ОБЕСПЕЧЕНИЕ КАЧЕСТВА И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекции – 24 часа

Форма аттестации - экзамен

Д.т.н., профессор

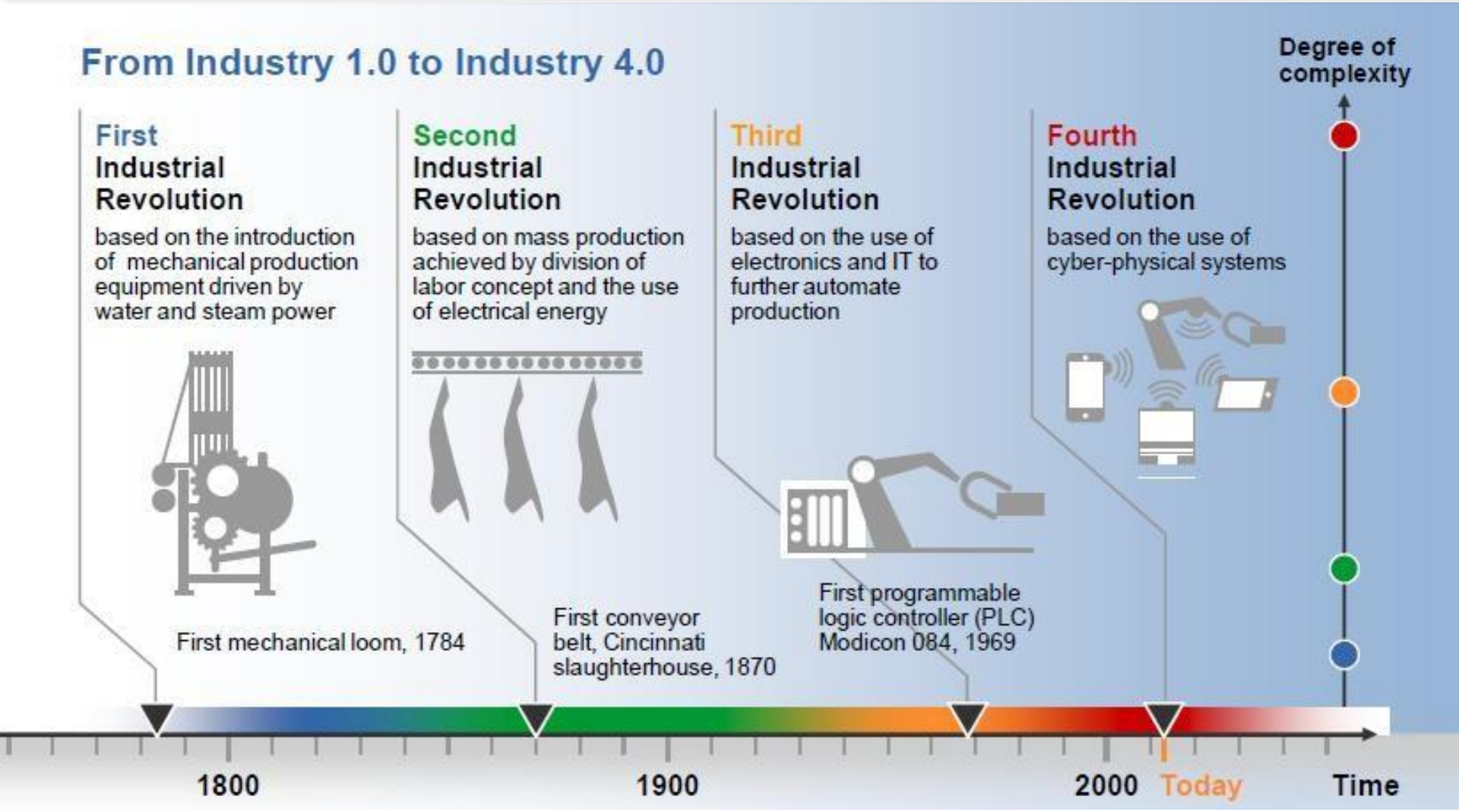
Гвоздев Владимир Ефимович

Ауд. 6-212

Слияние виртуального и реального миров с образованием гибридного

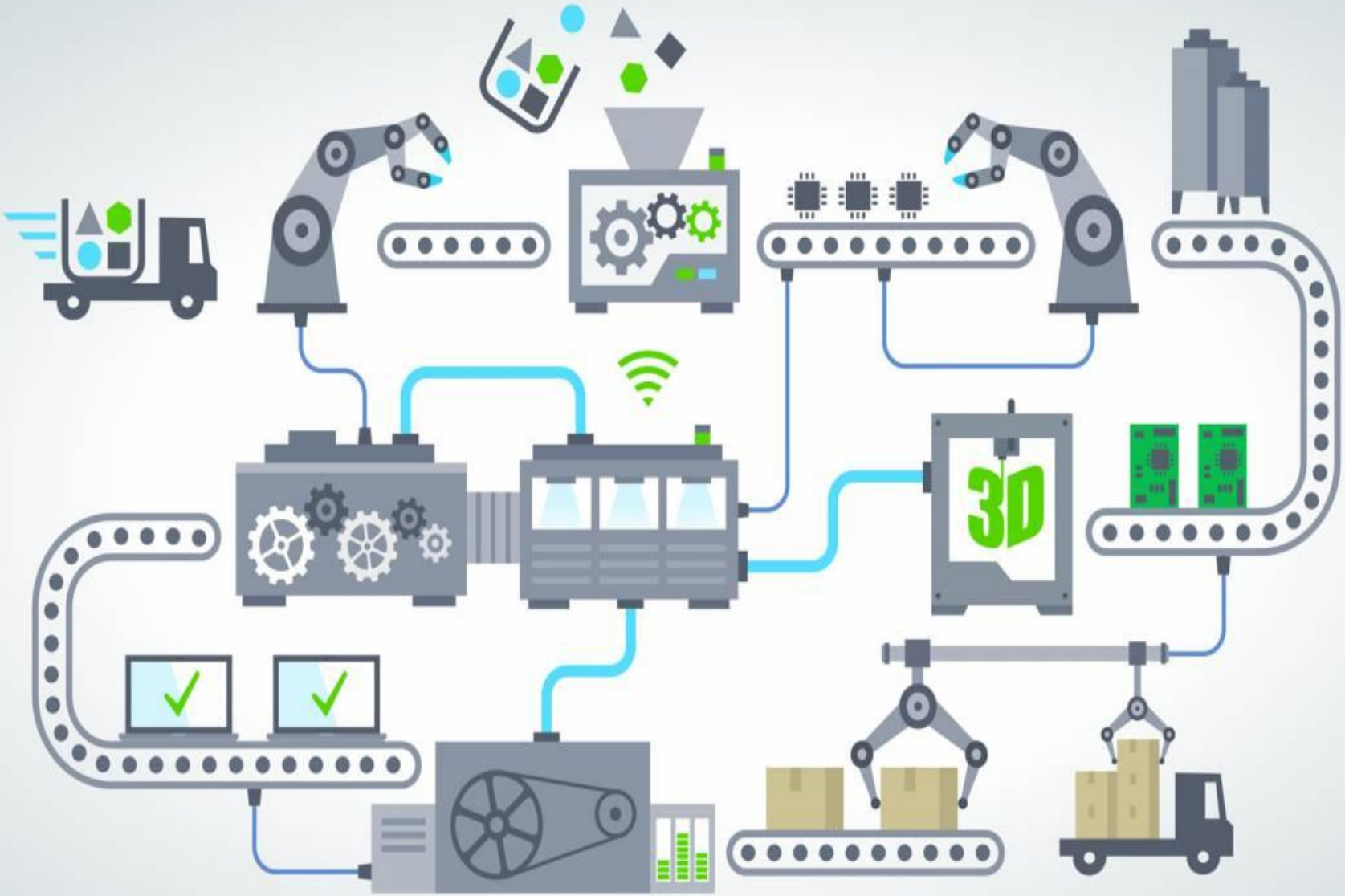
Введение в «Цифровую» экономику/ А.В. Кешелава В.Г. Буданов, В.Ю. Румянцев и др.; под общ. ред. А.В. Кешелава; гл. «цифр.» конс. И.А. Зимненко. – ВНИИГеосистем, 2017. – 28 с. (**На пороге «цифрового будущего»**. Книга первая).

Industry 4.0 | Что это?



Концепция четвертой промышленной революции («Индустрии 4.0») была сформулирована в 2011 году во время Ганноверской ярмарки группой представителей немецкой промышленности и бизнесс-сообщества в рамках инициативы по повышению конкурентоспособности Германии

Industry 4.0 | Где человек?



Industry 4.0 | Ключевые компоненты*

- Cyber-Physical Systems (CPS)
- Internet of Things (IoT)
- Internet of Services
- Smart Factory

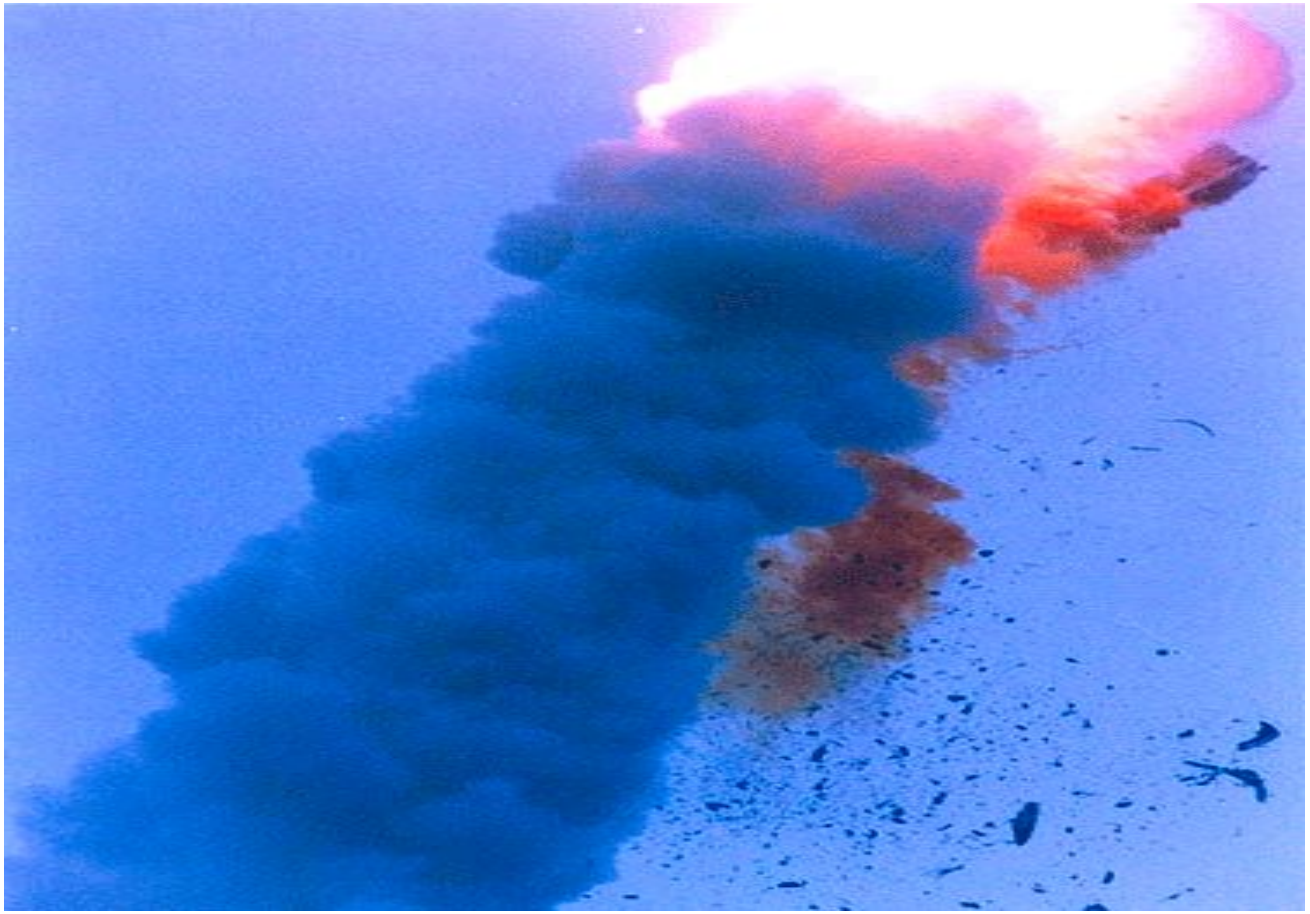
*[Hermann M., Pentek T., Otto B. Design Principles for Industrie 4.0 Scenarios: A Literature Review. Working Paper No. 01. 2015. http://www.snom.mb.tu-dortmund.de/cms/de/forschung/Arbeitsberichte/Design-Principles-for-Industrie-4_0-Scenarios.pdf]

ВОПРОС 1:

РОЛЬ ЗАДАЧ УПРАВЛЕНИЯ ФУНКЦИОНАЛЬНОЙ БЕЗОПАСНОСТЬЮ В ОБЕСПЕЧЕНИИ ЭФФЕКТИВНОГО ФУНКЦИОНИРОВАНИЯ АПК

1 Ошибка в космическом агентстве

В июне 1996 года специалисты Европейского космического агентства осуществляли запуск ракеты Ariane 5.



