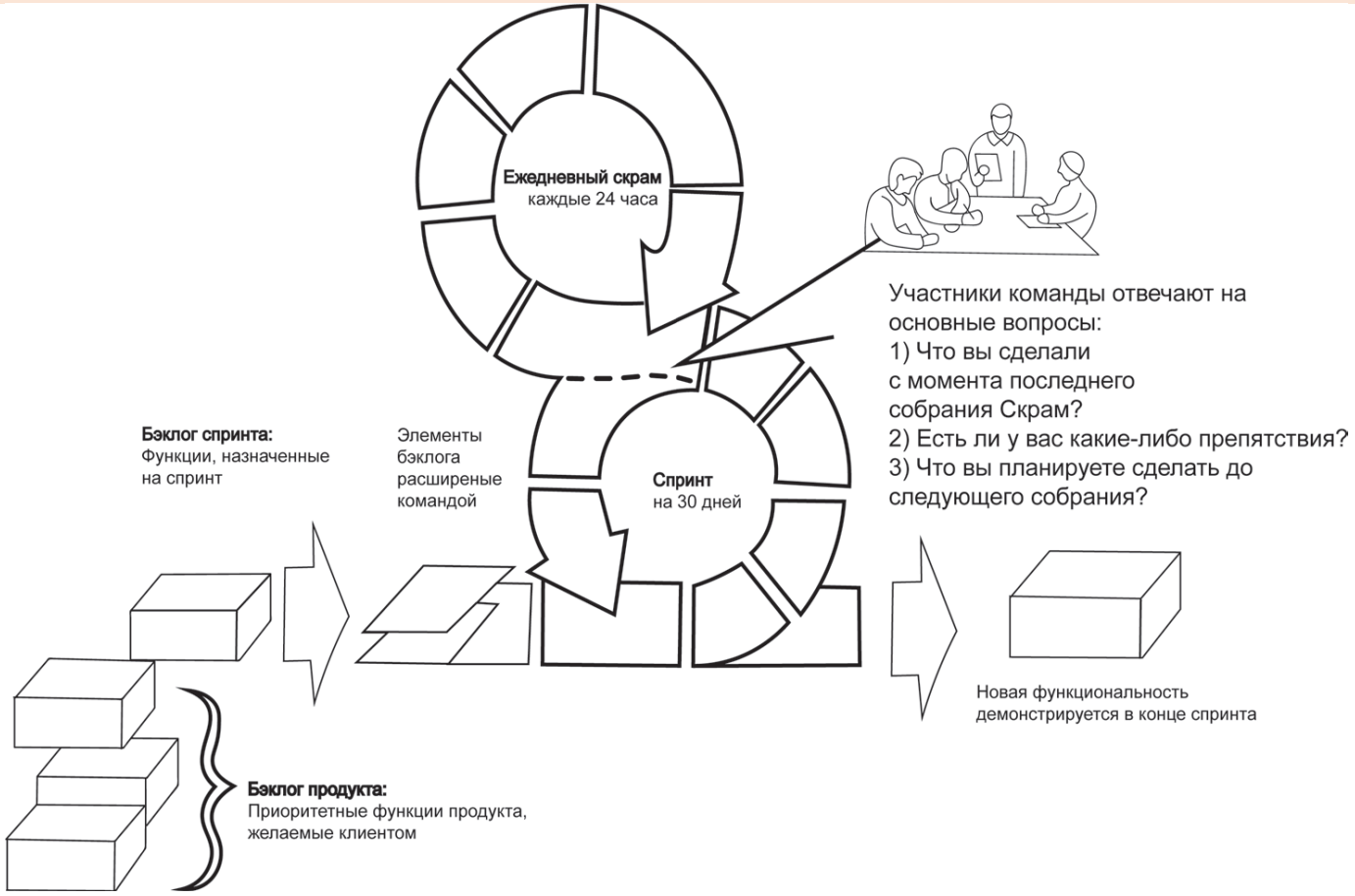


Рисунок 16. Пример Agile-процесса: Scrum (Бем и Тернер, 2004 г.) - https://sebokwiki.org/d/images/sebokwiki-farm!d/0/0c/Tale_of_Two_Implementations_Schwaber.jpg [1].

Рисунок 3.5. Состав Agile-команды [6]

ГЛАВА 4. ХАРАКТЕРНЫЕ ЧЕРТЫ И ВОЗМОЖНОСТИ ЯЗЫКА UML

4.1. ОСНОВНЫЕ ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА (АБСТРАКЦИЯ, ИНКАПСУЛЯЦИЯ, МОДУЛЬНОСТЬ, ОБОБЩЕНИЕ И НАСЛЕДОВАНИЕ, АГРЕГАЦИЯ И КОМПОЗИЦИЯ, ИНТЕРФЕЙСЫ, ПОЛИМОРФИЗМ)

4.2. ИСТОРИЯ СОЗДАНИЯ И НАЗНАЧЕНИЕ ЯЗЫКА МОДЕЛИРОВАНИЯ UML

4.3. СРЕДСТВА ЯЗЫКА UML ДЛЯ МОДЕЛИРОВАНИЯ СТРУКТУРЫ СИСТЕМЫ (ОБЪЕКТЫ, КЛАССЫ, СТРУКТУРИРОВАННЫЙ КЛАСС, ПАКЕТЫ, АКТЕРЫ)

4.4. СРЕДСТВА ЯЗЫКА UML ДЛЯ МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ СИСТЕМЫ (СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ, ДЕЯТЕЛЬНОСТИ, ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТИ, КОМБИНИРОВАННЫЙ ФРАГМЕНТ, КОНЕЧНЫЕ АВТОМАТЫ, ВРЕМЕННЫЕ ДИАГРАММЫ)

4.5. РЕАЛИЗАЦИЯ МОДЕЛИРОВАНИЯ (ДИАГРАММЫ КОМПОНЕНТОВ, ДИАГРАММЫ РАЗВЕРТЫВАНИЯ)

ГЛАВА 5. ЯЗЫК МОДЕЛИРОВАНИЯ SYSML И ЕГО ПРИМЕНЕНИЕ ДЛЯ РАЗРАБОТКИ МОДЕЛЕЙ СИСТЕМ

5.1. ИСТОРИЯ СОЗДАНИЯ И НАЗНАЧЕНИЕ ЯЗЫКА МОДЕЛИРОВАНИЯ SYSML

5.2. СРЕДСТВА ЯЗЫКА SYSML ДЛЯ МОДЕЛИРОВАНИЯ ТРЕБОВАНИЙ К СИСТЕМЕ

5.3. СРЕДСТВА ЯЗЫКА SYSML ДЛЯ МОДЕЛИРОВАНИЯ СТРУКТУРЫ СИСТЕМЫ

5.4. СРЕДСТВА ЯЗЫКА SYSML ДЛЯ МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ СИСТЕМЫ

5.5. СРЕДСТВА ЯЗЫКА SYSML ДЛЯ ПОСТРОЕНИЯ МОДЕЛИ

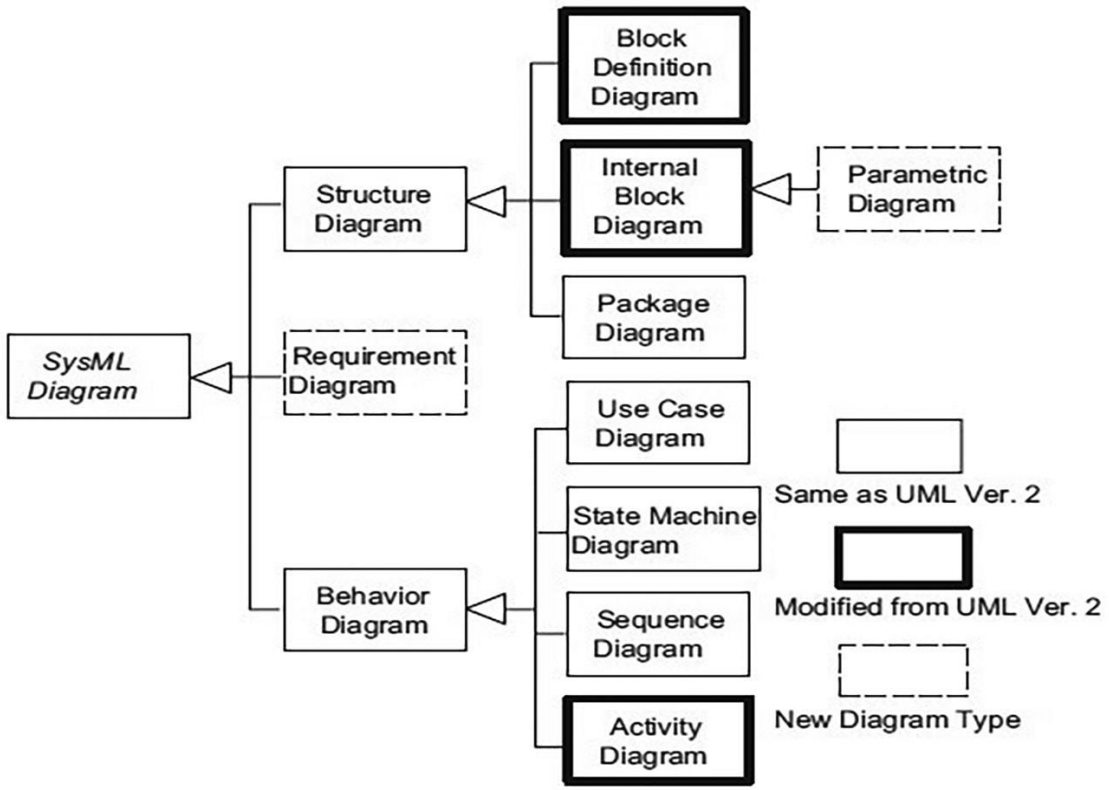


Fig. B.2 SysML Diagrams

ГЛАВА 6. ИНЖЕНЕРИЯ ТРЕБОВАНИЙ

- **6.1.** ИНЖИНИРИНГ ТРЕБОВАНИЙ. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ
- **6.2.** ТРЕБОВАНИЯ В ЖИЗНЕННОМ ЦИКЛЕ СИСТЕМ
- **6.3.** ПРЕОБРАЗОВАНИЕ ПОТРЕБНОСТЕЙ ПРЕДПРИЯТИЯ В ТРЕБОВАНИЯ
- **6.4.** КОНСТРУКЦИЯ ТРЕБОВАНИЙ
- **6.5.** ПРОЦЕССЫ И ДЕЯТЕЛЬНОСТИ ИНЖЕНЕРИИ ТРЕБОВАНИЙ В ЖИЗНЕННОМ ЦИКЛЕ СИСТЕМЫ
- **6.6.** ПОТРЕБНОСТИ И ТРЕБОВАНИЯ БЕЗОПАСНОСТИ
- **6.7. ПРИМЕНЕНИЕ ЯЗЫКА SYSML ДЛЯ МОДЕЛИРОВАНИЯ ТРЕБОВАНИЙ**

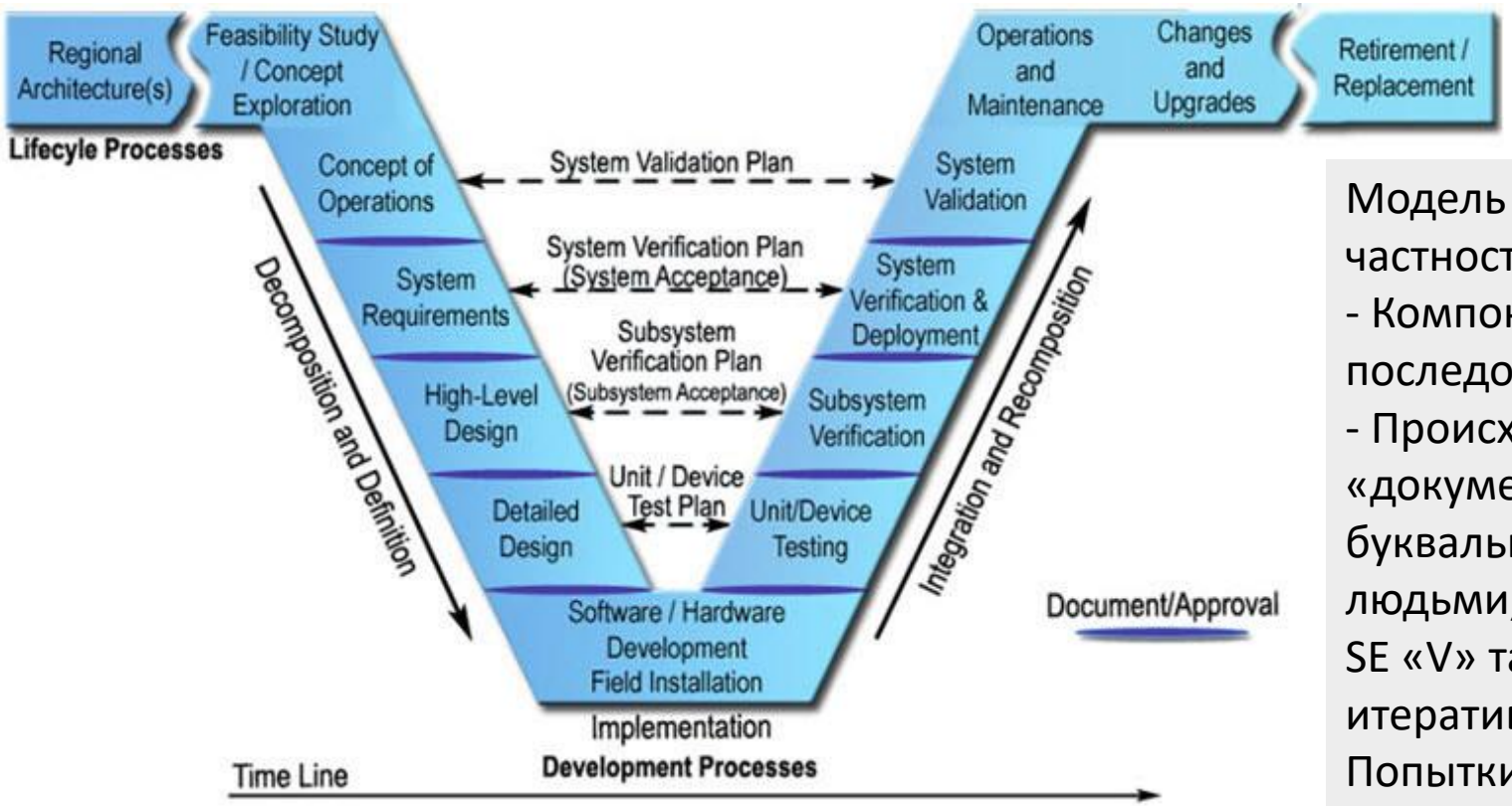
ГЛАВА 7. MBSE — СИСТЕМНАЯ ИНЖЕНЕРИЯ НА ОСНОВЕ МОДЕЛЕЙ