Ошибка в контролирующем программном обеспечении, написанном на языке программирования Ada, вызвало самоликвидацию ракеты через 37 секунд после взлета

Сбой в медицинском оборудовании

Сбои случались и в медицинском оборудовании. В 1980-годы несколько пациентов погибли после получения слишком большой дозы облучения рентгеновским аппаратом Therac-25 (лучевая терапия).


CONTROLLED
AREA
X-RAYS

Света не было не только в домах граждан, но и в больницах, школах, на вокзалах. Не работали компьютеры в полицейских участках, не функционировали радары сил противовоздушной обороны, самолёты не могли подняться со взлётных полос. Хуже того: умолкли диспетчерские вышки, и несколько страшных часов самолёты, находящиеся в воздухе, не могли сесть. Глобальный каскадный сбой, вызванный неприметной ошибкой, послужил причиной одного из самых масштабных отключений электроэнергии в истории

Авария в Мексиканском заливе: возможна ли программная ошибка?

В статье, появившейся в Houston Chronicle от 19 июля 2010 года, утверждается, что "экраны дисплеев на основной рабочей станции (ее называли A-chair), использовавшейся для управления буровой установкой на Deepwater Horizon, перед инцидентом несколько раз 'зависали'". Стефен Бертон, главный инженер компании Transocean по платформе Deepwater Horizon, заявил: "По существу, экраны могли перестать обновляться, и все данные... оказались бы заблокированы".

Вопрос 2:

Место системы информационной поддержки управления в обеспечении функционирования сложных систем

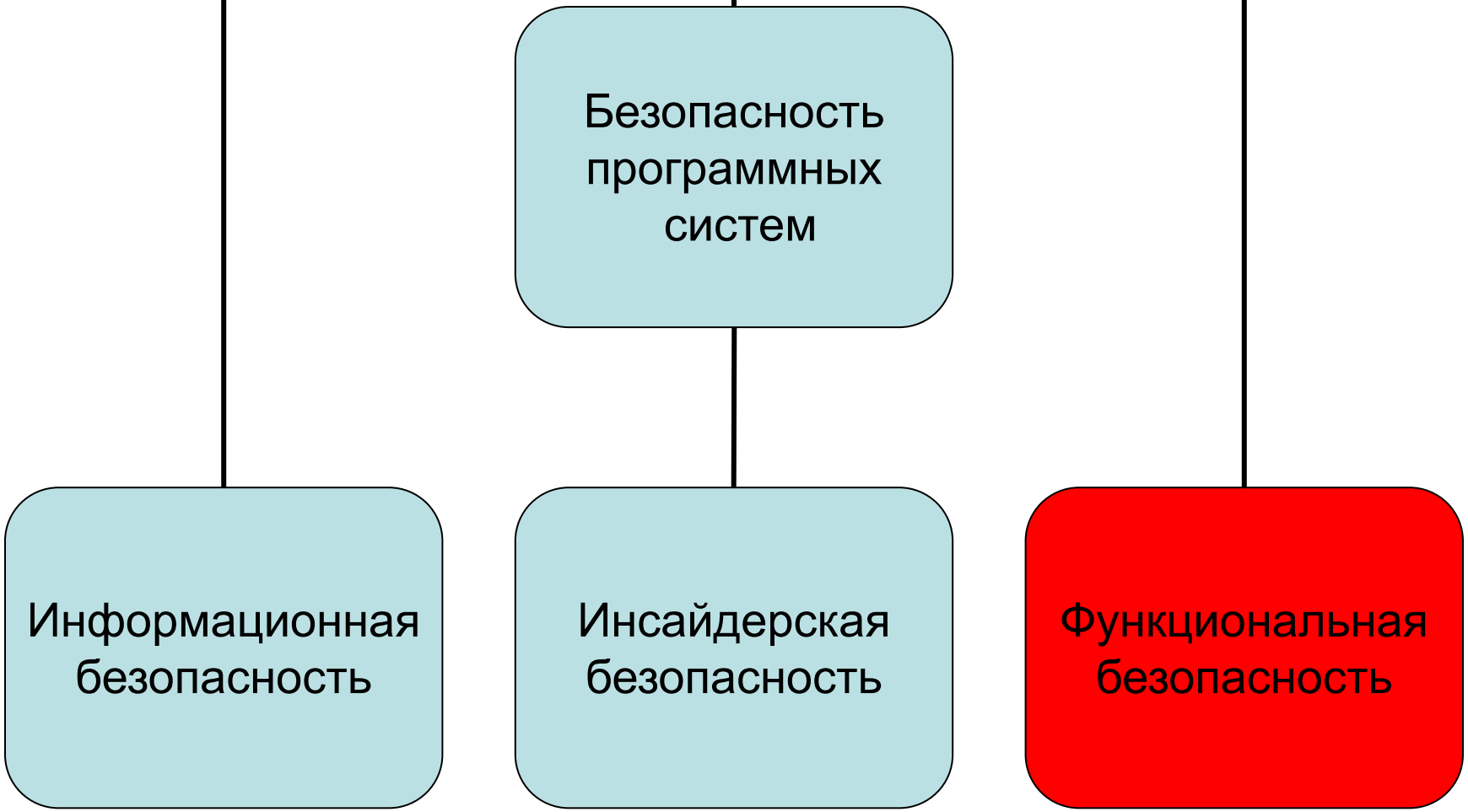
Место информационной системы в управлении бизнес-процессами

Вывод:

Необходимо рассматривать как единое целое Hard+soft+brain

Место программной компоненты в управлении сложными техническими системами

Основные подходы к обеспечению безопасности программных систем



Вывод:

Необходимо совершенствование технологий разработки, позволяющих обеспечивать защиту изделий как от злонамеренных действий, так и от непреднамеренных ошибок, допускаемых разработчиками на разных стадиях жизненного цикла программных систем.

Вопрос 3:

В чем причина кризиса в IT?

Статистические данные о текущей эффективности реализации программных проектов (по данным, относящимся к США)

Источники данных

The Standish Group Report CHAOS. Project Smart-2014, 16p.

CHAOS MANIFESTO The Laws of CHAOS and the

CHAOS 100 Best PM Practices – 2011, 51p.

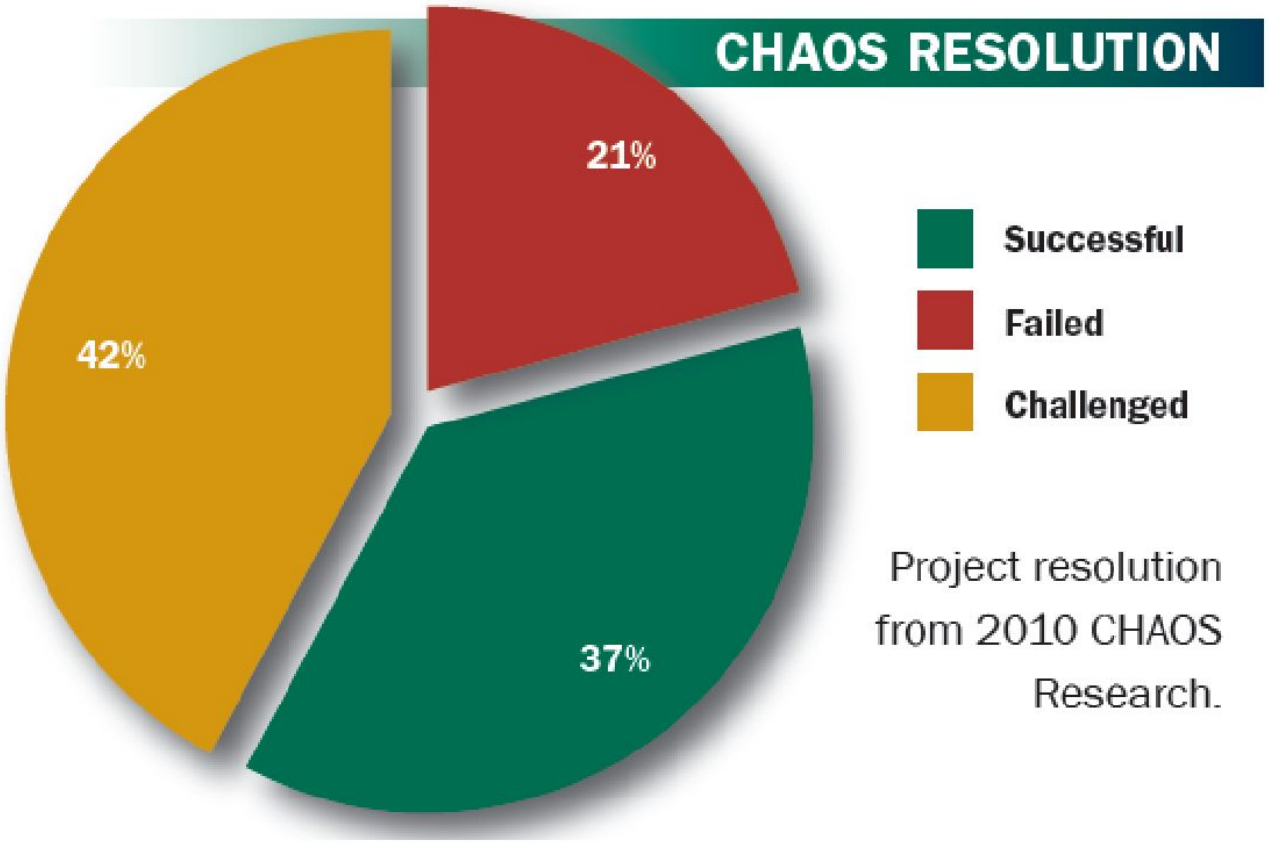

Dawn of a
New PM

About The Standish Group

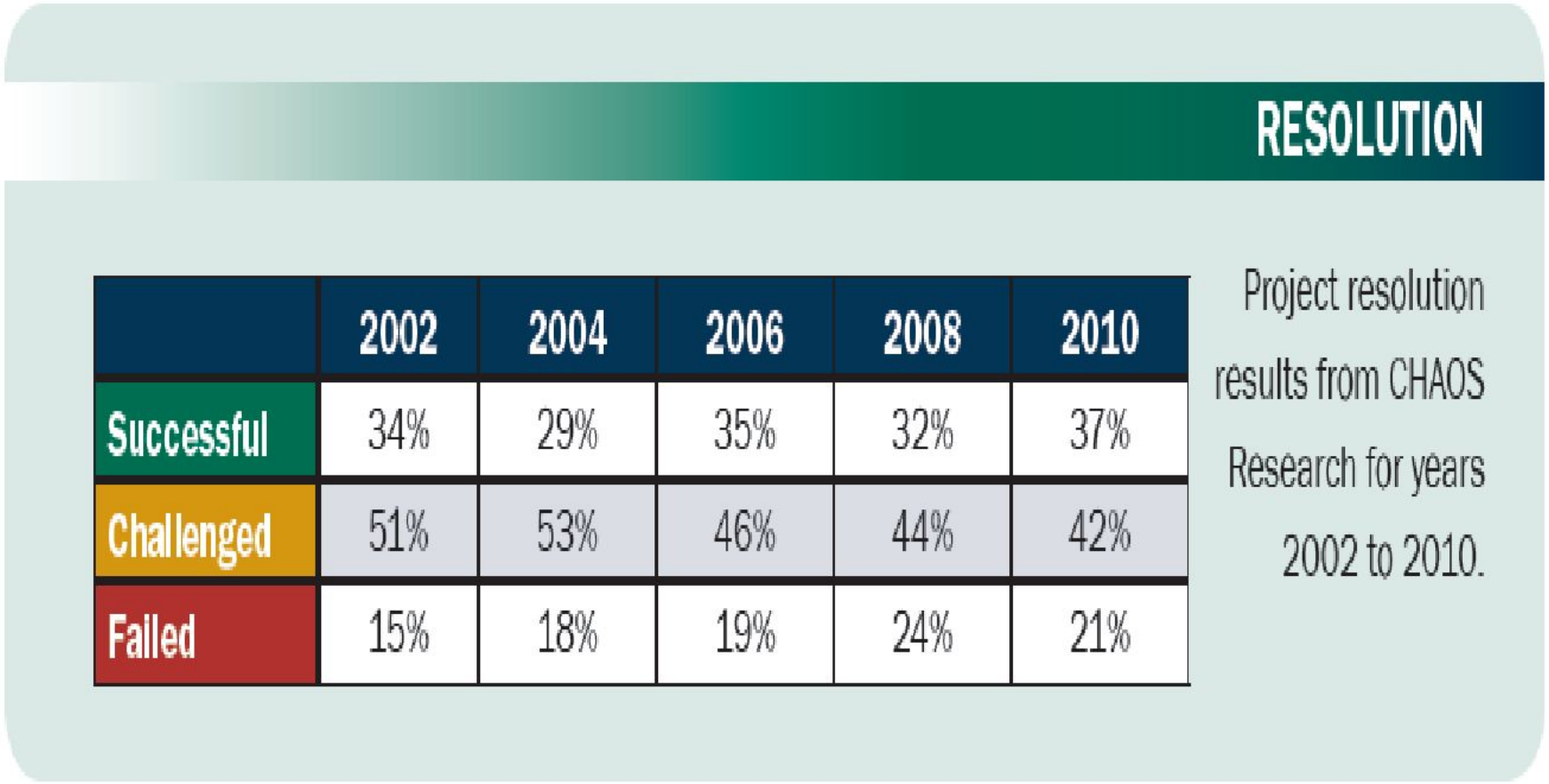
The Standish Group is a primary research advisory organization that focuses on software project performance. Using our extensive primary research you can improve your investments in software projects. We are a group of highly dedicated professionals with years of practical experience helping organizations improve.

The Standish Group was formed in 1985 with a vision of innovating group refection using case-based reasoning techniques. We do this in order to profile your projects and environments against thousands of cases to deliver more precise advice based on collective wisdom. For over 30 years The Standish Group has been researching and providing advice on how to increase the value of software investments

Эффективность реализации программных проектов по данным 2010 г.



Динамика эффективности реализации программных проектов



RESOLUTION

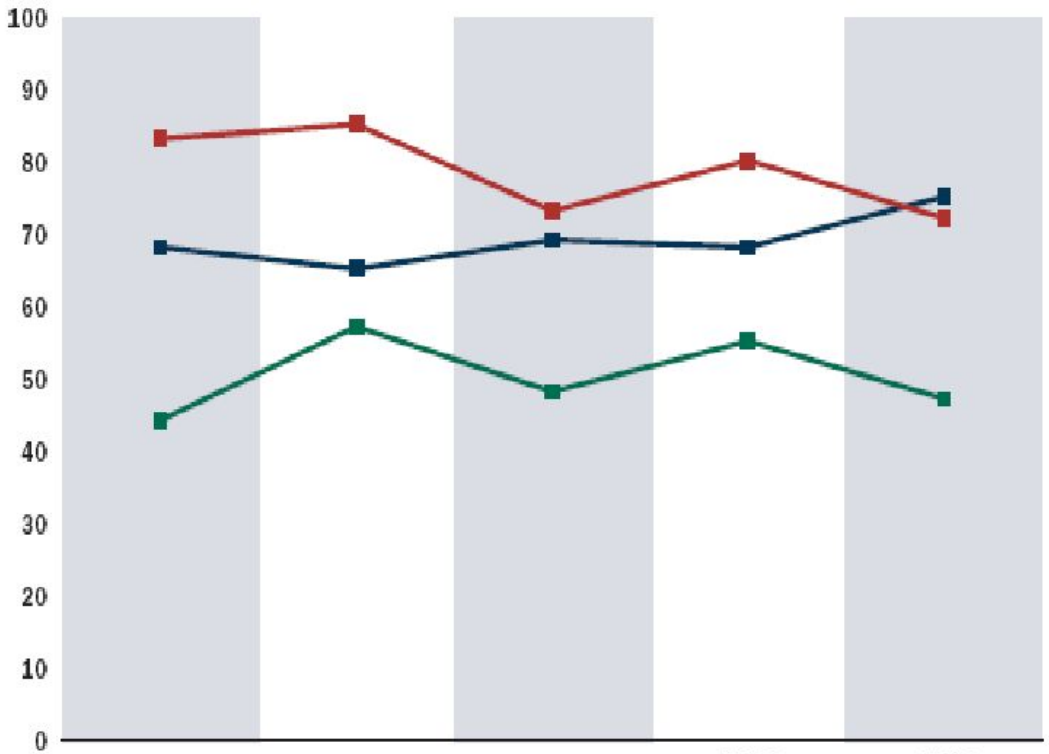
	2002	2004	2006	2008	2010
Successful	34%	29%	35%	32%	37%
Challenged	51%	53%	46%	44%	42%
Failed	15%	18%	19%	24%	21%

Project resolution results from CHAOS Research for years 2002 to 2010.

Последствия недостаточного качества реализации программных проектов

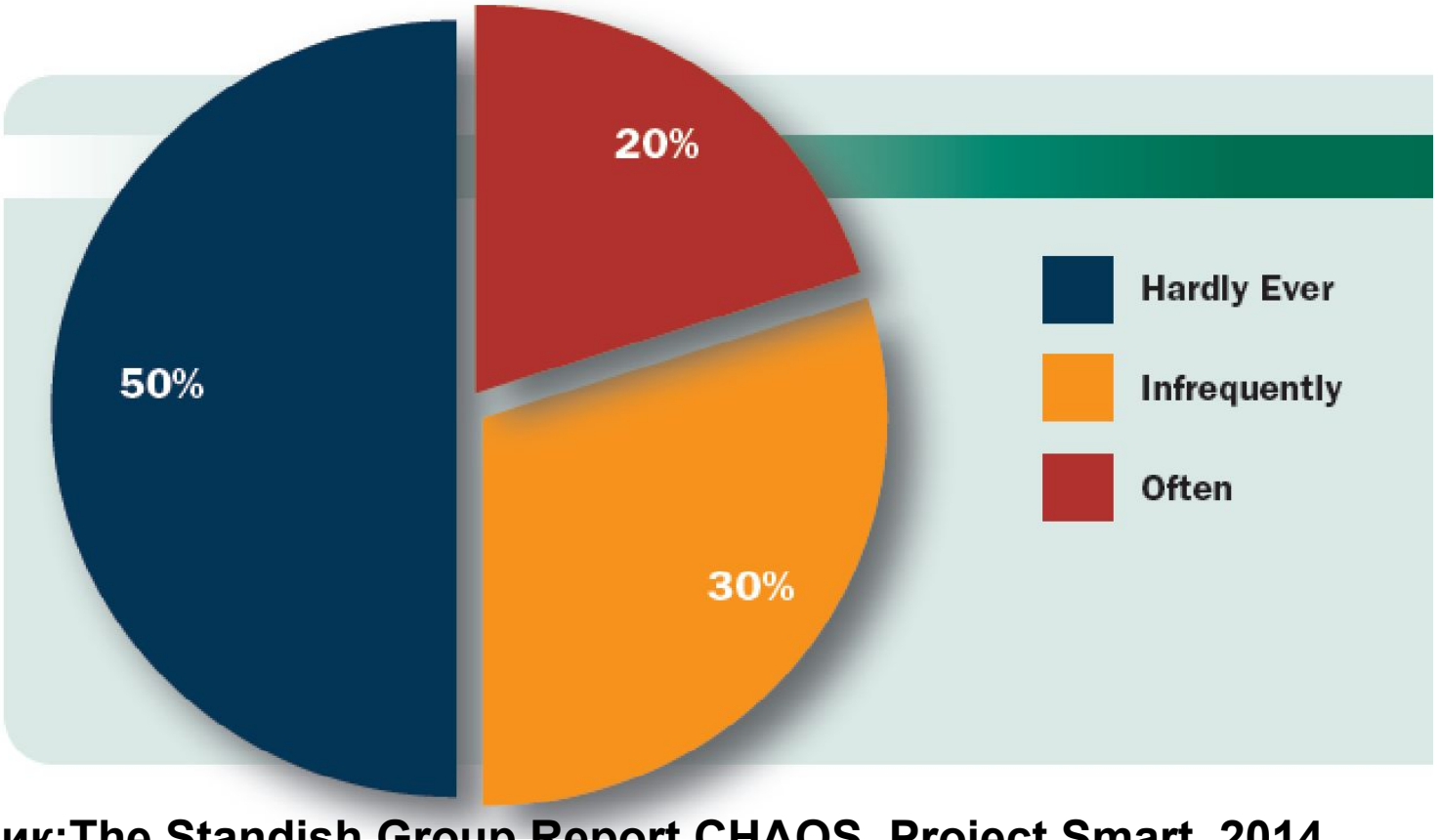
OVERRUNS AND FEATURES

Time and cost overruns, plus percentage of features delivered from CHAOS Research for the years 2002 to 2010.



	2002	2004	2006	2008	2010
Time	82%	84%	72%	79%	71%
Cost	43%	56%	47%	54%	46%
Features	67%	64%	68%	67%	74%

Реальная востребованность возможностей программного продукта



Источник:The Standish Group Report CHAOS. Project Smart, 2014

Статистические данные о эффективности реализации программных проектов (по данным, относящимся к США)

1. Ежегодные затраты на реализацию IT-приложений: 250 млрд. $
2. Количество реализуемых проектов: 175 000
3. Диапазон стоимости проектов: 34000$-2 322 000$

Статистические данные о эффективности реализации программных проектов (по данным, относящимся к США, продолжение)

4. Среднее превышение плановой стоимости -189%
5. Среднее превышение плановых сроков реализации – 222%
6. Реализуется лишь 61% функциональных и нефункциональных требований, заявленных в техническом задании
7. Общие потери, учитывающие упущенные возможности – триллион долларов

Основной вывод отчета**The Standish Group Report CHAOS. Project Smart, 2014**

В настоящее время недочетов в программных проектах больше, чем было пять-десять лет назад, несмотря на радикальное повышение зрелости инструментальных средств и технологий реализации программных продуктов

Факторы, приводящие к провалу проекта

N п/п	Факторы, приводящие к провалу проекта	Оценки респондентов
1.	**Недостаточная вовлеченность пользователей**	**12.8%**
2.	**Неполные требования и спецификации**	**12.3%**
3.	**Изменения в требованиях и спецификациях**	**11.8%**
4.	Недостаточная поддержка со стороны руководства	7.5%
5.	Низкая квалификация сотрудников	7.0%
6.	Недостаток ресурсов	6.4%
7.	Нереалистичные ожидания	5.9%
8.	Нечеткие цели	5.3%
9.	Нереалистичные временные границы проекта	4.3%
10.	Новые технологии	3.7%
11.	Иные	23%

Факторы успеха проекта и их значимость

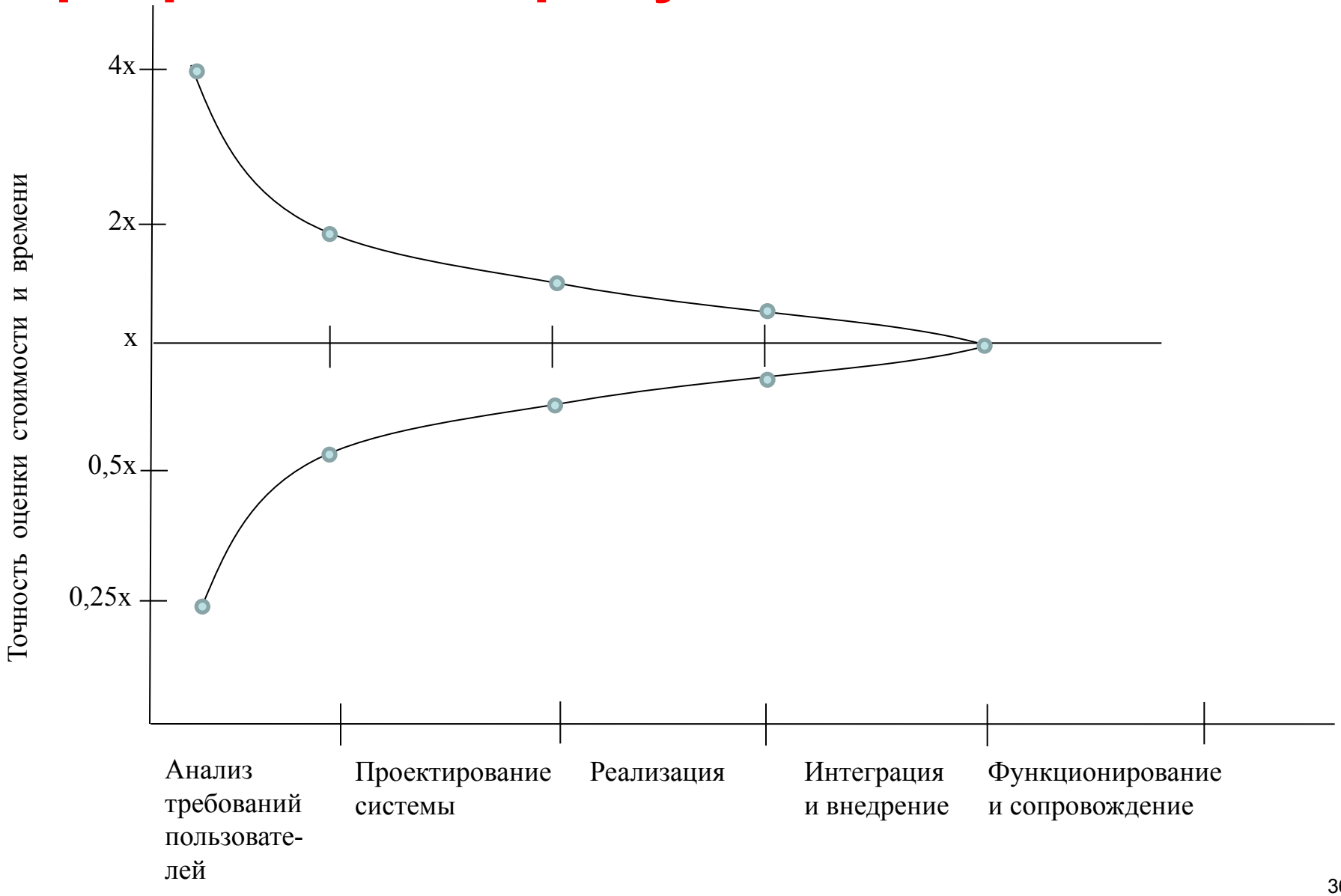
N п/п	Наименование фактора	Вес фактора
1.	Вовлеченность пользователей	0.2
2.	Участие кураторов	0.15
3.	Ясные бизнес-цели	0.15
4.	Эмоциональная зрелость	0.12
5.	Организация проекта	0.11
6.	Скорость реализации процессов	0.11
7.	Управление проектом	0.06
8.	Квалификация персонала	0.05
9.	Контроль управления	0.03
10.	Инструменты и инфраструктура	0.02