- **7.1.** ВВЕДЕНИЕ В MBSE. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ
- **7.2.** МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА SE И MBSE. V-МОДЕЛЬ В СРЕДЕ, ОСНОВАННОЙ НА МОДЕЛЯХ, РОМБ SE И ЦИФРОВОЙ ДВОЙНИК
- **7.3.** СТРУКТУРА ДОМЕНОВ ДЛЯ МОДЕЛИ АРХИТЕКТУРЫ СИСТЕМ
- **7.4.** АРХИТЕКТУРНО-АКЦЕНТИРОВАННЫЙ ПРОЦЕСС MBSE (MBSAP THE MODEL-BASED SYSTEM ARCHITECTURE PROCESS — MBSAP) РАЗРАБОТКИ МОДЕЛЕЙ СИСТЕМ

Введение

- **Системная инженерия на основе моделей [MBSE]** — это парадигма, которая использует формализованные представления систем, известные как модели, для поддержки и облегчения выполнения задач системной инженерии (SE) на протяжении всего жизненного цикла системы [1].
- Международный совет по системной инженерии (INCOSE) определяет **MBSE** как реализация подхода SE, на основе принципов цифрового моделирования на уровне определения системы и имитации физического и рабочего поведения на протяжении всего жизненного цикла системы [2].
- Модели SE обычно выражаются на стандартизированном языке моделирования таком, как, например, язык моделирования систем **SysML** [3], они выражают ключевую информацию о системе в кратком, последовательном, правильном и последовательном формате
- При правильном применении модели MBSE обеспечивают стандартизированную консолидацию и интеграцию системных знаний между инженерными дисциплинами и подсистемами и оптимизируют ключевые задачи системного проектирования, а также минимизируют риски разработки
- Понятия модели, свойства модели, характеристики системной модели

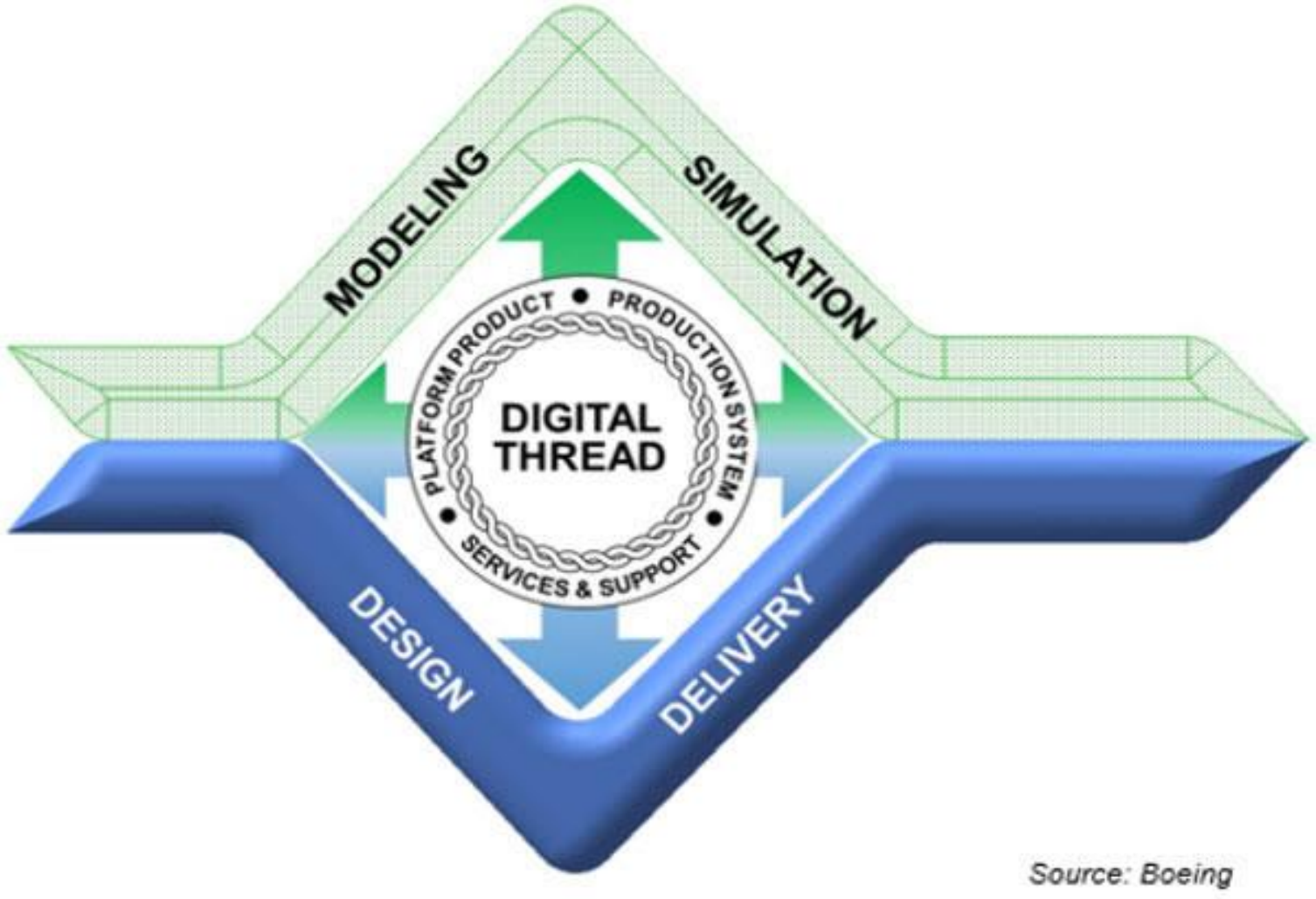
Традиционная модель “V” системной инженерии [5].



SOURCE: US Department of Transportation Federal Highway Administration
https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm

Модель SE «V» обладает рядом недостатков, в частности [5]:
- Компоновка «V» предполагает только последовательный процесс разработки
- Происхождение SE «V» относится к эпохе «документоцентризма», когда информация буквально передавалась между отдельными людьми, группами и фазами жизненного цикла
SE «V» также не отражает интегрированный и итеративный характер разработки продукта
Попытки обновить символ «V» только усложнили его и сделали его еще более трудным для понимания
Последовательность V-процесса затрудняет обмен данными и информацией в реальном времени между сложными взаимодействиями экосистемы MBE

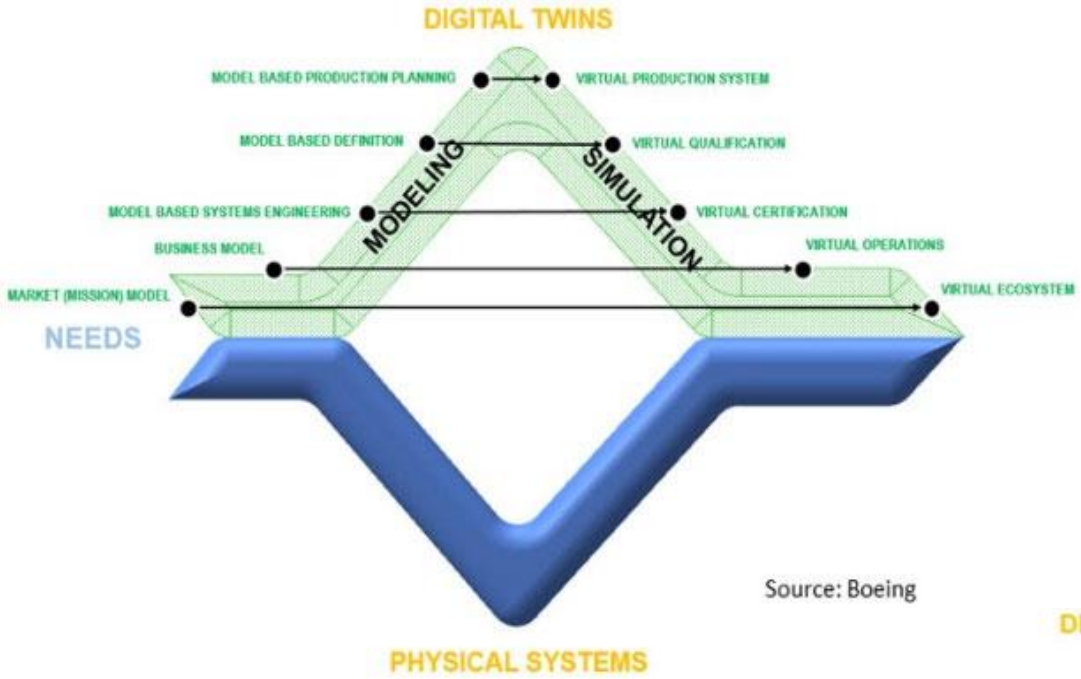
Ромб SE (SE diamond)



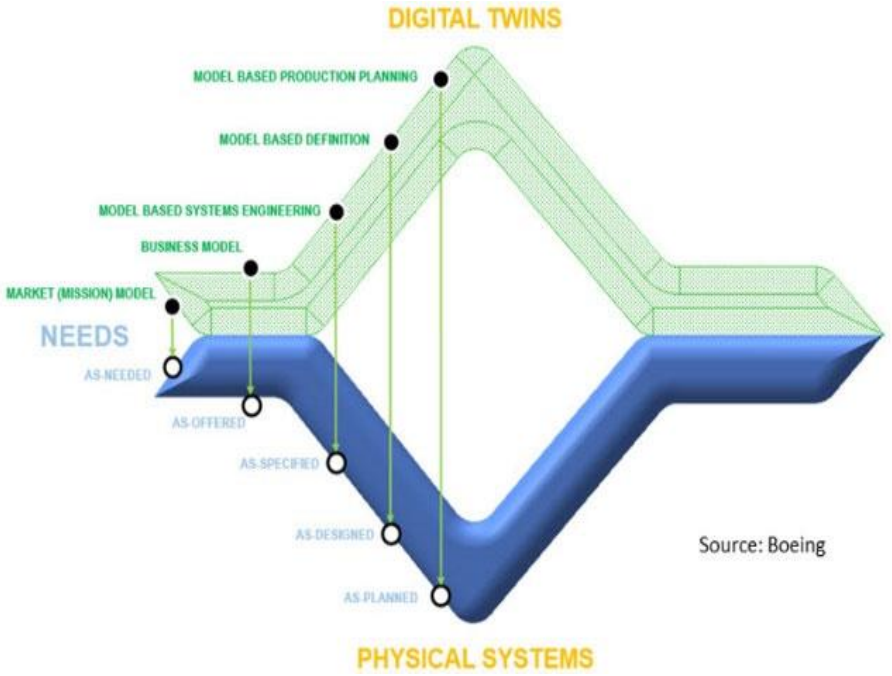
Source: Boeing

MBSE предполагает переход от мышления, ориентированного на документы, к мышлению цифрового проектирования, которое использует **Цифровой поток (Digital thread)** на протяжении всего жизненного цикла. Одной из наиболее важных задач является разработка физических и виртуальных представлений продуктов и услуг. Это само по себе приводит к циклу разработки виртуального продукта, который можно рассматривать как «зеркальное отражение» цикла разработки физического продукта. Компания Boeing разработала подход, иллюстрируемый на следующем слайде, демонстрирующий принципы использования конфигураций цифровых двойников **«как спроектировано»**, **«как построено»**, **«как поставлено»** как самостоятельных продуктов.