Вывод:

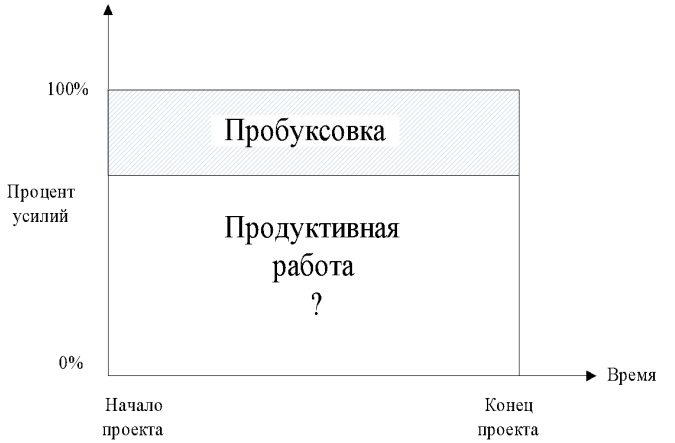
...В следующий раз, услышав о провале проекта по внедрению корпоративной информационной системы, подумайте прежде всего о плохой работе менеджмента, а потом уже об отказе программного обеспечения. Плохое программное обеспечение не убивает компании, это делает плохой менеджмент...

Источник:www.rbcgrp.com/erp-bi.html

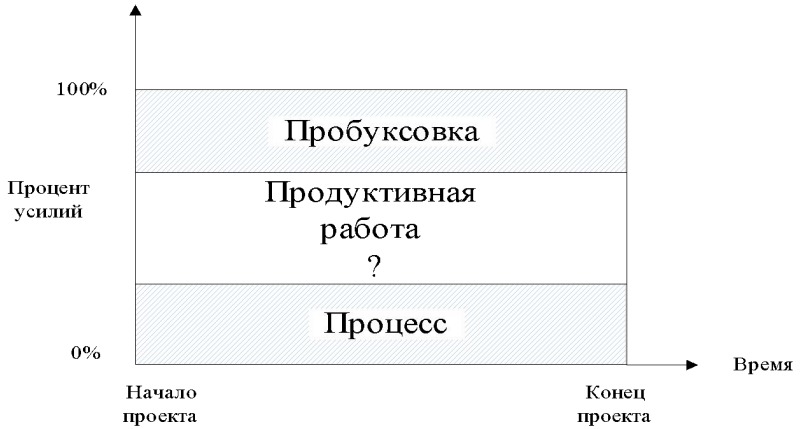
Конус неопределенности программного продукта



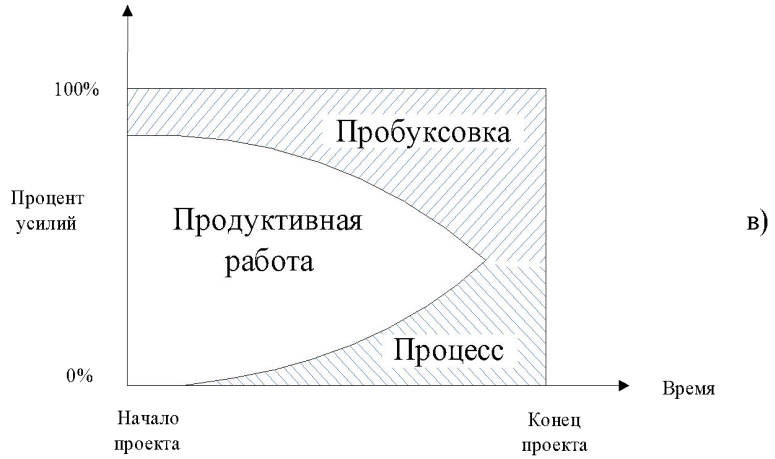
Роль дисциплины при проектировании сложных программных систем



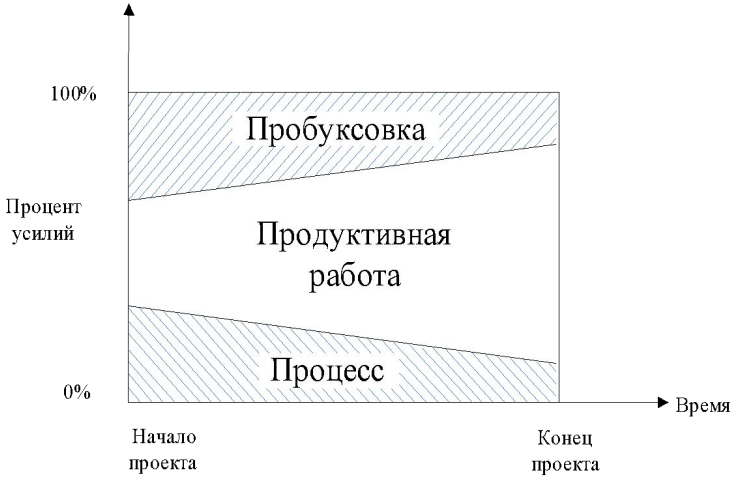
а)



б)



в)



г)

Роль моделирования при реализации программных проектов

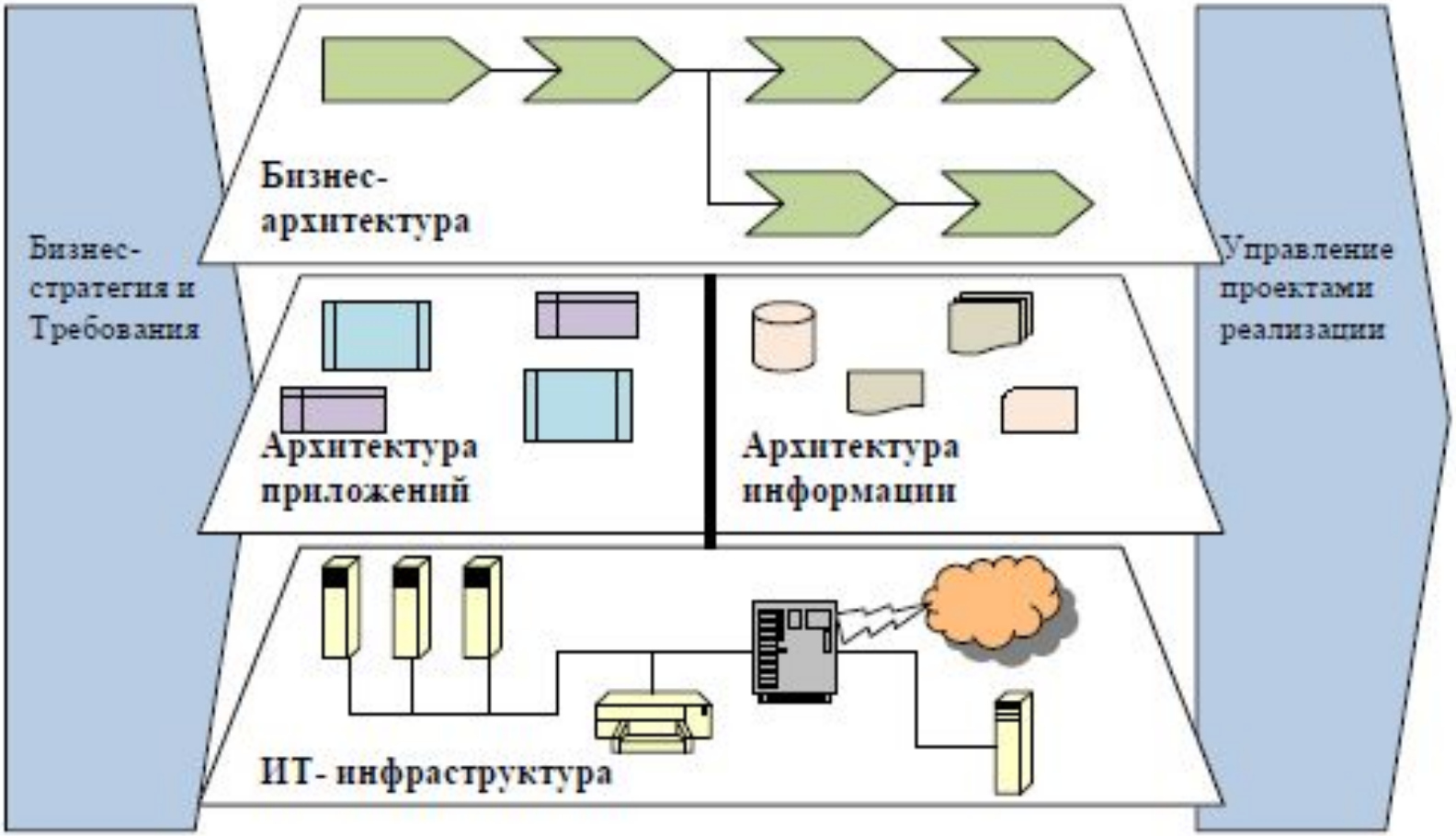
Возможности модели жизненного цикла программной системы (потенциальность модели) должна соответствовать сложности реализации программного продукта.

Сложность реализации программного продукта определяется уровнем неопределенности требований к потребительским свойствам конечного продукта

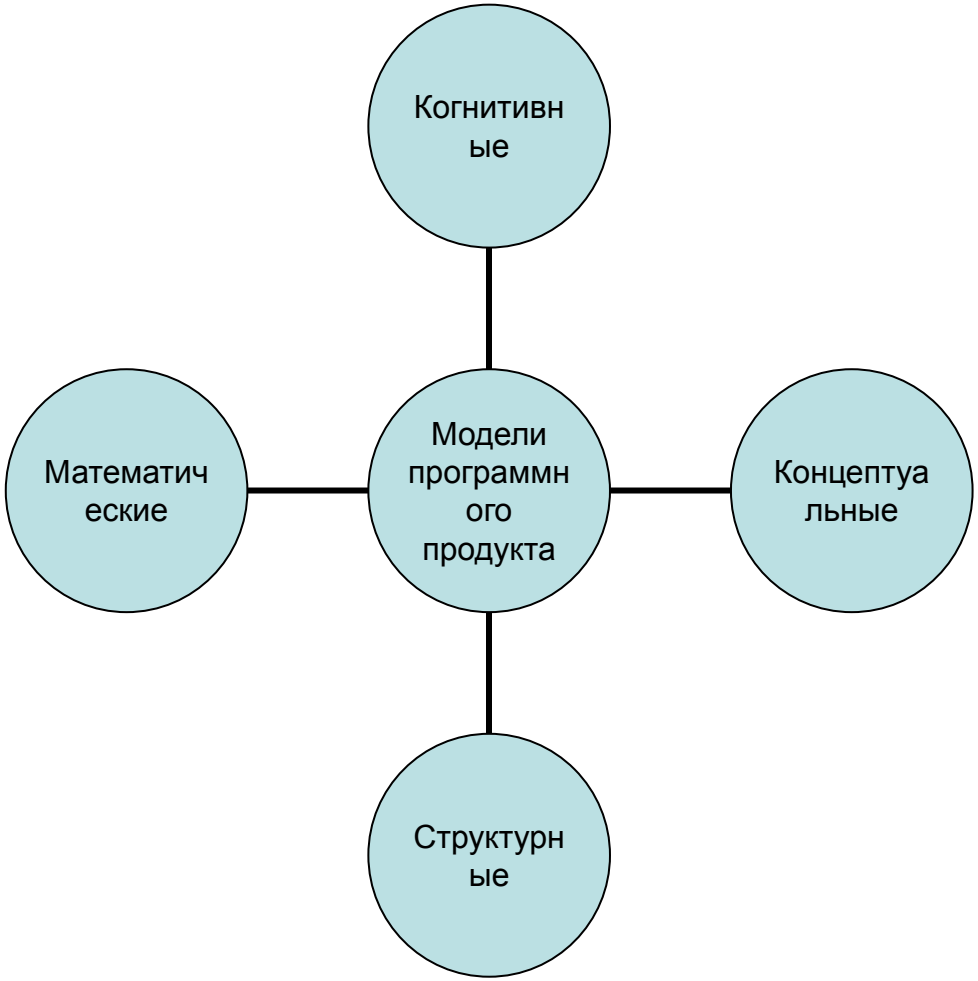
Некоторые модели жизненного цикла

Основные компоненты информационной системы

Содержание MDA



Примеры классов (фреймов) моделей программных систем



Code-and-fix model

Реализация программного продукта сводится к непосредственному кодированию задачи в том виде, как она понимается.