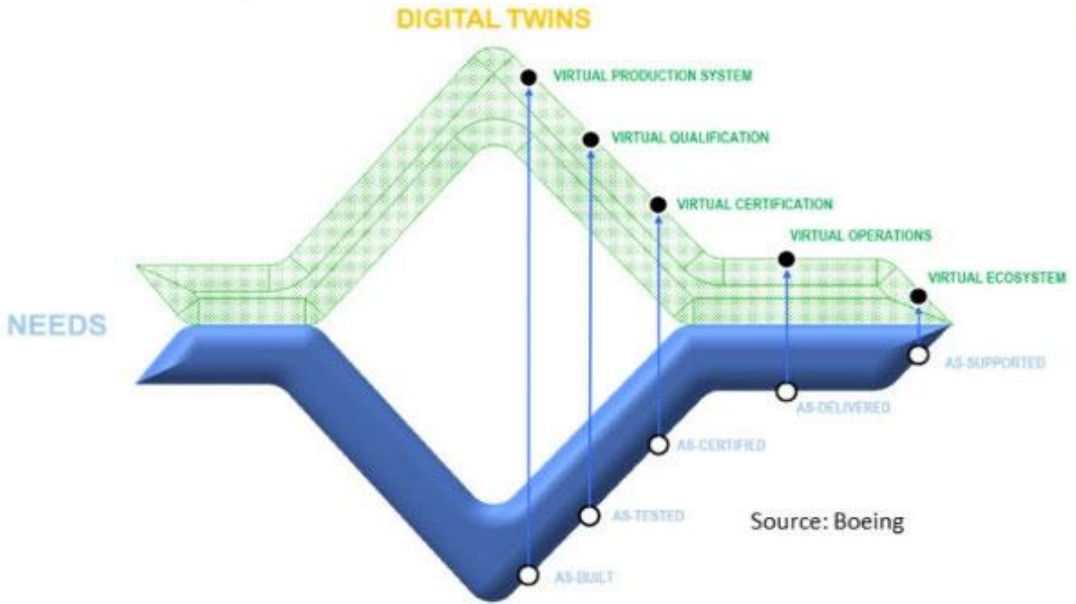
Цифровой поток



Source: Boeing

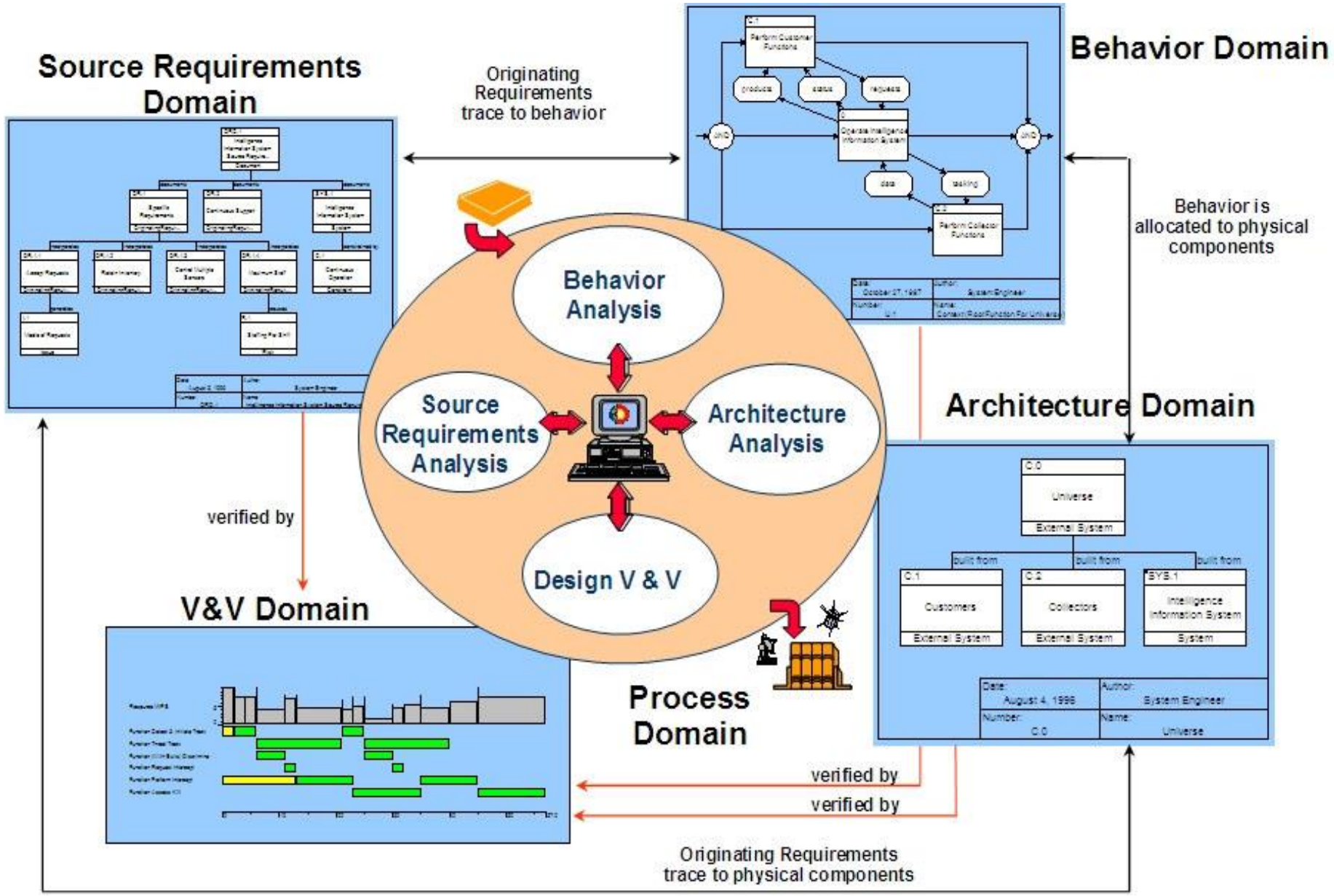


Source: Boeing



Source: Boeing

Домены создания модели



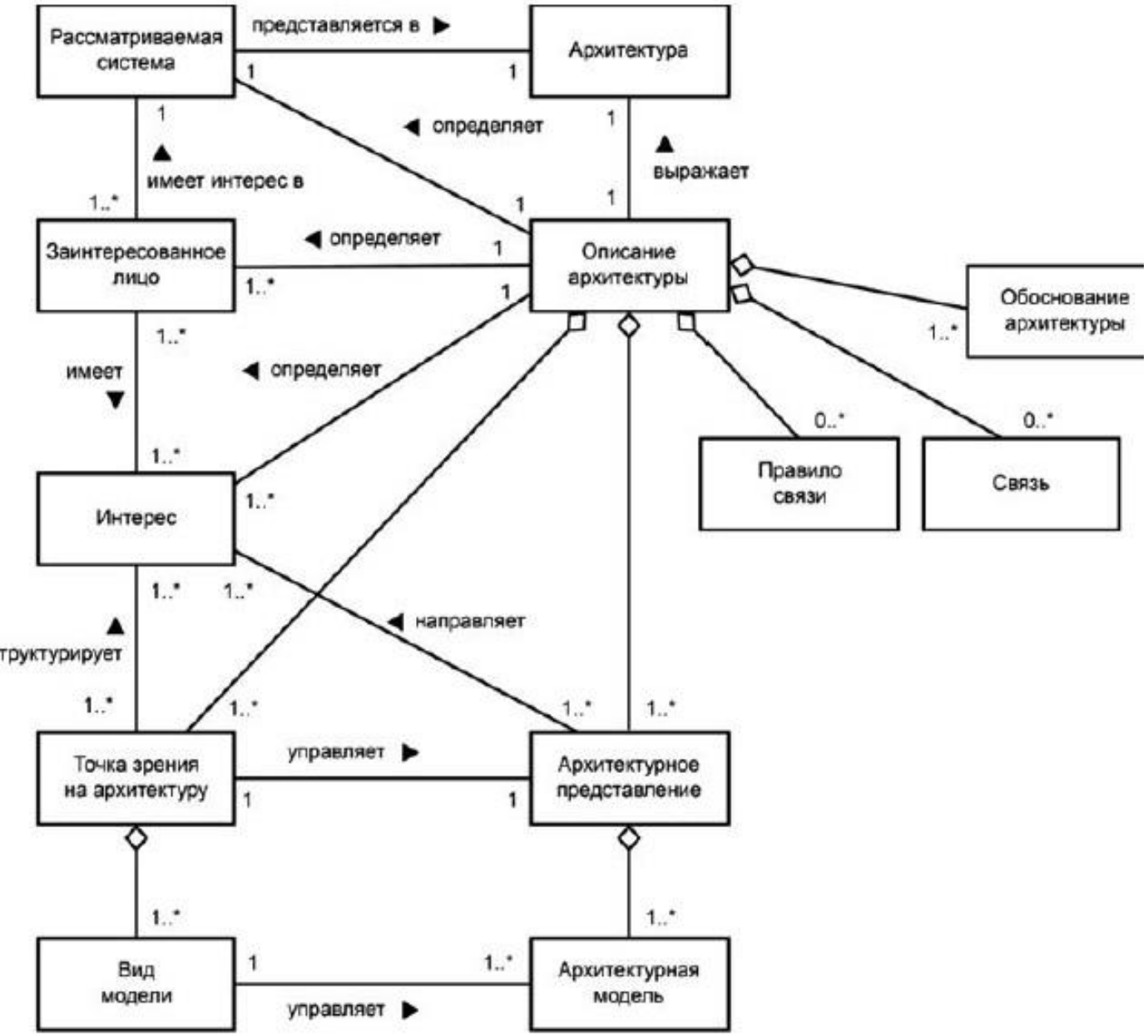
ГЛАВА 8. ОПИСАНИЕ АРХИТЕКТУРЫ СИСТЕМЫ

8.1. ОПИСАНИЕ АРХИТЕКТУРЫ (ГОСТ Р 57100-2016/ISO/IEC/IEEE 42010)

8.2. МОДЕЛЬ ЗАХМАНА ОПИСАНИЯ АРХИТЕКТУРЫ ПРЕДПРИЯТИЯ

8.3. ПРИМЕР ОПИСАНИЯ АРХИТЕКТУРЫ ДЛЯ ПРИЛОЖЕНИЙ ИНТЕРНЕТА ВЕЩЕЙ (IoT)

Концептуальная модель описания архитектуры



Архитектуризация системной архитектуры выражается в **описании архитектуры** и помогает понять сущность системы и ключевые свойства, определяющие ее поведение, состав и эволюцию, в свою очередь, влияющие на такие факторы, как целесообразность, полезность и управляемость системы

Именно этой теме посвящен международный стандарт **ИСО 42010**, направленный на создание **описаний архитектуры** при разработке, анализе и поддержке архитектуры системы

Описание архитектуры используется для документирования архитектуры интересующей (целевой) системы

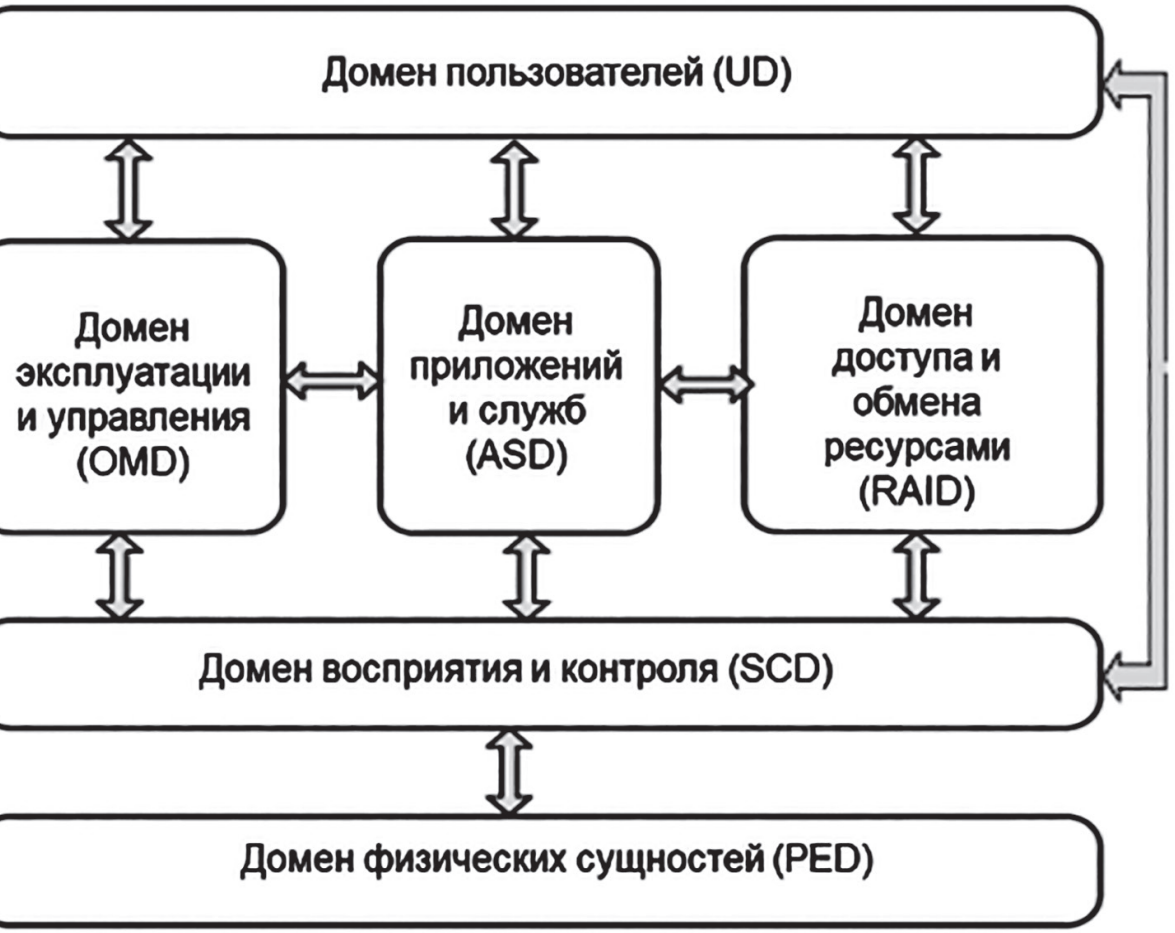
Описание архитектуры является **рабочим продуктом**, результатом проектирования архитектуры системы

Модель Захмана описания архитектуры

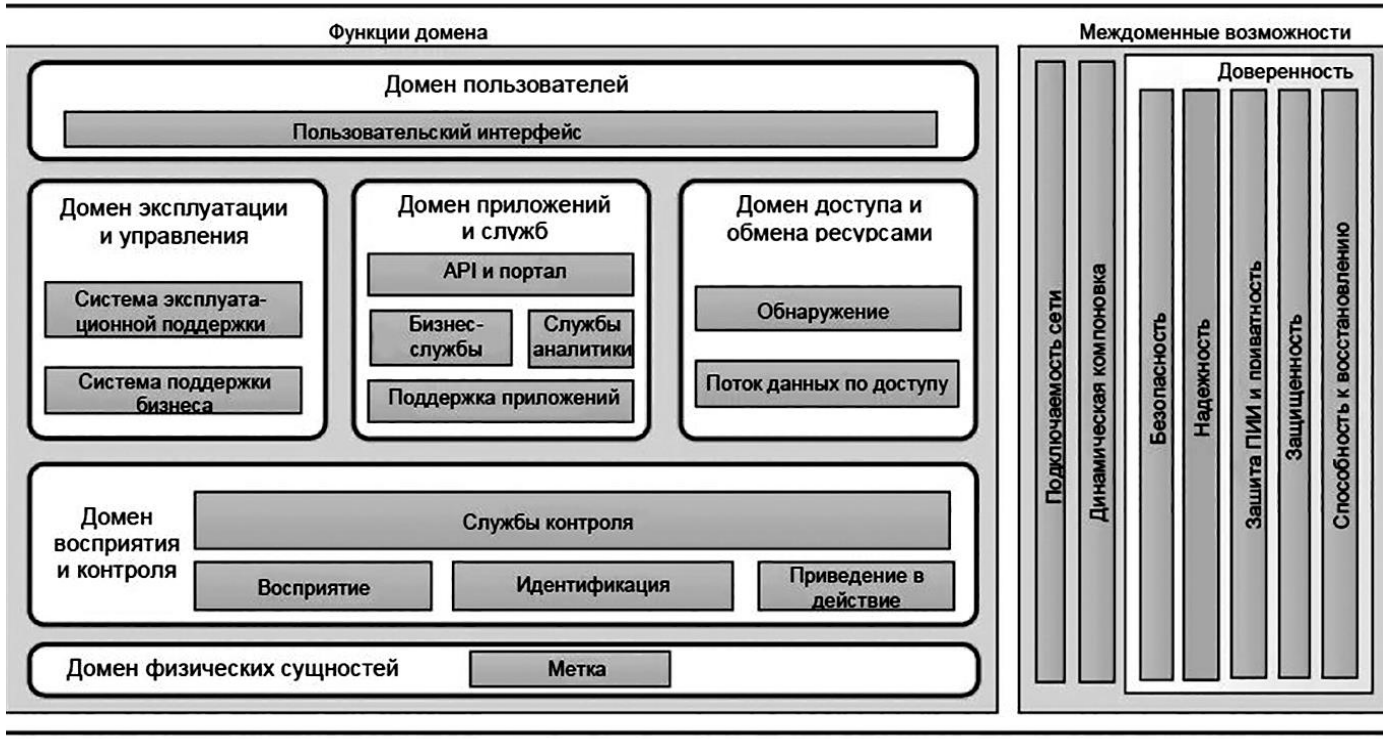
Интернет Вещей/ Типовая архитектура IoT

стандарт определяет:
– общие ожидаемые характеристики систем IoT;
– концептуальную модель, определяющую ключевые понятия системы IoT;
– типовую модель, определяющую общую структуру элементов архитектуры;
– набор соответствующих архитектурных представлений с нескольких точек зрения

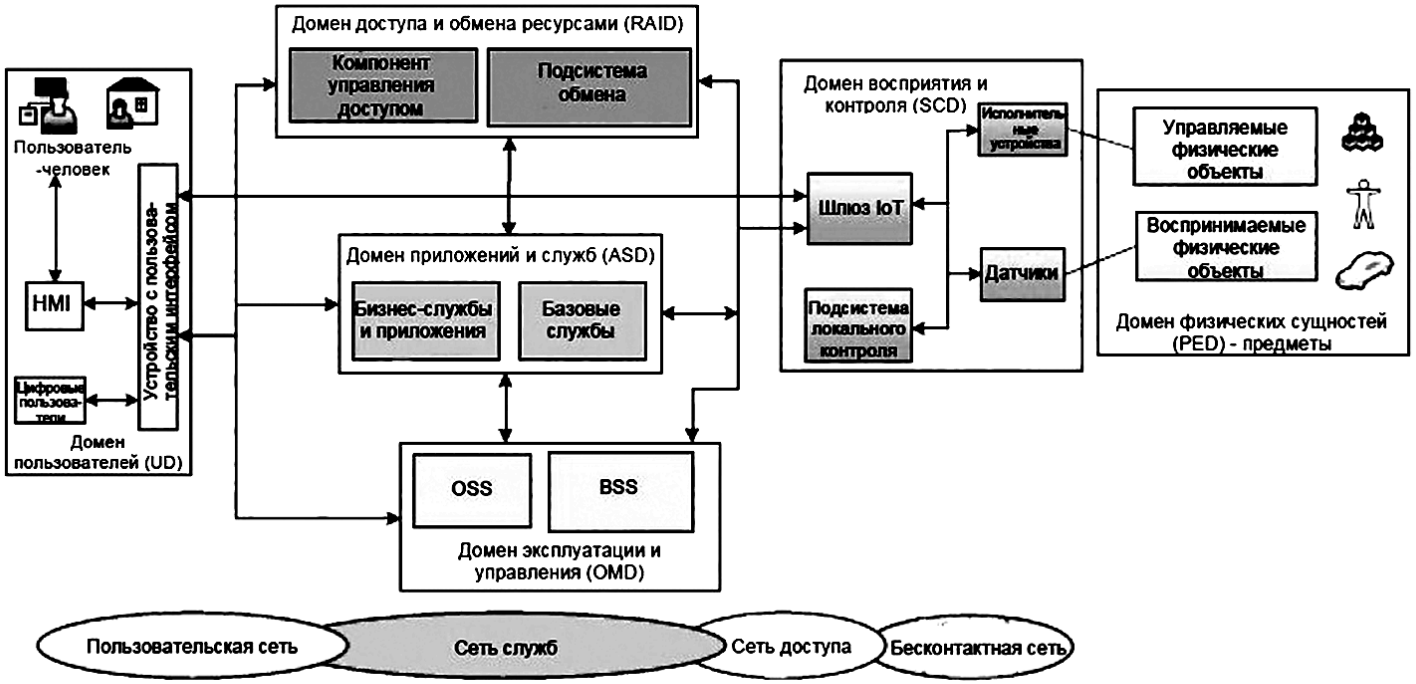
Типовая модель основе доменов



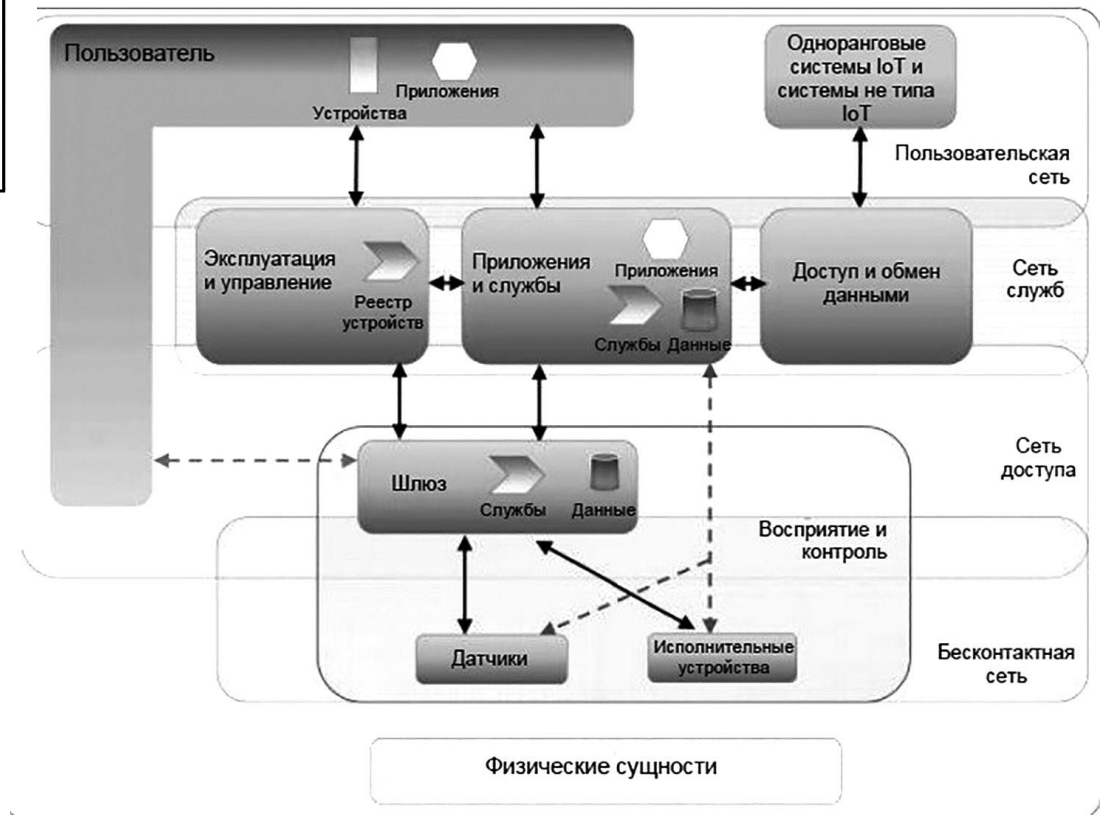
Функциональное представление модели IoT



Представление развертывания системы IoT



Сетевое представление модели IoT



ГЛАВА 9. ЦИФРОВЫЕ ДВОЙНИКИ

9.1. КРАТКАЯ ИСТОРИЯ И ОПРЕДЕЛЕНИЯ ЦИФРОВЫХ ДВОЙНИКОВ

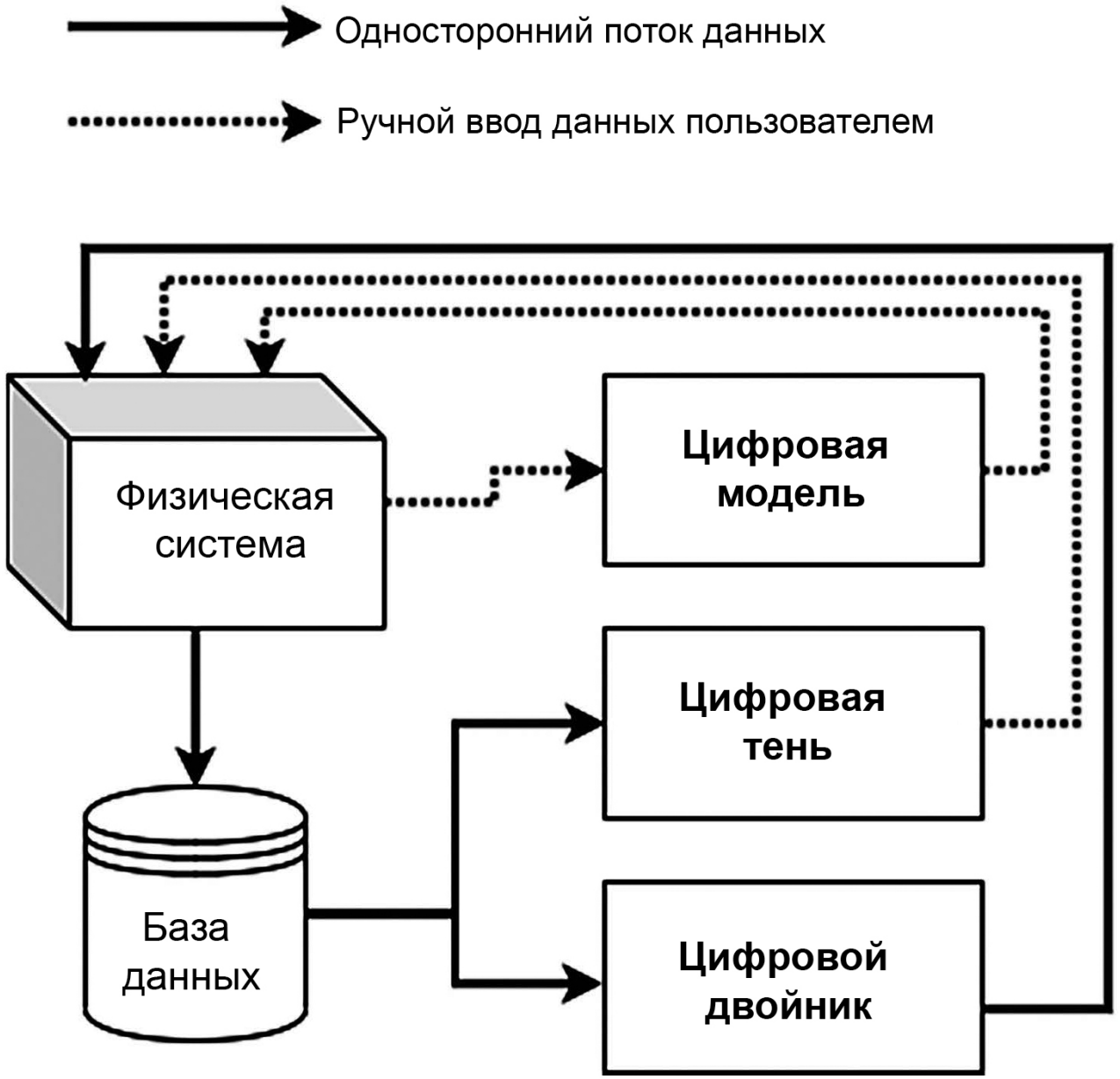
9.2. АНАЛИЗ ПРОЦЕССА СТАНДАРТИЗАЦИИ В ОБЛАСТИ ЦИФРОВЫХ ДВОЙНИКОВ

9.3. АНАЛИЗ АРХИТЕКТУРНЫХ РЕШЕНИЙ ДЛЯ ЦИФРОВЫХ ДВОЙНИКОВ

9.4. ТЕХНОЛОГИЯ ЦИФРОВЫХ ДВОЙНИКОВ И MBSE

9.5. ЦИФРОВЫЕ ДВОЙНИКИ ДЛЯ СОЗДАНИЯ ПРОМЫШЛЕННОЙ МЕТАВСЕЛЕННОЙ

ЦИФРОВЫЕ ДВОЙНИКИ



В литературе различаются три уровня виртуального представления цифрового двойника [34]:

- «цифровая модель» (digital model — DM)
- «цифровая тень» (digital shadow — DS);
- «цифровой двойник» (digital twin — DT).