Особенностями этой модели являются:

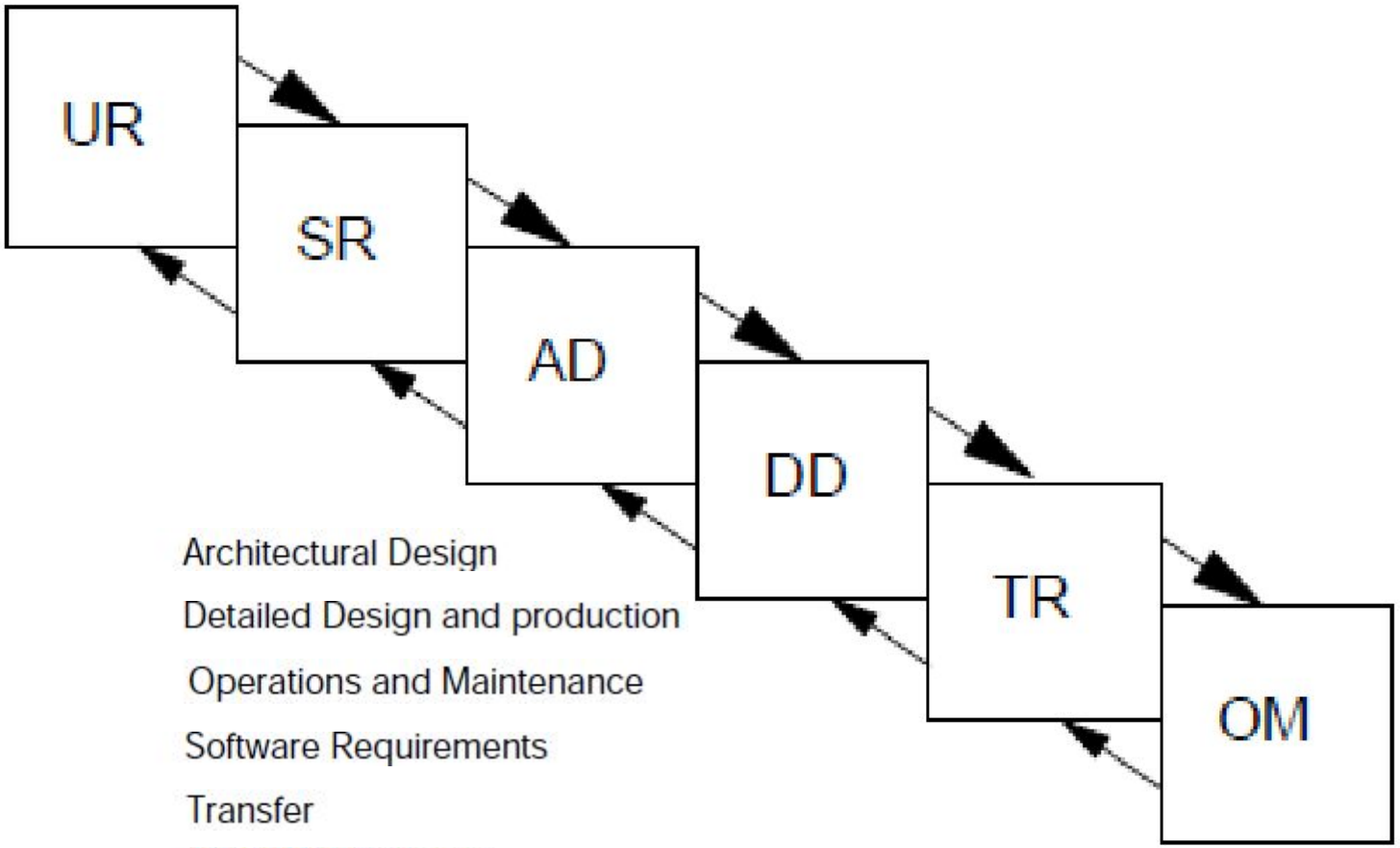
- Трудность модификации и развития ПП из-за недостаточно проработанной проектной стадии.
- Вследствие того, что задача кодировалась как понималась, т.е. стадия изучения и согласования пользовательских требований реализовывалась посредством экспериментирования с уже готовой программой, функциональные возможности программного продукта редко полностью согласуются с потребностями пользователей
- Сложность тестирования программного продукта.

Stagewise model

Разработка программного продукта сводится к следующей последовательности действий:

- Планирование разработки.
- Разработка операционной спецификации.
- Кодирование.
- Параметрическое тестирование модулей.
- Тестирование сборки.
- Опытную эксплуатацию.
- Оценку системы пользователем.

The waterfall approach



AD	Architectural Design
DD	Detailed Design and production
OM	Operations and Maintenance
SR	Software Requirements
TR	Transfer
UR	User Requirements

Figure 1.3.1 *The waterfall approach*

The evolutionary development approach

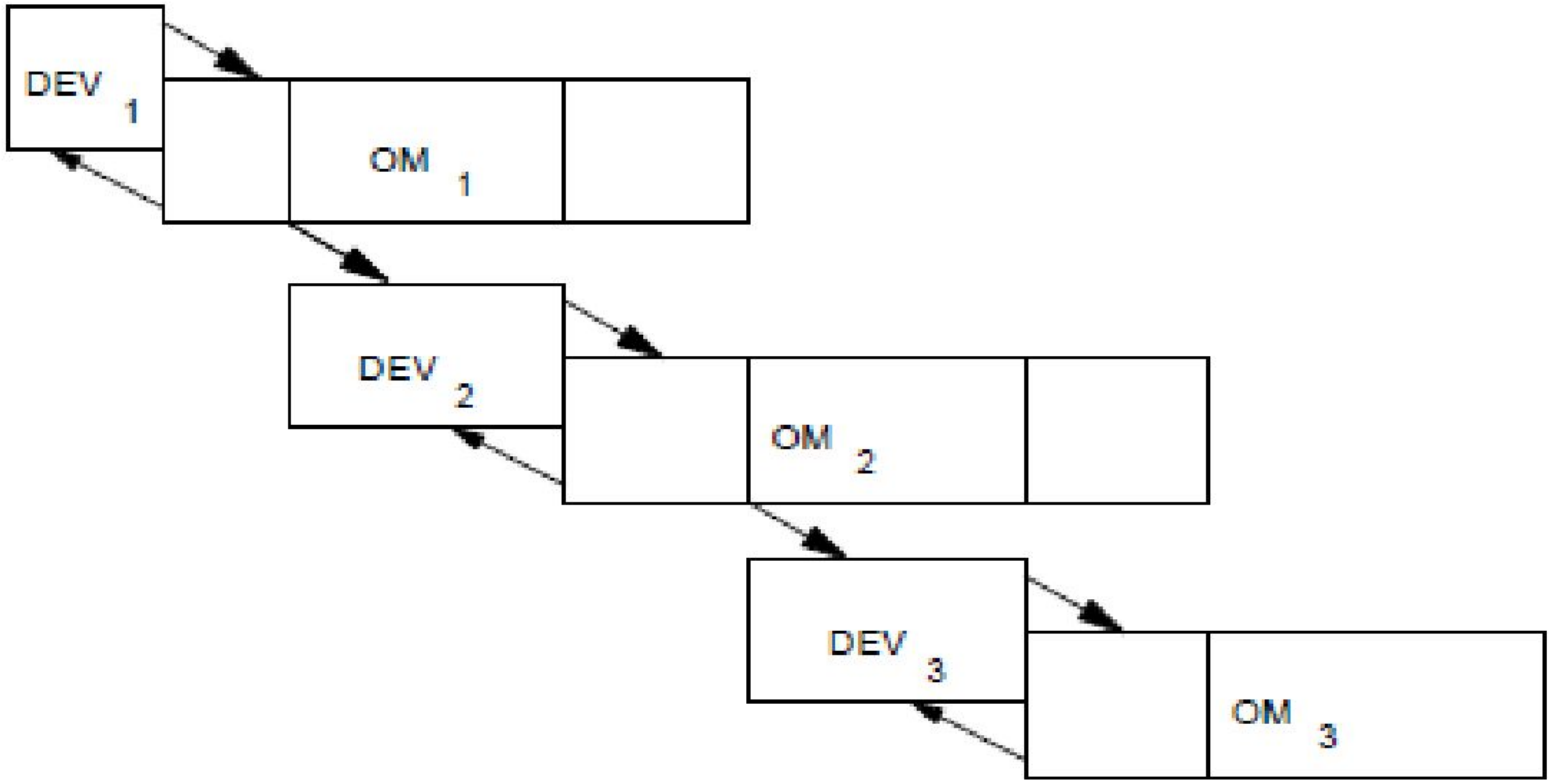
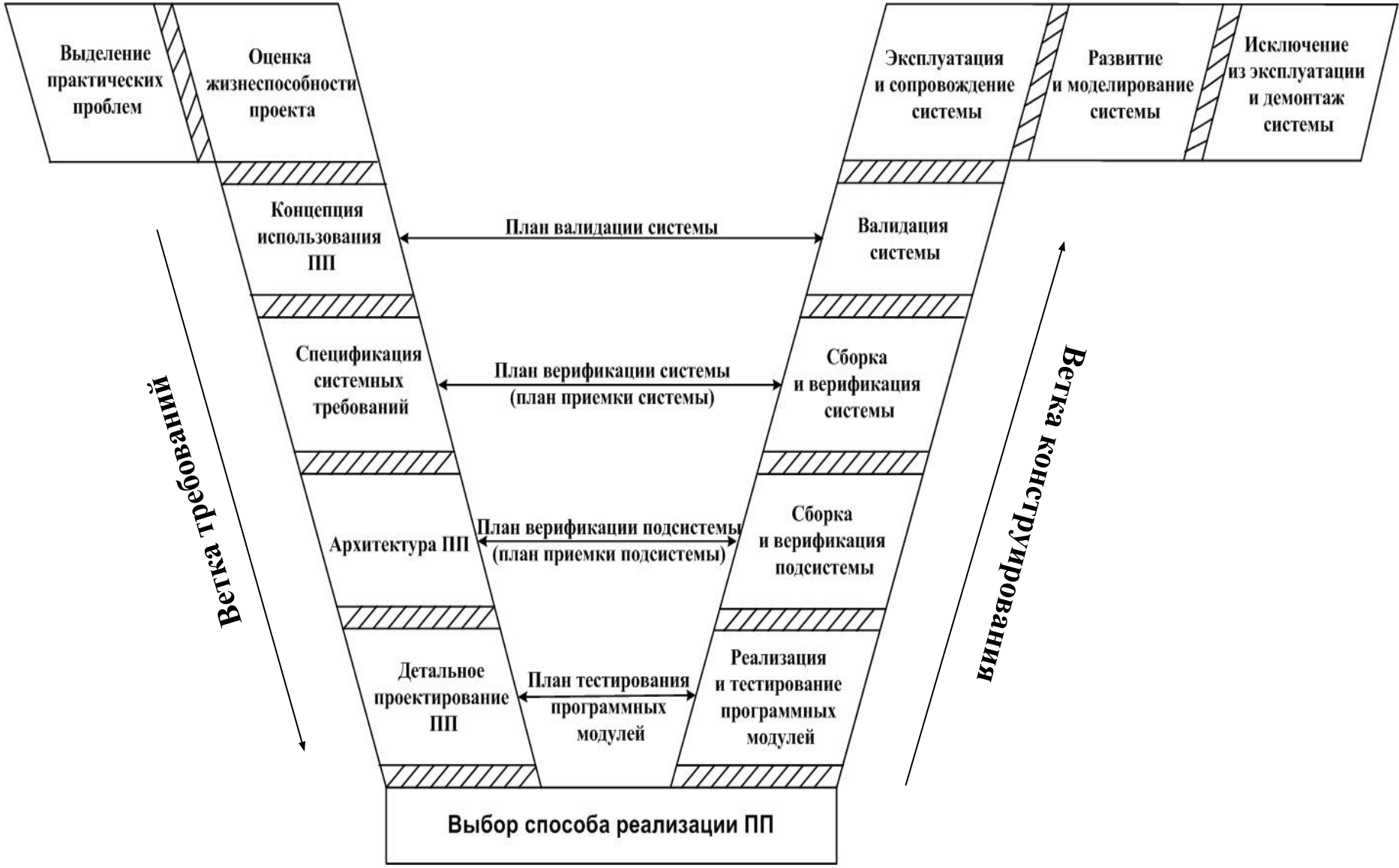


Figure 1.3.3 *The evolutionary development approach*

Общий вид V-модели жизненного цикла



Взаимосвязь разработки и испытаний

1. Это две стороны одной медали: разработки и тестирование неделимое целое
2. Создание процесс созидательный, испытания – разрушительный.
3. Испытания – процесс затратный не повышающий качества продукта
4. Модель жизненного цикла создает методическую основу выбора стратегии испытаний

Куликов С.С. Тестирование программного обеспечения. Базовый курс.- Минск, Четыре четверти, 2017.-312 с.

Краткий очерк истории тестирования

50-60 годы

1. Концепция «исчерпывающего тестирования» (exhausting testing): проверка всех возможных путей выполнения со всеми возможными исходными данными.
2. Процесс тестирования предельно формализован, отделен от процесса разработки ПО и «математизирован»

Ограничения:

1. Невозможно найти ошибки в документации
2. Исчерпывающее тестирование практически невозможно (слишком большое число возможных путей)

Краткий очерк истории тестирования (продолжение)

70-е годы

1. Возникли две фундаментальные идеи тестирования:

   (а). Доказательство работоспособности программ в некоторых заданных условиях (positive testing)

   (б). Доказательство неработоспособности программ в некоторых заданных условиях (negative testing)

Ограничения:

Необходимо заранее четко определить условия использования

Философия «белого» и «черного» ящиков

1. «Белый» ящик. Цель – проверить каждый путь алгоритма. При этом спецификация не представляет интереса
2. «Черный» ящик. Цель: проверить поведение программы при всех возможных сочетаниях входных данных.«…Меня не интересует, как выглядит эта программа и выполнил ли я все команды или все пути. В буду удовлетверен, если программа будет вести себя так, как указано в спецификациях…»

Источник: Г.Майерс Надежность программного обеспечения М.: Мир, 1980

Стратегии тестирования интеграции

1. Восходящее тестирование
2. Нисходящее тестирование
3. Модифицированный нисходящий метод
4. Метод сэндвича
5. Метод «большого скачка»

Источник: Г.Майерс Надежность программного обеспечения М.: Мир, 1980

Краткий очерк истории тестирования (продолжение)

80-е годы

1. Ключевое изменение места тестирования в разработке ПО: вместо финальной стадии реализации проекта тестирование стало применяться в течение всего жизненного цикла разработки, что позволило не только сократить «латентный период» ошибки, но и в известной степени предотвратить их появление

2. Бурное развитие и формализация методов тестирования

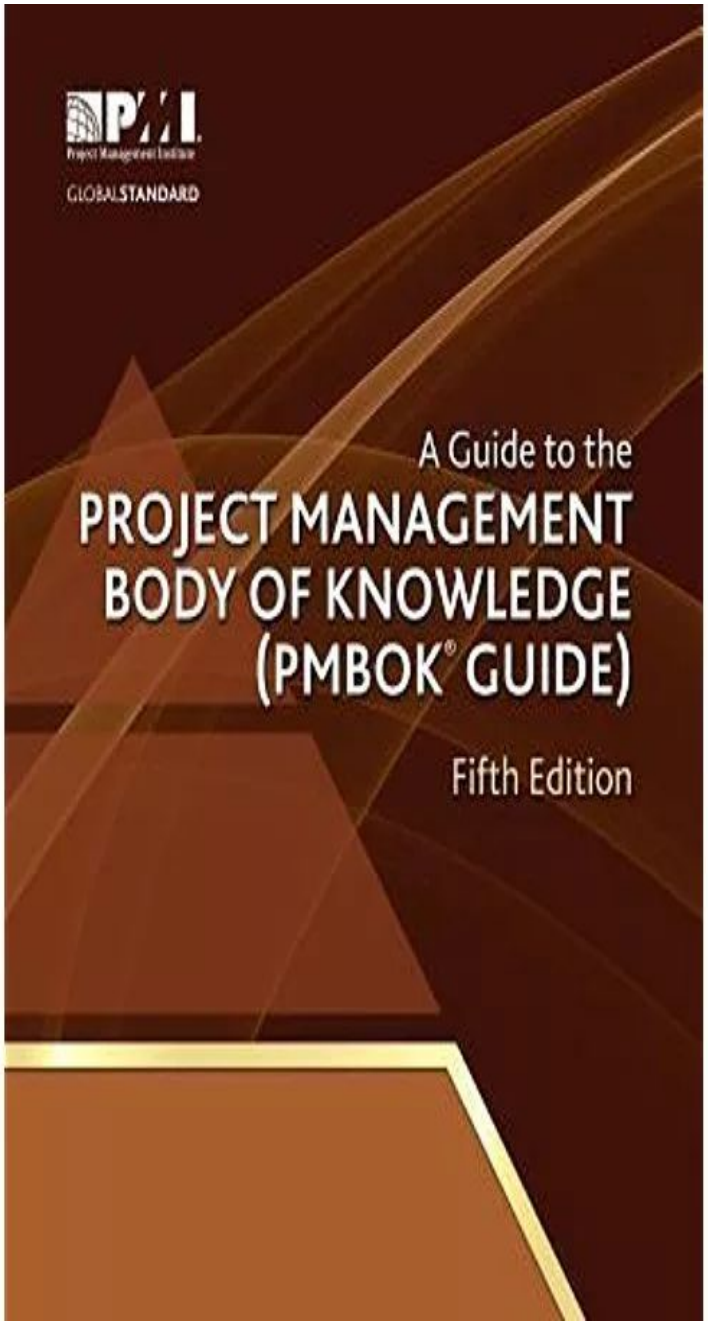
3. Первые попытки автоматизации тестирования

Краткий очерк истории тестирования (продолжение)

90-е годы

Переход от тестирования к более всеобъемлющему процессу, именуемому «Управление качеством» («quality assurance»), который охватывает весь цикл разработки ПО и захватывает процесс планирования, проектирования, реализации ПО.

(фактически признание неразделимого единства основного, вспомогательного и обеспечивающего процессов в рамках программного проекта-PMBOK)

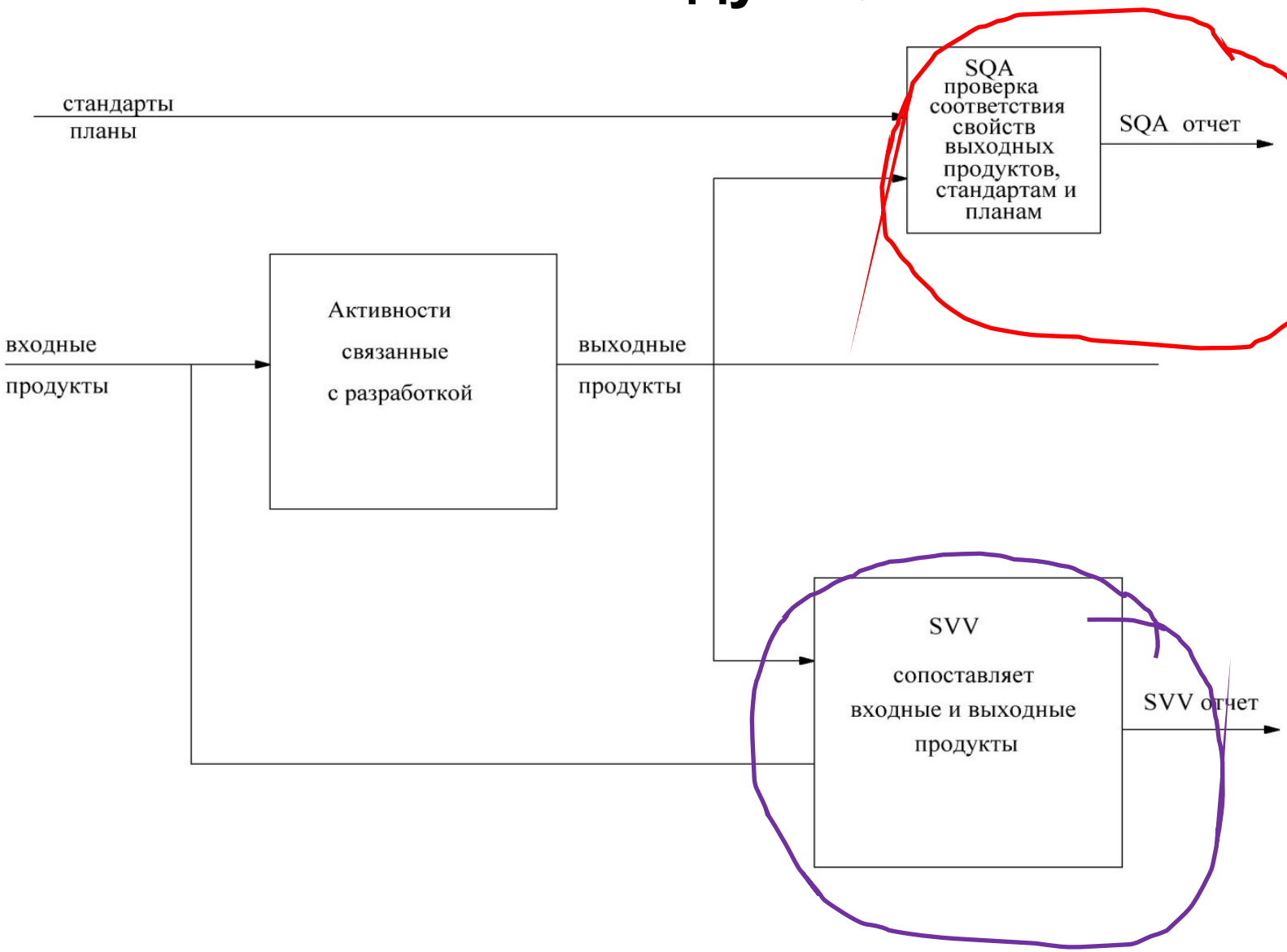

PMI
Project Management Institute
GLOBAL STANDARD
A Guide to the
PROJECT MANAGEMENT
BODY OF KNOWLEDGE
(PMBOK® GUIDE)
Fifth Edition

ESA PSS-05-11 Issue 1 Revision 1
March 1995

Guide to software quality assurance

Prepared by:
ESA Board for Software Standardisation and Control (BSSC)

Различие между SQA и SVV

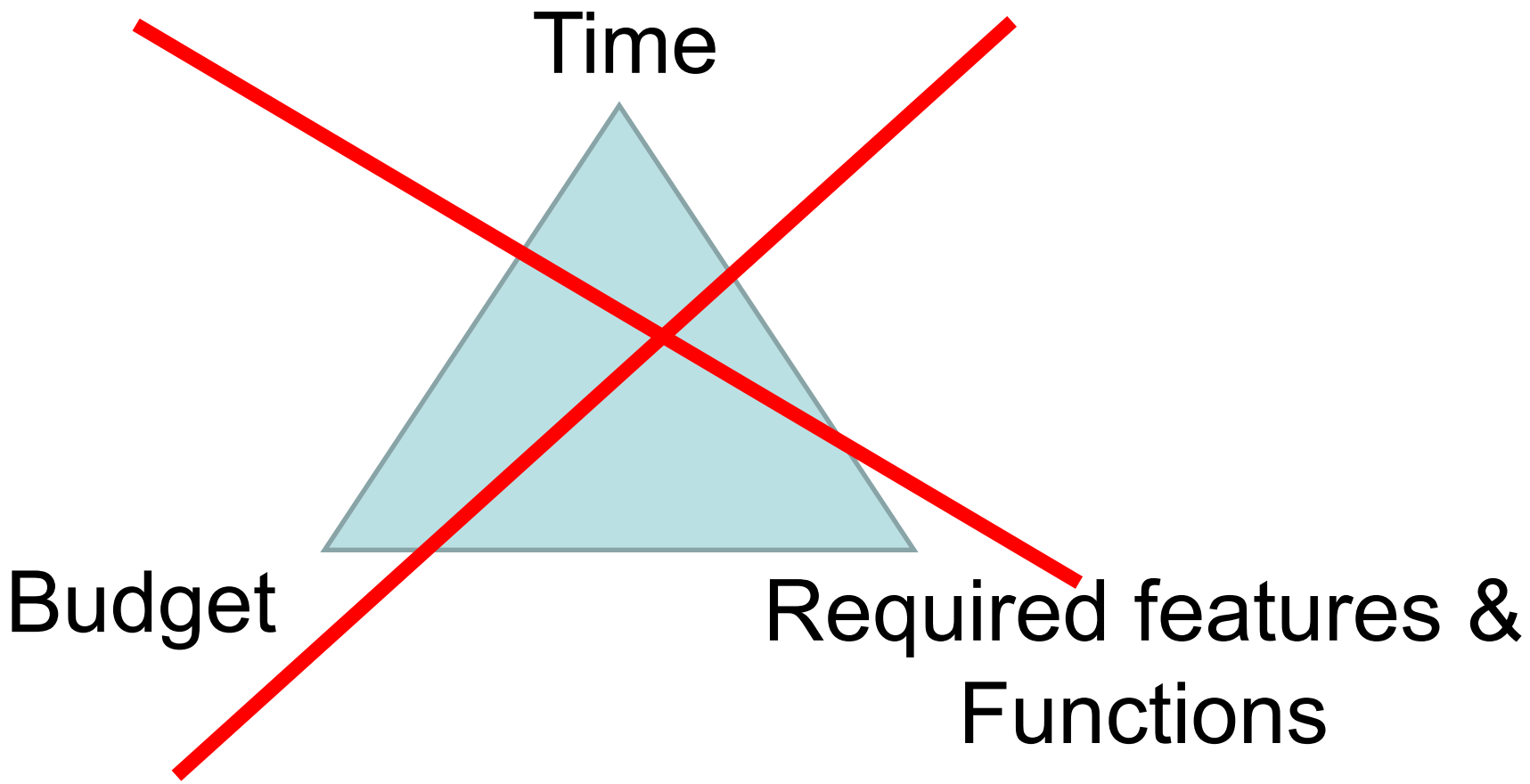


Краткий очерк истории тестирования (продолжение)