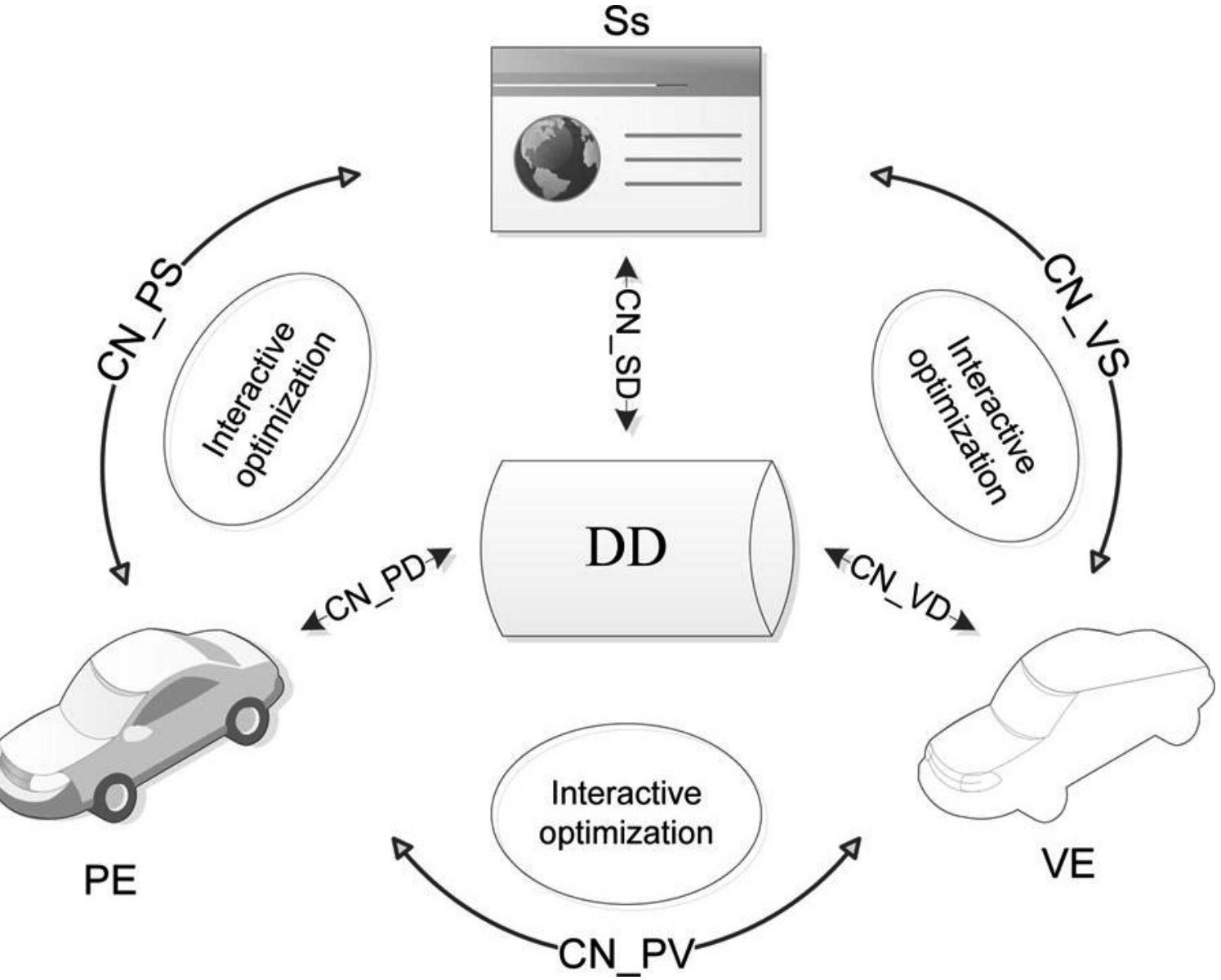
СТАНДАРТИЗАЦИЯ ЦИФРОВЫХ ДВОЙНИКОВ

Функциональное представление архитектуры цифрового двойника производства

Шестимерный цифровой двойник цеха

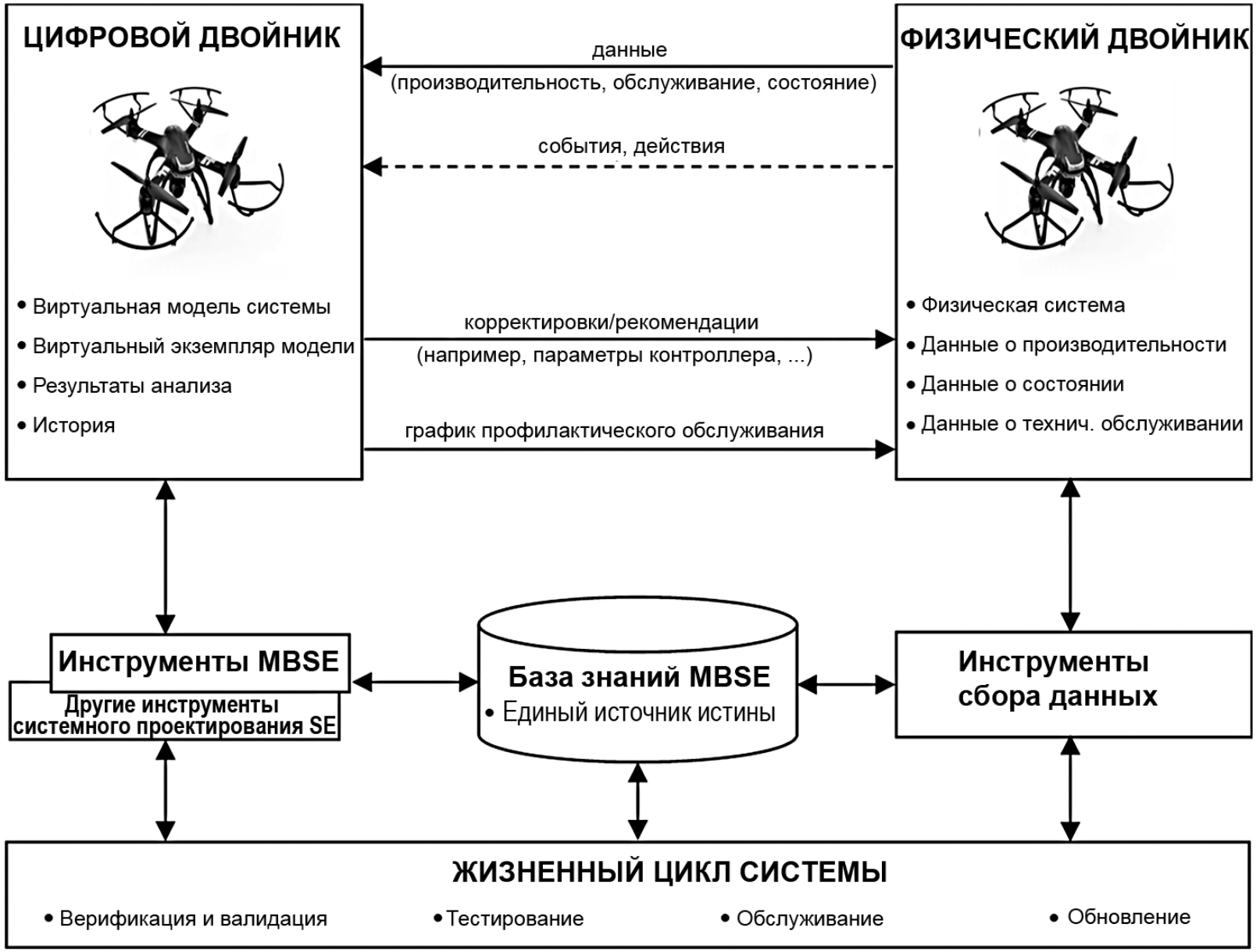
Цеховой цифровой двойник в интеллектуальном производстве

Пятимерная модель цифрового двойника

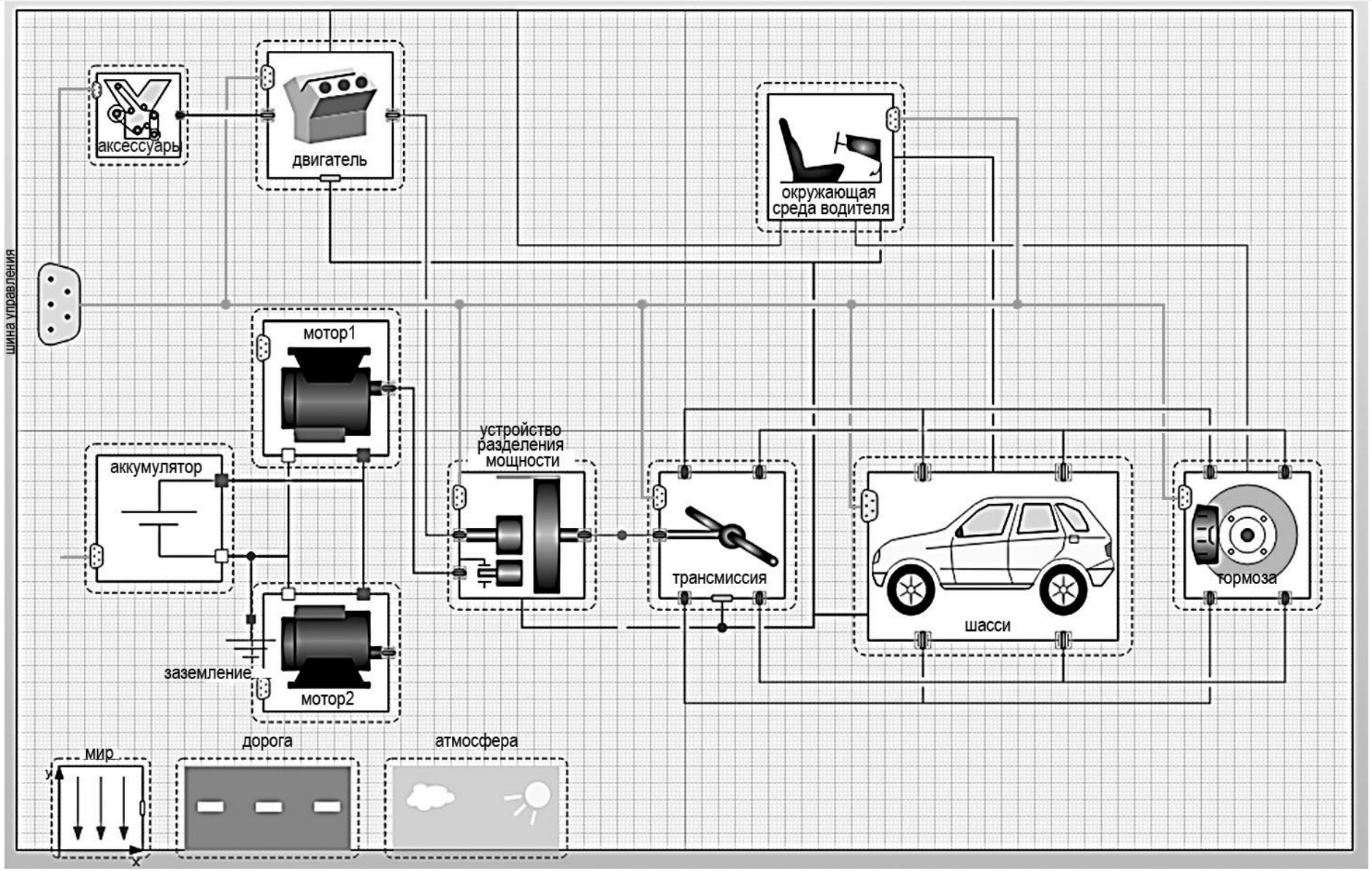


включает следующие измерения (модули):
- физический объект (PE)
- виртуальный объект (VE)
- модуль услуг (Ss)
- модуль данных (DD)
- модуль связи (CN)

Концепция цифрового двойника в рамках MBSE

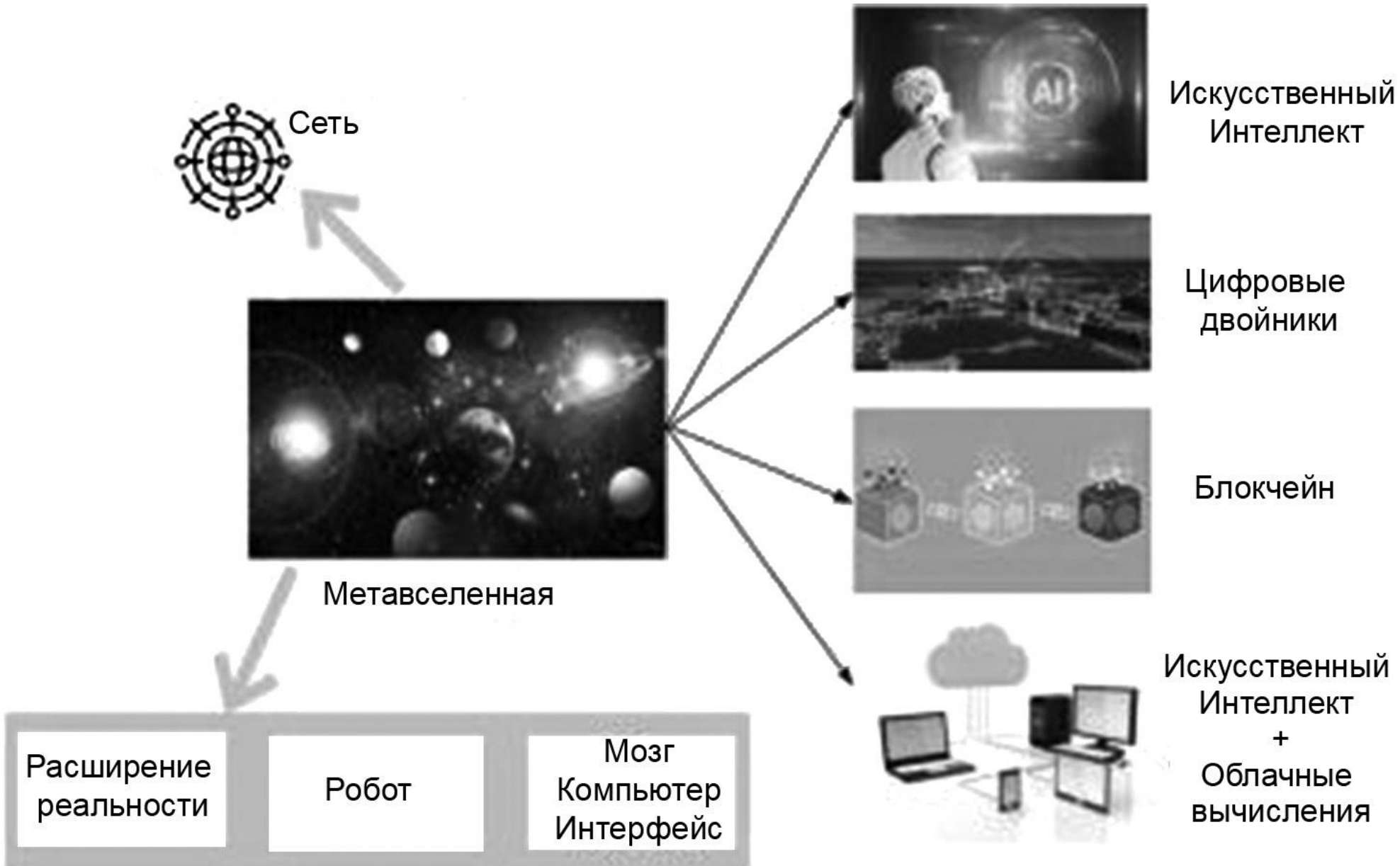


ИНТЕЛЛЕКТУАЛЬНЫЕ ЦИФРОВЫЕ ДВОЙНИКИ

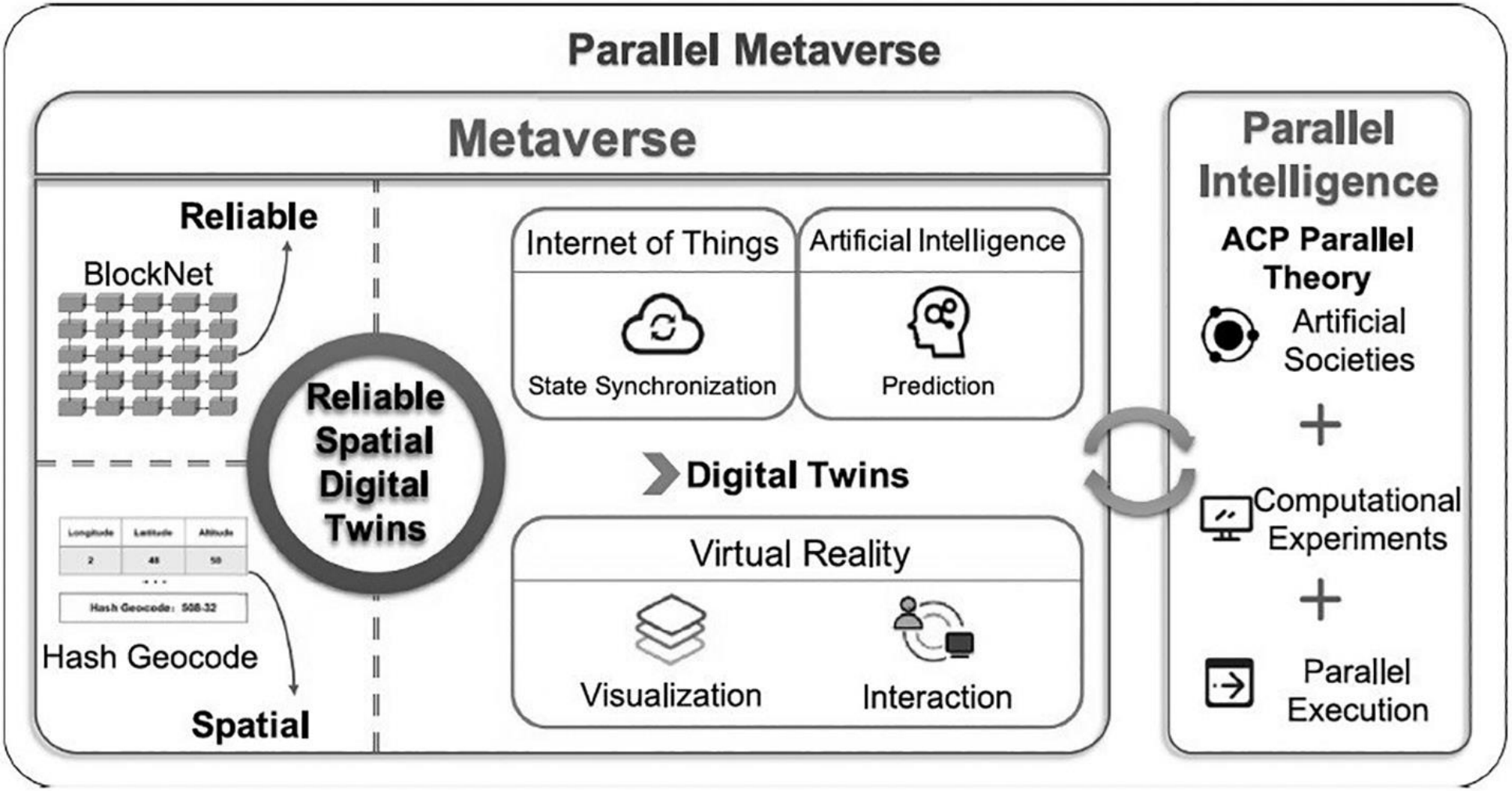


Анализ тенденций развития цифровых двойников нового поколения
В.А. Сухомлин, Д.Е. Намиот, Д.А. Гапанович
International Journal of Open Information Technologies ISSN: 2307-8162 vol. 12, no. 7, 2024

ИНТЕЛЛЕКТУАЛЬНЫЕ ЦИФРОВЫЕ ДВОЙНИКИ



ИНТЕЛЛЕКТУАЛЬНЫЕ ЦИФРОВЫЕ ДВОЙНИКИ



Каждая метавселенная строится из надежных пространственных цифровых двойников (spatial digital twin), использующих критически важные технологии блокчейна (BlockNet) для безопасного хранения многомерных данных и безопасности процесса цифрового картографирования метавселенной

ГЛАВА 10. СИСТЕМА СТАНДАРТОВ СИСТЕМНОЙ ИНЖЕНЕРИИ. ПРОЦЕССНЫЕ СТАНДАРТЫ

10.1. ТАКСОНОМИЯ БАЗОВЫХ СТАНДАРТОВ СИСТЕМНОЙ ИНЖЕНЕРИИ

10.2. ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА СИСТЕМ — ISO/IEC/IEEE 15288

10.3. ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ISO/IEC 12207)

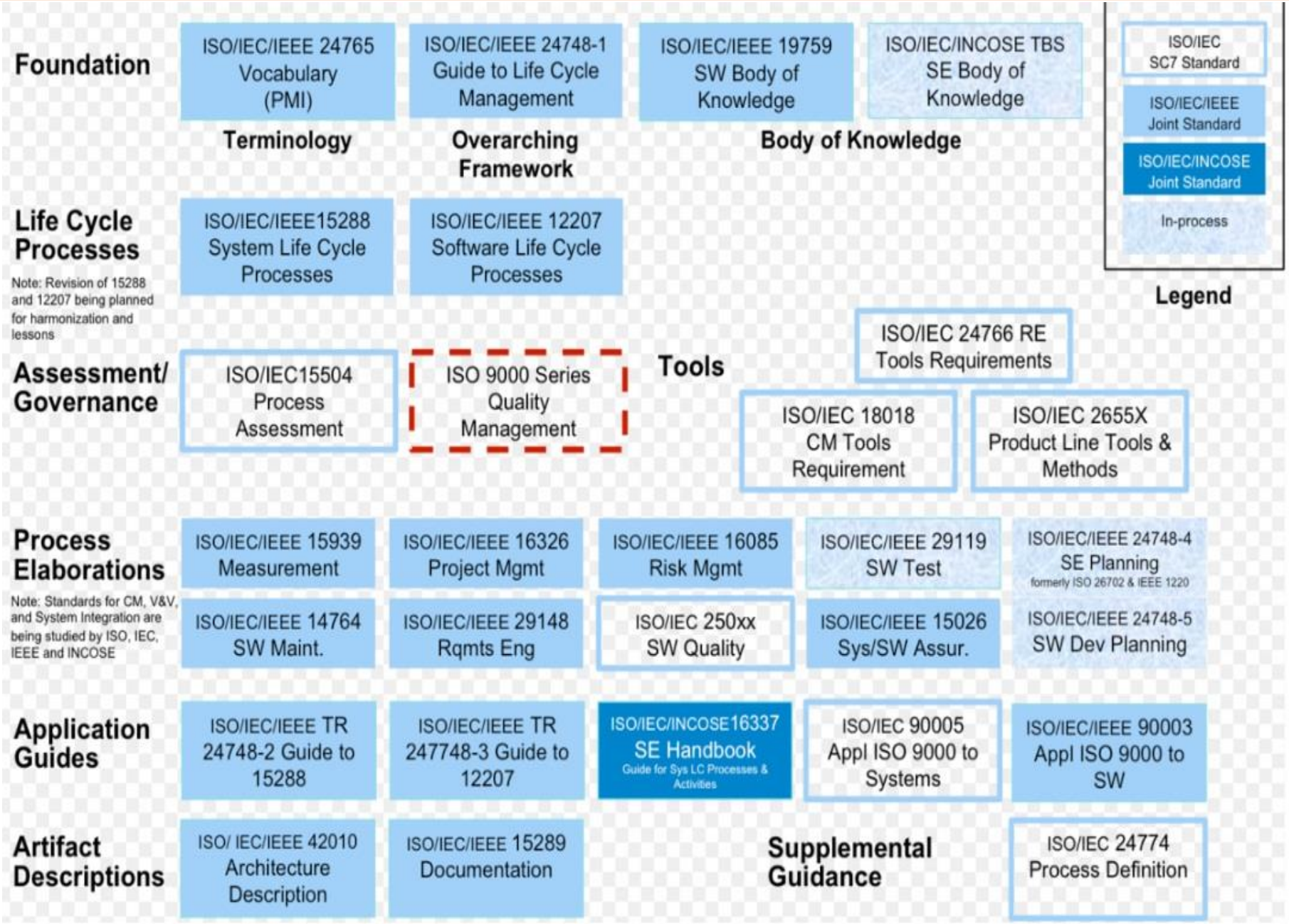
10.4. КАТЕГОРИЯ ПРОЦЕССОВ В КОНТЕКСТЕ СИСТЕМЫ

10.5. КАТЕГОРИЯ СПЕЦИАЛЬНЫХ ПРОЦЕССОВ ПРОГРАММНЫХ СРЕДСТВ)

10.6. СРАВНЕНИЕ ISO/IEC/IEEE 15288:2023 С ISO/IEC/IEEE 15288:2015

10.7. МОДЕЛЬНО-ОРИЕНТИРОВАННАЯ СИСТЕМНАЯ И ПРОГРАММНАЯ ИНЖЕНЕРИЯ (MODEL-BASED SYSTEMS AND SOFTWARE ENGINEERING (MBSSE)»

Таксономия базовых стандартов SE



Таксономия базовых стандартов SE

Первый сверху уровень охватывает **Основы (Foundation).** Его составляют следующие стандарты:

- **ISO/IEC FDIS 24765** – словарь терминов СИ, в основу которого положен глоссарий терминов в области программной инженерии IEEE 610.12-1990
- **ISO/IEC TR 24748-1** Systems and software engineering. Guide to Life Cycle Management (Разработка систем программного обеспечения. Менеджмент жизненного цикла. Часть 1. Руководство по менеджменту жизненного цикла) - Содержит описание концепций ЖЦ, а также целей и результатов типовых стадий ЖЦ
- **SWEBOK** (Software Engineering Body of Knowledge) — **ISO/IEC TR 19759:2015** – стандарт, содержащий описание свода профессиональных знаний в области программной инженерии
- **SEBoK** (Guide to the Systems Engineering Body of Knowledge) - свод профессиональных знаний в области программной инженерии

Второй уровень Life Cycle Processes (Процессы жизненного цикла) включают два особенно важных стандарта:

- **ISO/IEC/IEEE 15288:2015** «Systems and software engineering — System life cycle processes» (ГОСТ Р ИСО/МЭК 15288-2005 Информационная технология (ИТ). Системная инженерия. Процессы жизненного цикла систем)
- **ISO/IEC/IEEE 12207** «Systems and software engineering — System life cycle software» (ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология (ИТ). Системная и программная инженерия. Процессы жизненного цикла программных средств)

Таксономия базовых стандартов SE

Третий уровень сверху **Assesment/Governance (Оценка/Управление)** содержит следующие стандарты:

- **ISO/IEC 15504** Информационные технологии — Оценка процесса, также именуется. Software Process Improvement and Capability Determination, SPICE (заменен на **ISO/IEC 33001:** 2015 - ГОСТ Р ИСО/МЭК 33001 2017 ОЦЕНКА ПРОЦЕССА. Понятия и терминология
- **ISO 9000** Series Quality Management (серия международных стандартов, содержащих термины и определения, основные принципы менеджмента качества, требования к системе менеджмента качества) **- ГОСТ Р ИСО 9000-2015 -** СИСТЕМЫ МЕНЕДЖМЕНТА КАЧЕСТВА
- Еще группе стандартов, определяющих требования к средствам (TOOLS) и принципам их использования в жизненном цикле систем

Четвертый уровень Process Elaborations (Разработка процесса) сверху содержит ряд стандартов, содержащих рекомендации по реализации основных процессов жизненного цикла систем. К ним относятся стандарты:

- **ISO/IEC FDIS 16326** – Системная и программная инженерия. Процессы ЖЦ. Управление проектами
- **ISO/IEC 16085** – Системная и программная инженерия. Процессы ЖЦ. Управление рисками. - Определяет процесс управления рисками в ЖЦ. - Может использоваться самостоятельно или в качестве дополнения к процессам ЖЦ, определенным в ISO/IEC 15288 и ISO/IEC 12207
- **ISO/IEC 15939** – Системная и программная инженерия. Процесс измерения. - Определяет процесс измерения, пригодный для использования в области системной и программной инженерии, а также в области менеджмента