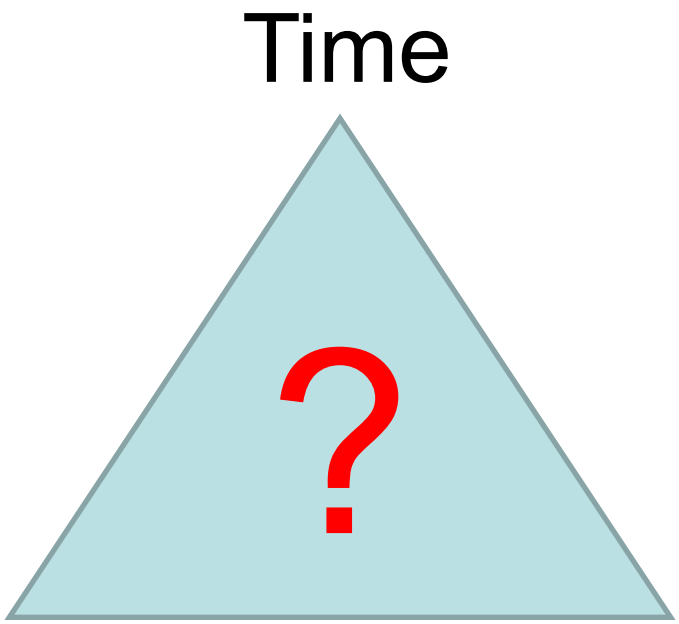
2000-е годы

1. Поиск новых подходов к обеспечению качества
2. Автоматизация тестирования
3. Во главу процесса тестирования ставятся не соответствие программы требованиям, а её способность предоставить конечному пользователю возможность эффективно решать свои задачи

Project Triangle (PMI-1994)



Project Triangle
(Standish Group-2015)



Краткий очерк истории тестирования (продолжение)

Текущее состояние

1. Гибкие методологии и гибкое тестирование
2. Глубокая интеграция процессов испытаний с процессами разработки и автоматизации
3. Огромный набор методологий и инструментальных средств
4. Кросс функциональные команды (программист и тестировщик могут выполнять работу друг друга)

Основные выводы

1. Границы, критерии качества систем, требования к потребительским свойствам становятся все более размытыми.

2. Рост масштабов и сложности систем (поведения, устройства, проектирования, внедрения, сопровождения,…)

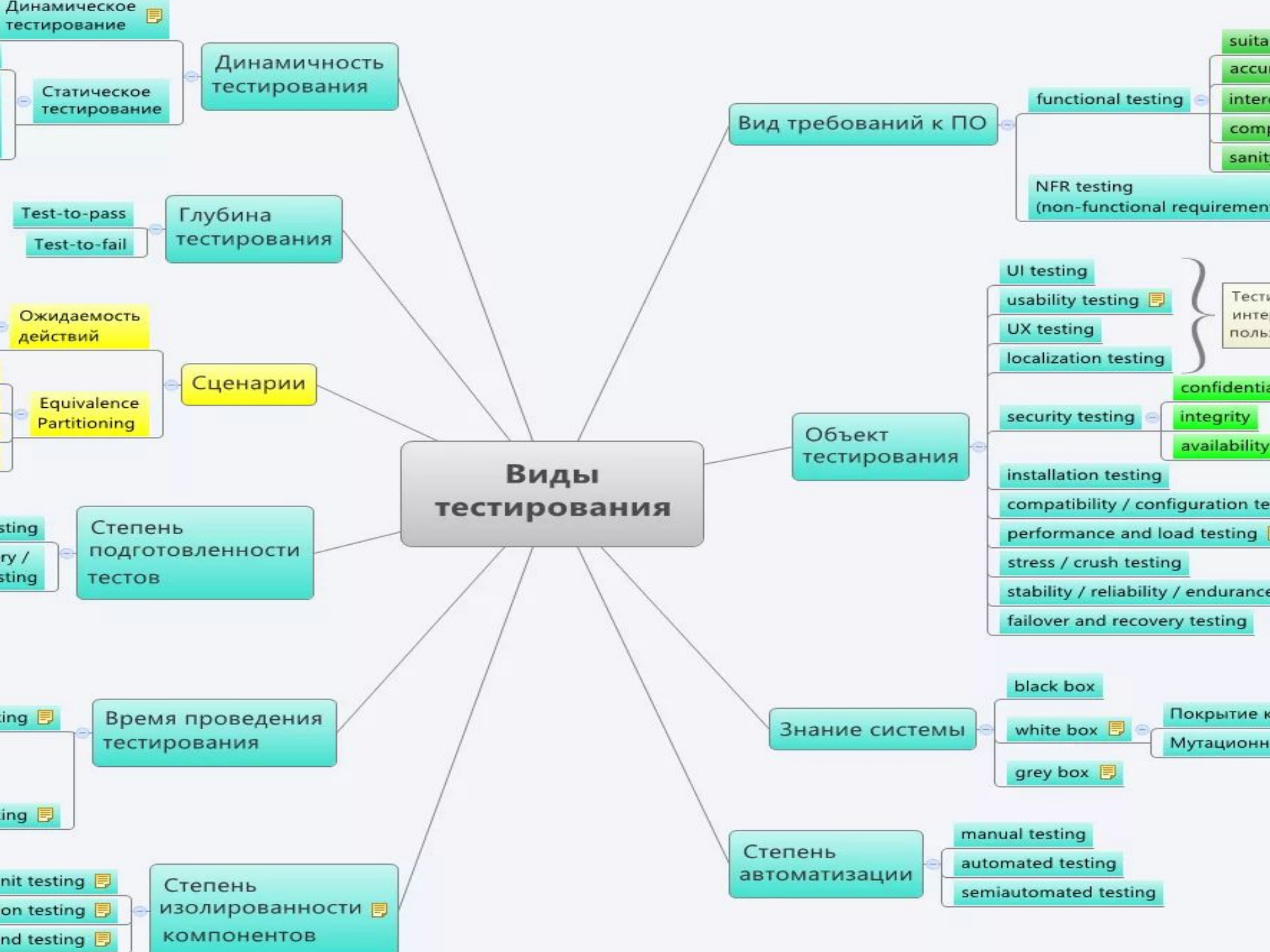
3. Подходы к разработке и испытаниям, хорошо зарекомендовавшие себя при реализации локальных программных продуктов, локальных и корпоративных информационных систем, в новых условиях имеют ограниченное применение.


test
include
tools
methods
time
expected
also
example
may
black
code
system
process
development
regression
defects
used
often
requirements
tester
box
quality
product
developers
non-functional
tested
integrated
offered
needed
Certification
tests
case
program
cases
result
data
common
results
coverage
performance
level
found

Разноаспектные подходы к тестированию программных средств (классические подходы)

ВИДЫ тестирования

QATestLab
quality is a rule


Виды тестирования
Динамичность тестирования
Динамическое тестирование
Статическое тестирование
Глубина тестирования
Test-to-pass
Test-to-fail
Сценарии
Ожидаемость действий
Equivalence Partitioning
Степень подготовленности тестов
Время проведения тестирования
Степень изолированности компонентов
Вид требований к ПО
functional testing
NFR testing
Объект тестирования
UI testing
usability testing
UX testing
localization testing
security testing
confidentia
integrity
availability
installation testing
compatibility / configuration te
performance and load testing
stress / crush testing
stability / reliability / endurance
failover and recovery testing
Знание системы
black box
white box
grey box
Покрытие к
Мутационн
Степень автоматизации
manual testing
automated testing
semiautomated testing

Классификация видов тестирования:

- По знанию внутренностей системы
- По объекту тестирования
- По субъекту тестирования
- По времени проведения тестирования
- По критерию “позитивности” сценариев
- По степени изолированности тестируемых компонент
- По степени автоматизированности тестирования
- По степени подготовленности к тестированию

Виды тестирования

Способы тестирования

- Тестирование проводится не только на той стадии разработки программ, которая специально для этого предназначена, но и на предшествующих стадиях – при автономной отладке программ, еще до объединения их в единый программный комплекс. Такое тестирование называется ***автономным***. Его обычно проводят сами разработчики, которые проверяют точное соответствие программы выданной им спецификации.

- ***Комплексное тестирование*** призвано проверить все аспекты работы программы от правильности взаимодействия внутренних программных компонентов до правильности взаимодействия программного комплекса с его пользователями.

- Во время ***пользовательского тестирования*** результаты работы программы проверяются с прикладной точки зрения.

- ***Техническое тестирование*** дает возможность проверить безопасную и эффективную работу созданной программы в нормальном и пиковом режимах ее использования. Функциональность на этом этапе проверяется только в смысле ее влияния на важнейшие технические параметры программы, например, на время реакции системы на запрос пользователя.

- Важной в тестировании является возможность проведения ***регрессивного тестирования***. Регрессивные тесты, повторяемые после каждого исправления программы, позволяют убедиться, что функциональность программы, не связанная с внесенным исправлением, не затронута этим исправлением и не утрачена из-за него.

Понятия альфа- тестирования

…После того, как отдельные программные модули готовы, они объединяются в некое единое целое. Это еще не полнофункциональная программа, но она уже способна работать и выполнять, хотя бы частично, свои главные задачи. Такой вариант программы и называют ***альфа-версией***…

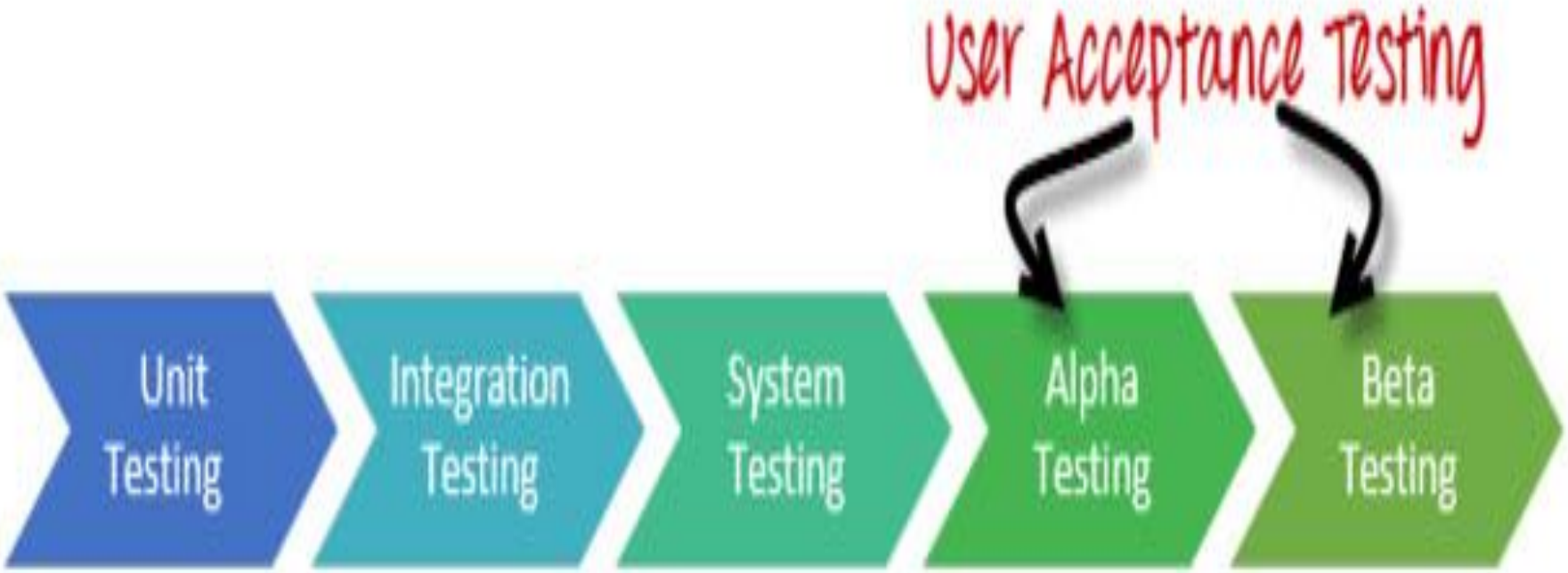
Альфа-тестирование — это этап отладки и проверки альфа-версии программы, а люди, которые будут заниматься ее тестированием — альфа-тестерами. Это могут быть штатные тестировщики компании или люди, которые работают по договору, но это квалифицированные специалисты, умеющие работать со специализированным программным обеспечением и пользоваться специальными методиками.