Таксономия базовых стандартов SE

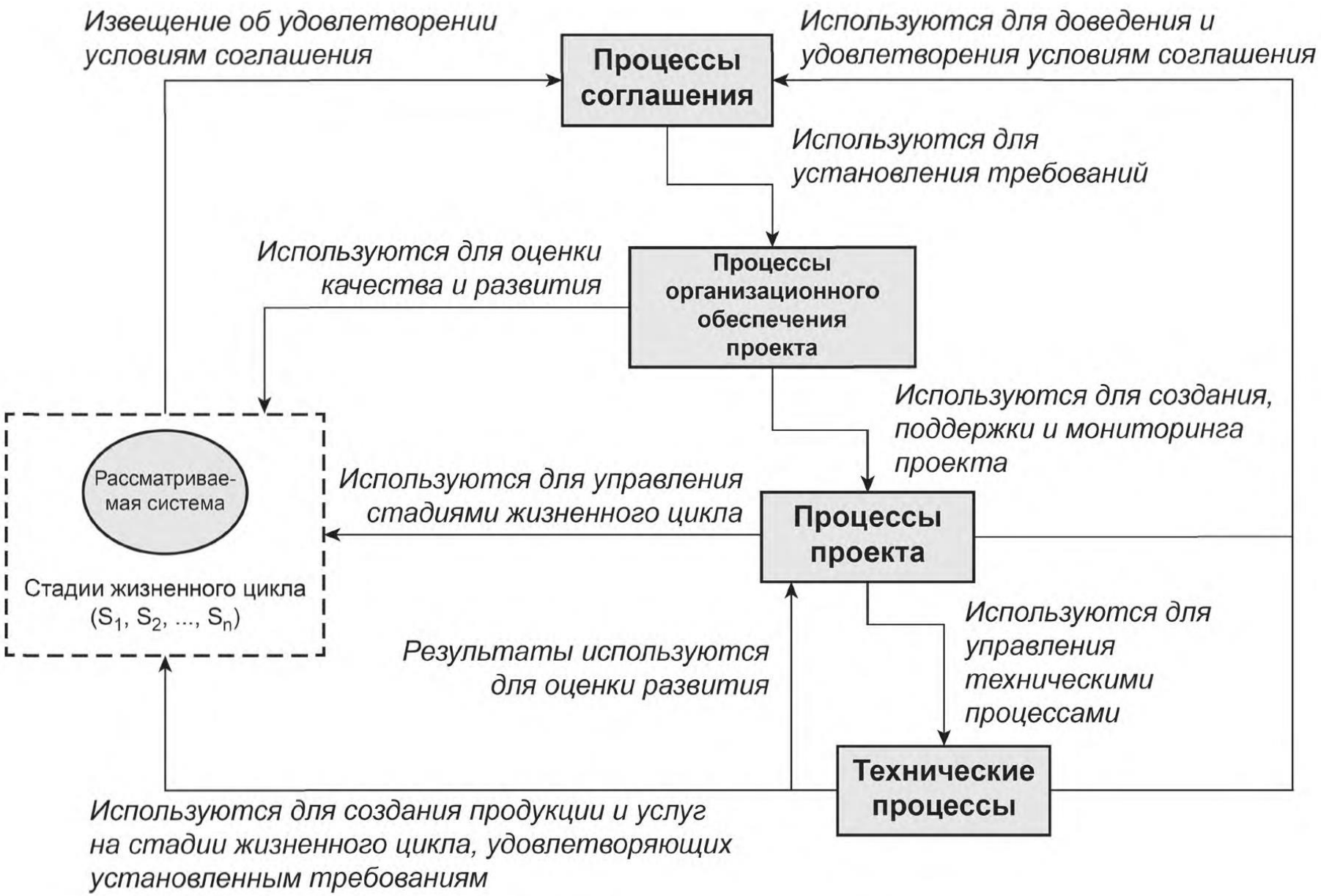
- **ISO/IEC 14764** – Системная инженерия. Процессы ЖЦ ПС. Сопровождение. - Описывает организацию процесса сопровождения, определенного в ISO/IEC 12207
- **ISO/IEC TR 90005** – Системная инженерия. Рекомендации по применению ISO 9001 к процессам ЖЦ систем. Содержит рекомендации по применению организациями ISO 9001 при закупке, поставке, разработке, применении и сопровождении систем

Пятый уровень Application guides (Рекомендации по применению) включает руководства по применению основных стандартов системной инженерии: **ISO/IEC TR 24748-2** – для ISO 15288**, ISO/IEC TR 24748-3** – для ISO 12207, **ISO/IEC TR 90005** -для ISO 9001 применительно к процессам ЖЦ систем, ISO/IEC TR 90005 -для ISO 9001 применительно к процессам программных систем, а также **ISO/IEC TR 16337** – важнейший методологический документ, содержащий описание ключевых процессов и активностей, выполняемых разработчиками систем и представляющий собой руководство системного инженера (Systems Engineering Handbook. A GUIDE FOR SYSTEM LIFE CYCLE PRO CESSES AND ACTIVITIES FOURTH EDITION INCOSE-TP-2003-002-04 2015)

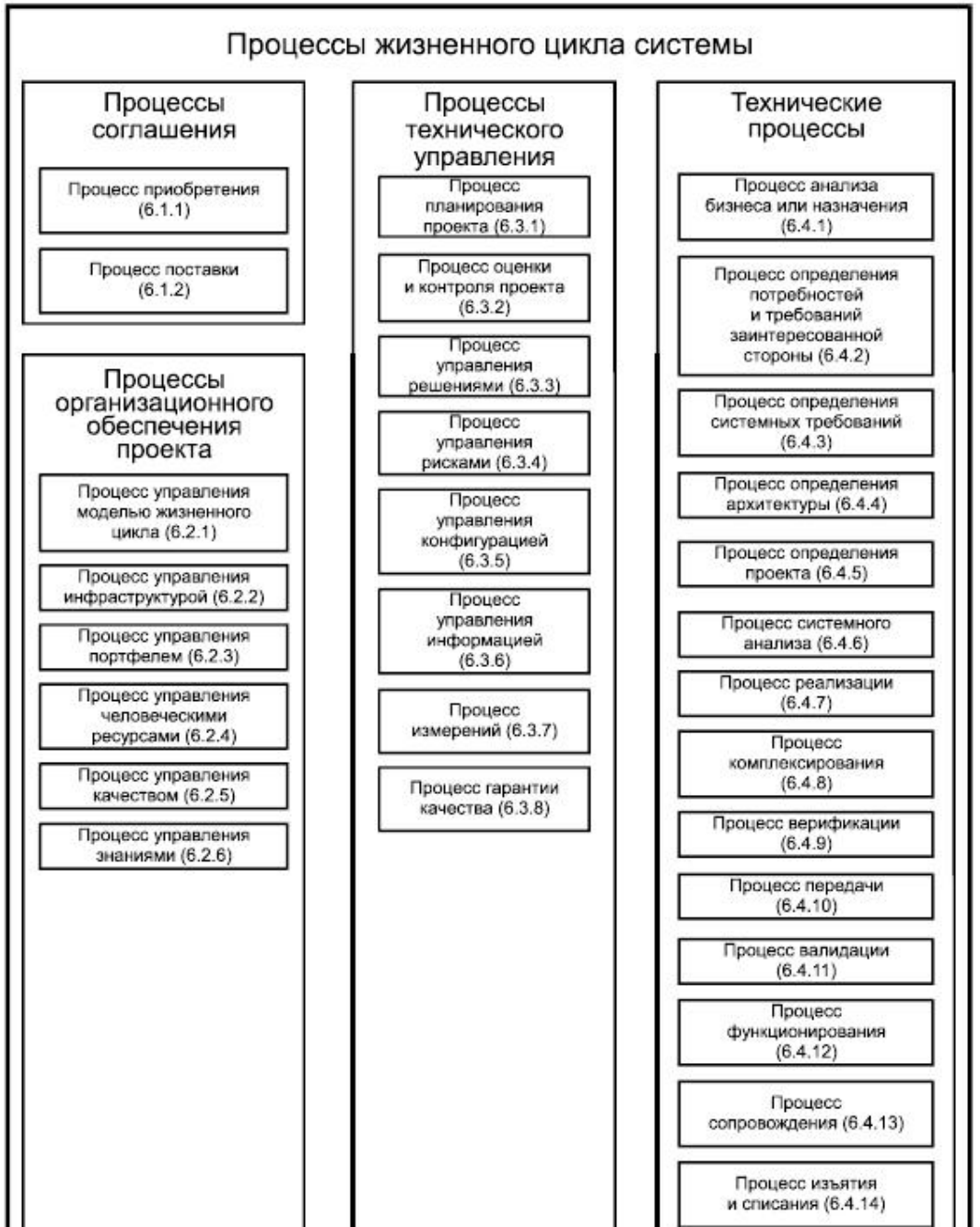
Шестой уровень - самый нижний, **Artifact Description (Описание артефактов)**. Он содержит следующие руководства:

- **ISO/IEC TR 24774** - Руководящие указания по описанию процесса - Руководство по описанию процессов ЖЦ систем
- **ISO/IEC TR 15289** – рекомендации к разработке основных типов документации, используемых на протяжении жизненного цикла информационных систем и программного обеспечения.
- **ISO/IEC TR 42010** - руководство к разработке описания архитектуры, как важнейшего артефакта проекта

Понятия процесса



Процессы жизненного цикла системы



ГЛАВА 11. ЭТАЛОННАЯ МОДЕЛЬ МОДЕЛЬНО-ОРИЕНТИРОВАННОЙ СИСТЕМНОЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ (MBSSE) И ЕЕ СВЯЗЬ С ПРОЦЕССНЫМИ СТАНДАРТАМИ СИСТЕМНОЙ ИНЖЕНЕРИИ

11.1. МЕТОДЫ И ИНСТРУМЕНТЫ ДЛЯ МОДЕЛЬНО-ОРИЕНТИРОВАННОЙ СИСТЕМНОЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ (ISO/IEC/IEEE 24641:2023)

11.2. ОСНОВНЫЕ ПОНЯТИЯ СТАНДАРТА ISO/IEC/IEEE 24641:2023

11.3. ЭТАЛОННАЯ МОДЕЛЬ MBSSE

11.4. ОПИСАНИЕ ЦЕЛЕЙ ПРОЦЕССОВ ЭТАЛОННОЙ МОДЕЛИ

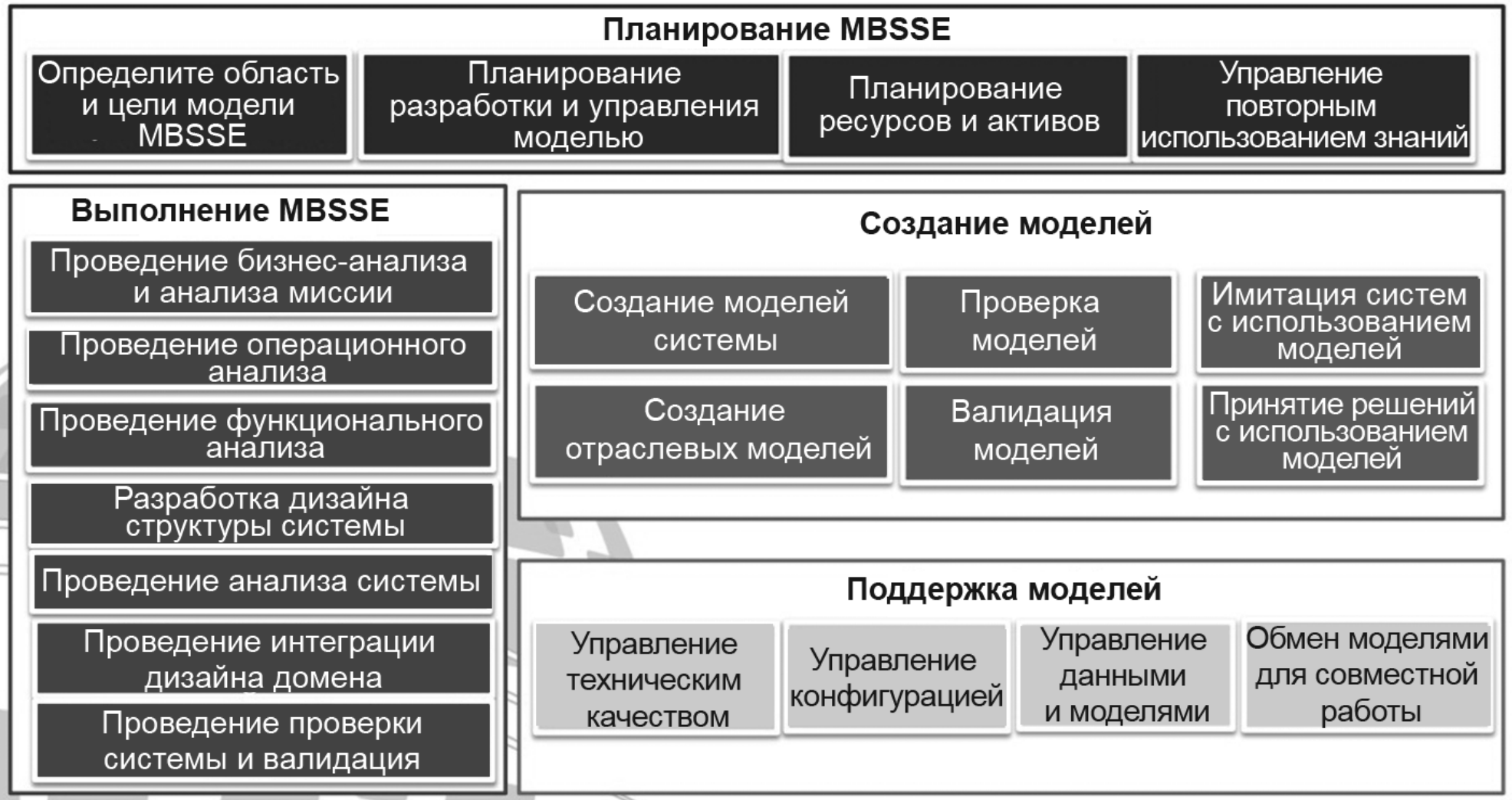
11.5. ВЗАИМОСВЯЗЬ ISO/IEC/IEEE 24641 С ГЛАВНЫМИ МЕЖДУНАРОДНЫМИ СТАНДАРТАМИ СИСТЕМНОЙ ИНЖЕНЕРИИ

11.6. ТРЕХМЕРНАЯ ЭТАЛОННАЯ СТРУКТУРА (ФРЕЙМВОРК) MBSSE

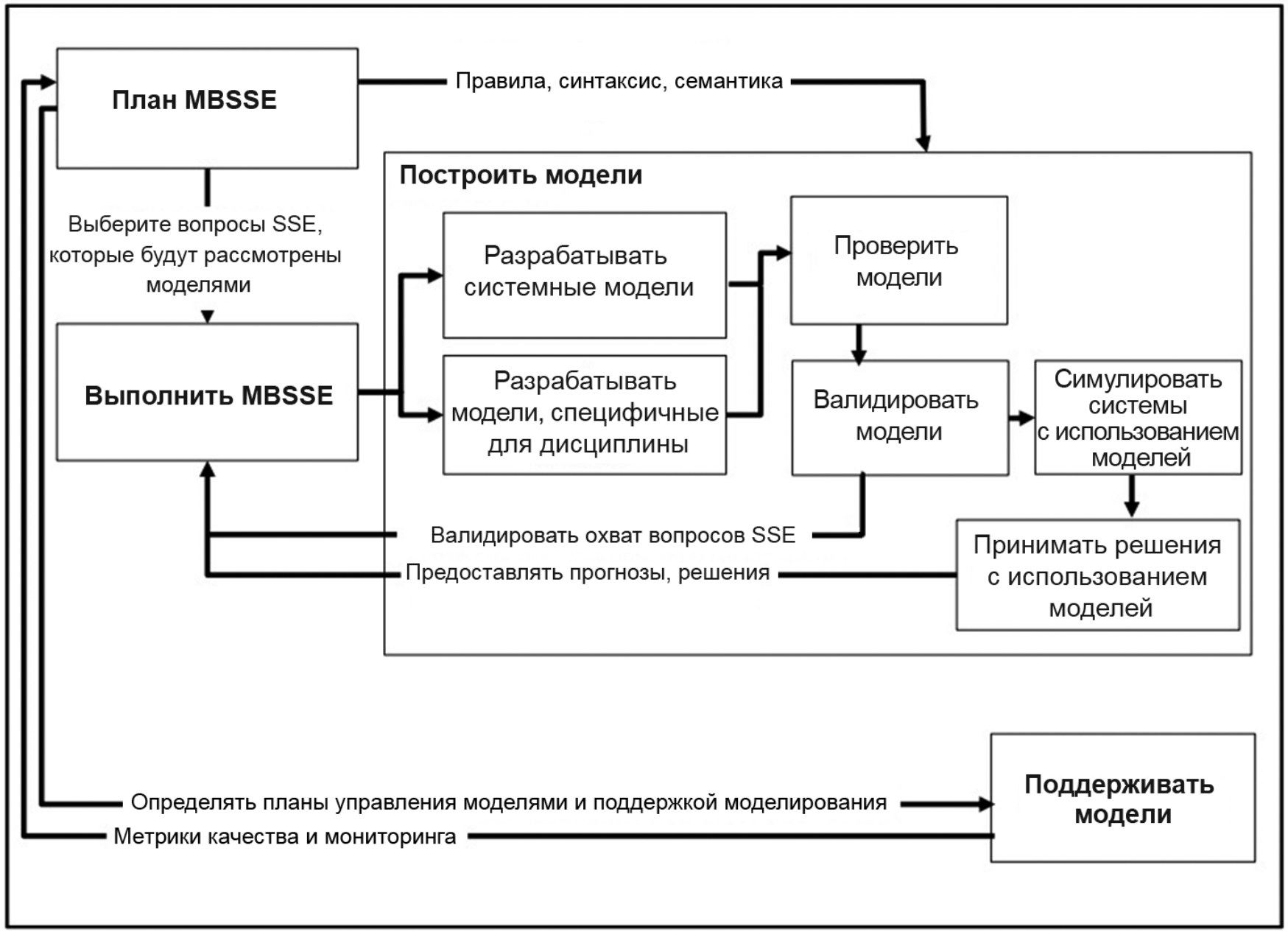
11.7. MBSSE-РАЗМЕРНОСТИ СИСТЕМНОЙ МОДЕЛИ

11.8. ПОТЕНЦИАЛЬНО ВОЗМОЖНЫЕ MBSSE-РОЛИ

Эталонная модель MBSSE



Связь процессов внутри эталонной модели



Процессы построения моделей и пирамида «Данные — Информация — Знания — Мудрость» (DIKW)

Процессы построения моделей и пирамида «Данные — Информация — Знания — Мудрость» (DIKW) [ISO/IEC/IEEE FDIS 24641:2022(E). Systems and software engineering — Methods and tools for model-based systems and software engineering.]

ГЛАВА 12. ИНТЕГРАЦИЯ СИСТЕМЫ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. МЕТОДИЧЕСКИЕ АСПЕКТЫ

12.1. НАЗНАЧЕНИЕ СТАНДАРТА ISO/IEC/IEEE FDIS 24748-6 ИНТЕГРАЦИЯ СИСТЕМ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (SYSTEM AND SOFTWARE INTEGRATION)

12.2. КОНЦЕПЦИЯ СИСТЕМНОЙ ИНТЕГРАЦИИ И ЕЕ ОСНОВНЫЕ ПОНЯТИЯ

12.3. ПЛАНИРОВАНИЕ И ЦЕЛИ ПРИМЕНЕНИЯ ПРОЦЕССА ИНТЕГРАЦИИ

12.4. ДРУГИЕ ПРОЦЕССЫ, СВЯЗАННЫЕ С ПРОЦЕССОМ ИНТЕГРАЦИИ

12.5. ТРЕБОВАНИЯ К ИНФОРМАЦИОННЫМ ЭЛЕМЕНТАМ

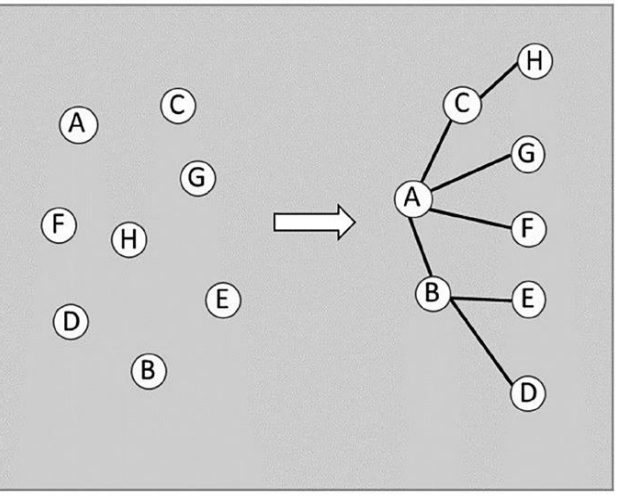
12.6. МАТРИЦЫ СВЯЗИ (COUPLING MATRIXES)

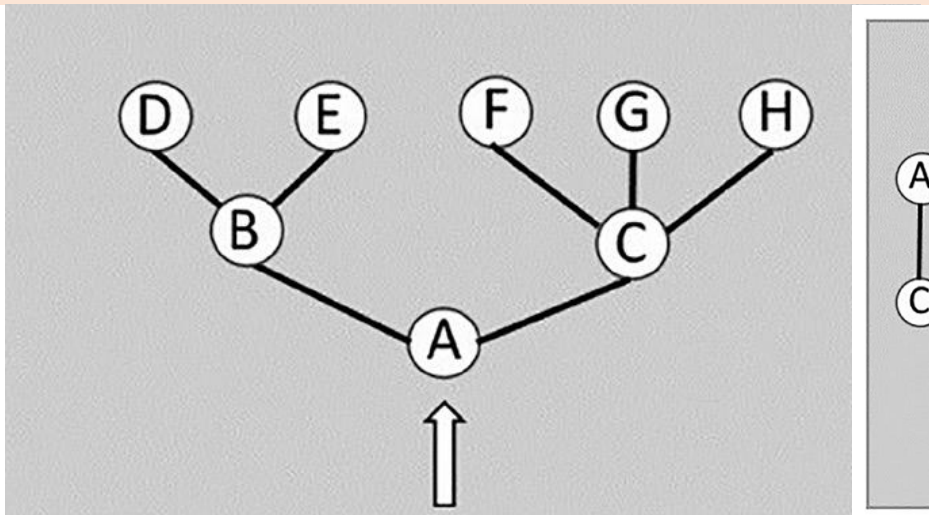
ГЛАВА 12. ИНТЕГРАЦИЯ СИСТЕМЫ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. МЕТОДИЧЕСКИЕ АСПЕКТЫ

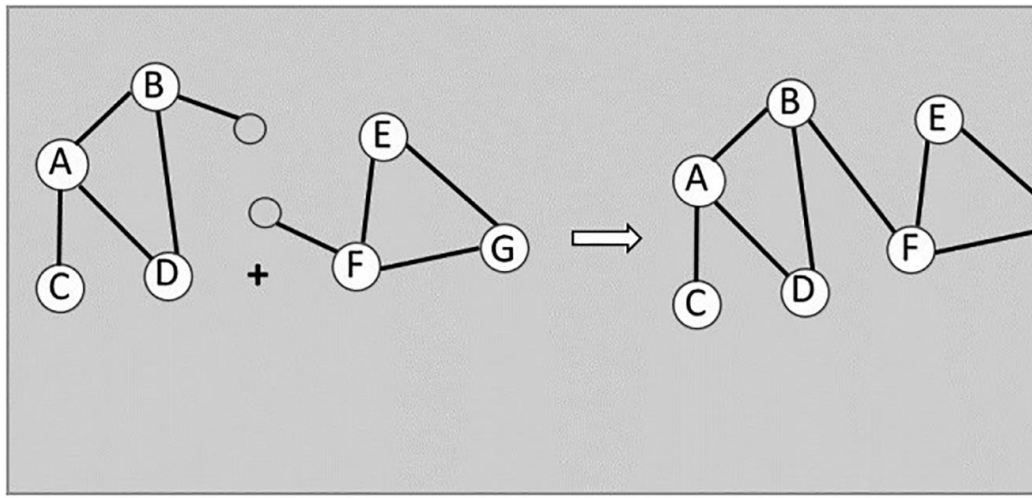
НАЗНАЧЕНИЕ СТАНДАРТА ISO/IEC/IEEE FDIS 24748-6 ИНТЕГРАЦИЯ СИСТЕМ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (SYSTEM AND SOFTWARE INTEGRATION

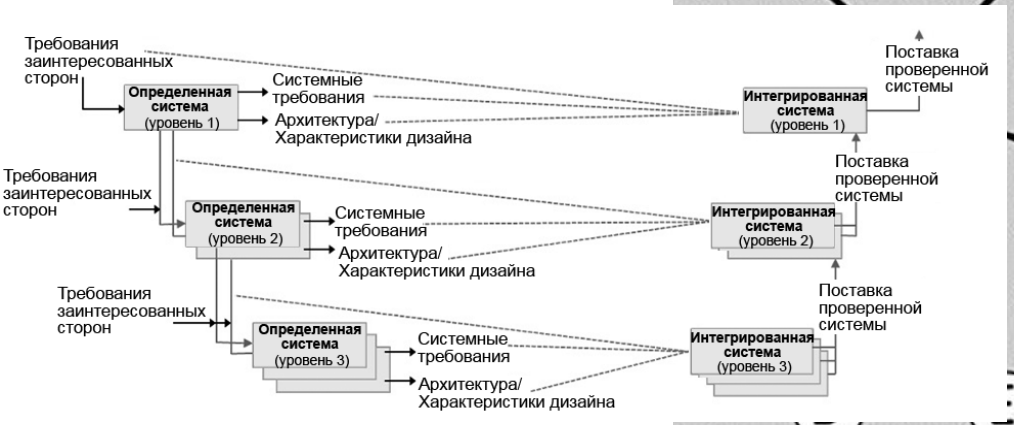
- **Понятия агрегации и синтеза**
- Интеграция системы основана на понятиях агрегации и синтеза. Например, крылья, корпус, двигатели, средства управления аппаратным и программным обеспечением, процедуры воздушного движения и информационные элементы объединяются вместе с другими элементами, не перечисленными здесь, для синтеза воздушного судна, работающего в предполагаемой среде. Синтез выполняет функции, невозможные для отдельных элементов, которые агрегируются.
- Агрегат имеет функциональную согласованность, которая позволяет выполнять последующие деятельности по проверке и, возможно, испытаниям. Каждый агрегат характеризуется конфигурацией, которая определяет реализованные элементы системы или программного обеспечения, которые агрегируются, а также статус их конфигурации. Агрегация может выполняться итеративно или рекурсивно.
- **Понятие интеграции**
- Интеграция включает в себя планирование и агрегирование все более полного набора элементов и активацию их интерфейсов для синтеза части системы, которая может быть проверена и, возможно, валидирована в соответствии со стратегическими решениями по реализации процесса валидации.
- Интеграция обеспечивает взаимодействие между некоторым набором элементов для удовлетворения их требований и обеспечивает основу для интеграции еще более полной части системы. Конечным результатом является вся система или система программного обеспечения, которая интегрирована внутри себя и в своих интерфейсах для целей и использования системы.
- Интеграция системы может включать сочетание аппаратного обеспечения, программного обеспечения, данных, людей, процессов (например, процессов предоставления услуг пользователям), процедур (например, инструкций оператора), средств, материалов и природных объектов, из которых состоит система.

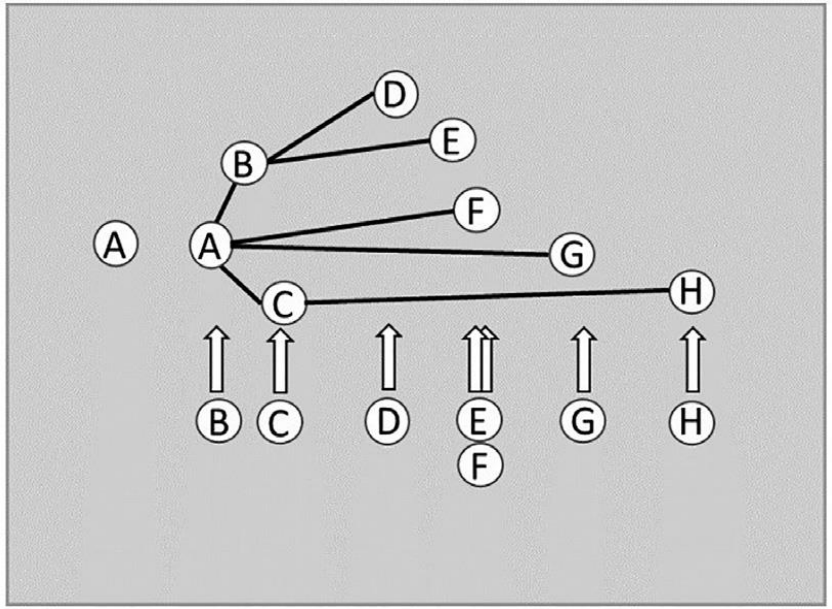
Методы интеграции











ГЛАВА 13. МАТЕМАТИЧЕСКИЕ ОСНОВЫ СИСТЕМНОЙ ИНЖЕНЕРИИ

13.1. ИСТОРИЯ СИСТЕМНОЙ ИНЖЕНЕРИИ

13.2. КОТИЛЕДОНЫ УАЙМОРА (WYMORE) И СИСТЕМНАЯ ИНЖЕНЕРИЯ

13.3. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ СИСТЕМЫ ПО УАЙМОРУ

13.4. МАТЕМАТИЧЕСКАЯ ПЛАТФОРМА УАЙМОРА И КОНЕЧНЫЕ АВТОМАТЫ ЯЗЫКА SYSML

13.5. ТЕОРИЯ КАТЕГОРИЙ КАК ФОРМАЛЬНАЯ МАТЕМАТИЧЕСКАЯ ОСНОВА ДЛЯ СИСТЕМНОГО ПРОЕКТИРОВАНИЯ НА ОСНОВЕ МОДЕЛЕЙ

13.6. ОБЪЕДИНЕНИЕ ТЕОРЕТИЧЕСКОЙ ОСНОВЫ СИСТЕМ УАЙМОРА И ФОРМАЛИЗМА МОДЕЛИРОВАНИЯ DEVS: К НАУЧНЫМ ОСНОВАМ ДЛЯ MBSE