Понятие бета-тестирования

По окончании работы с альфа-версией выпускается ***бета-версия***. Она представляет собой реально работающую версию программы с полным функционалом. И задача бета-тестов – оценить возможности и стабильность работы программы с точки зрения ее будущих пользователей. Поэтому на роль бета-тестеров приглашаются просто люди, имеющие опыт работы с программами такого типа или, что еще лучше, с предыдущей версией этой же программы.

…Может быть объявлено открытое бета-тестирование, когда на роль бета-тестеров приглашают всех желающих…

Место альфа- и бета тестирования в испытании ПО



Сценарное тестирование

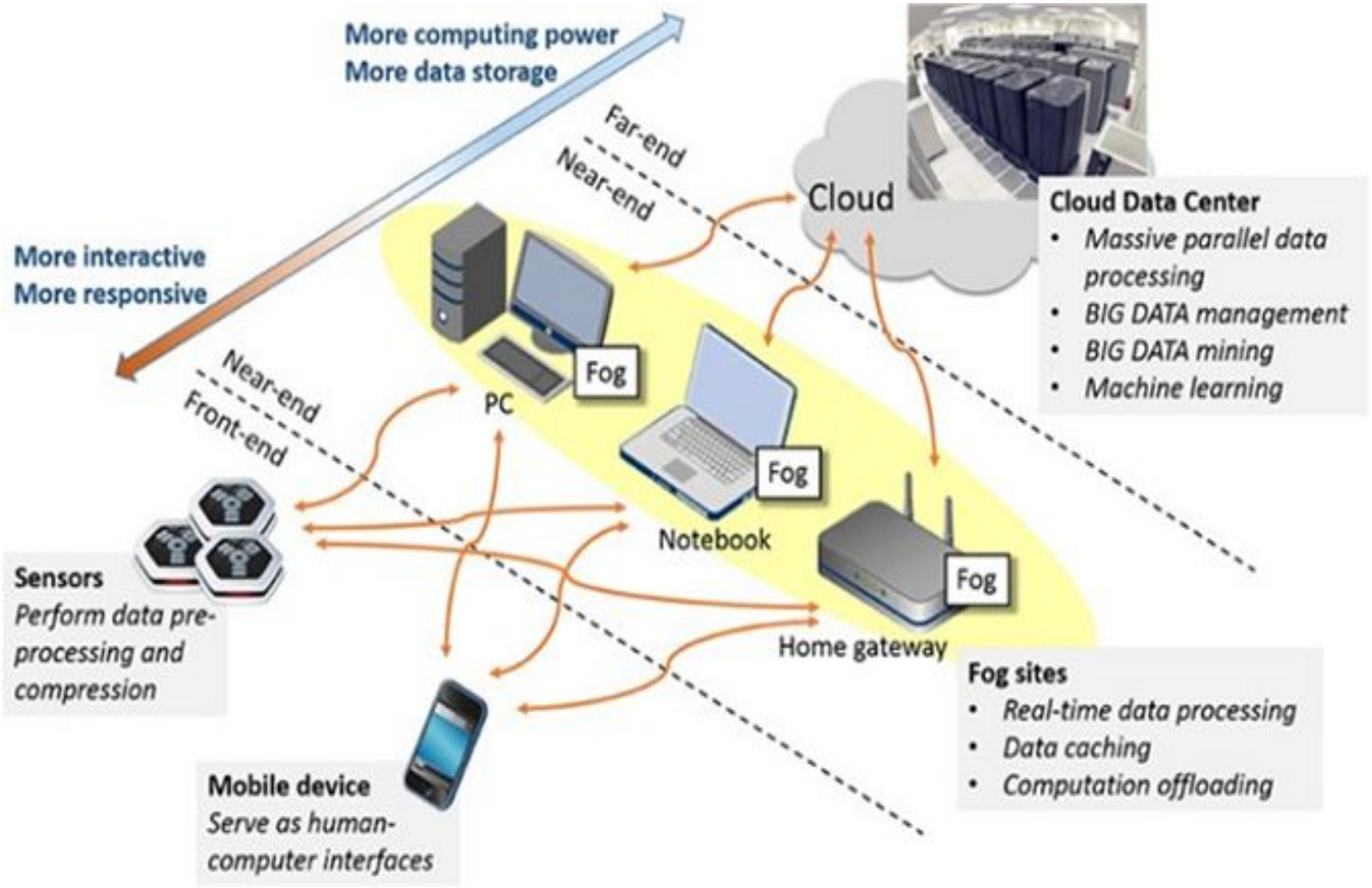
Сценарное тестирование- классическое тестирование по предварительно написанным и задокументированным сценариям.

В пользу сценарного тестирования:

сравнительная легкость планирования: тест-кейсы можно легко поделить между различными тестировщиками или командами.

Новые подходы к тестированию программных средств

Cloud, Fog, Edge computing



Ad hoc тестирование

Содержание Ad hoc тестирования

Свободное тестирование **(ad-hoc testing)** – это вид тестирования, который выполняется ***без подготовки к тестированию продукта, без определения ожидаемых результатов, проектирования тестовых сценариев***. Это неформальное, импровизационное тестирование. Такой способ тестирования в большинстве случаев дает большее количество заведенных отчётов об ошибке. Это обусловлено тем, что тестировщик на первых шагах приступает к тестированию основной функциональной части продукта и выполняет как позитивные, так и негативные варианты возможных сценариев.

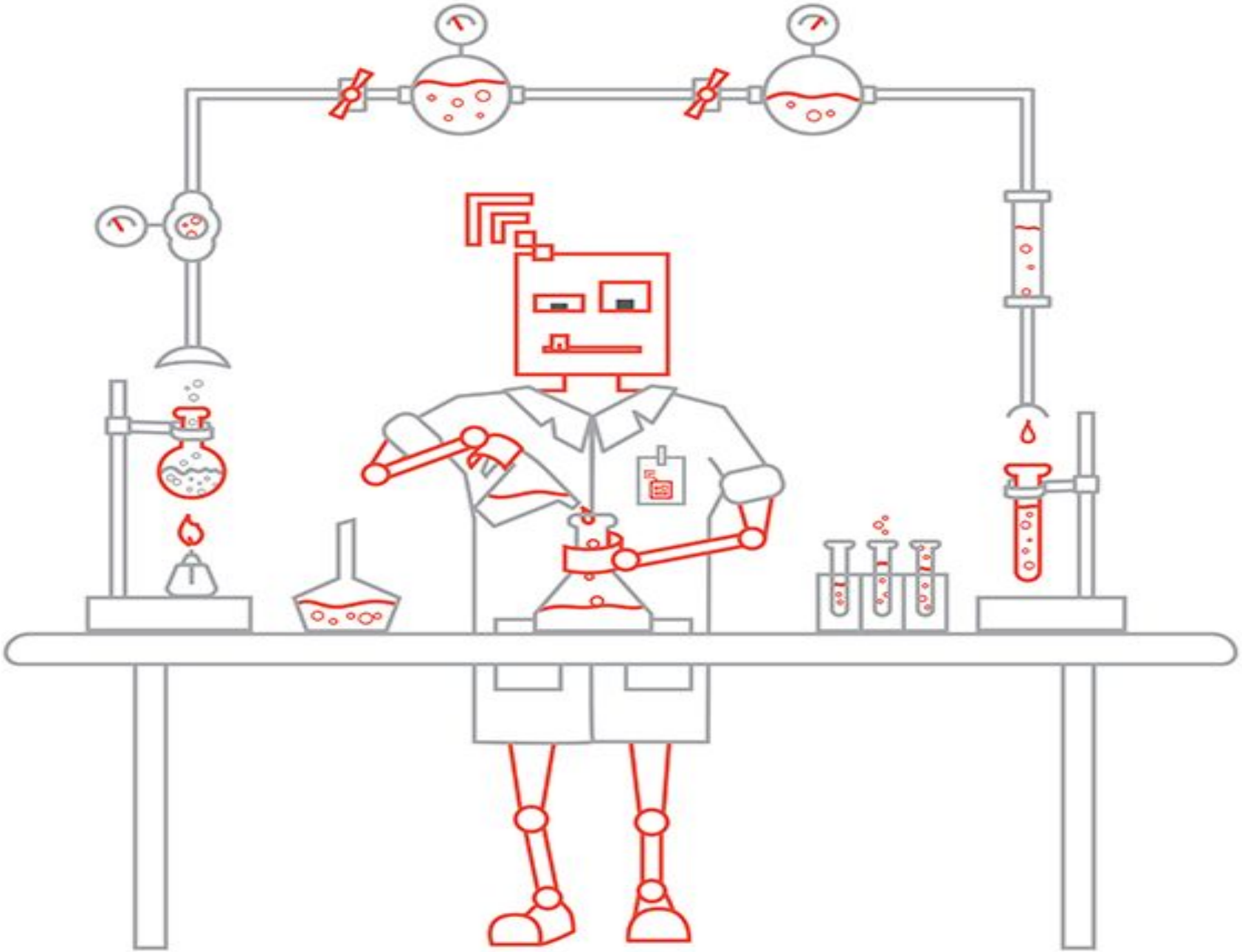
Виды свободного тестирования (ad-hoc testing)

- **Buddy testing** – процесс, когда 2 человека, как правило разработчик и тестировщик, работают параллельно и находят дефекты в одном и том же модуле тестируемого продукта. Такой вид тестирования помогает тестировщику выполнять необходимые проверки, а разработчику исправлять множество дефектов на ранних этапах.
- **Pair testing** – процесс, когда 2 тестировщика проверяют один модуль и помогают друг другу. К примеру, один может искать дефекты, а второй их документировать. Таким образом, у одного тестера будет функция, скажем так, обнаружителя, у другого – описателя.
- **Monkey testing** – произвольное тестирование продукта с целью как можно быстрее, используя различные вариации входных данных, нарушить работу программы или вызвать ее остановку (простыми словами – сломать).

Основные преимущества ad-hoc testing

- Нет необходимости тратить время на подготовку документации.
- Самые важные дефекты зачастую обнаруживаются на ранних этапах.
- Часто применяется, когда берут нового сотрудника. С помощью этого метода, человек усваивает за 3 дня то, что, разбираясь с тестовыми случаями, разбирал бы неделю - это называется форсированное обучение новых сотрудников.
- Возможность найти трудновоспроизводимые и трудноуловимые дефекты, которые невозможно было бы найти, используя стандартные сценарии проверок.

Исследовательское тестирование
(Exploratory testing)



Понятие исследовательского тестирования

Исследовательское тестирование **(exploratory testing)** — это одновременное изучение программного продукта, проектирование тестов и их выполнение. Это неформальный метод проектирования тестов, при котором тестировщик активно контролирует проектирование тестов и то, как эти тесты выполняются, и использует полученную во время тестирования информацию для проектирования новых тестов. Если каждый следующий тест, который выполняет тестировщик, выбирается по результатам предыдущего теста, это означает, что мы используем исследовательское тестирование.

Когда следует применять исследовательское тестирование?

- Когда нужно обеспечить быструю обратную связь для нового продукта или новой функциональности продукта.
- Когда нужно быстро ознакомиться с продуктом.
- Когда уже были проведены основные виды тестирования и время позволяет разнообразить методы тестирования.
- Когда нужно найти дефект, локализованный в определенном модуле в кратчайшие сроки.
- Когда проверяется работа другого специалиста по тестированию.
- Когда нужно изучить состояние конкретного риска для принятия решения о необходимости покрытия конкретной области тестами.

Предпосылки к использованию исследовательского тестирования в чистом виде

- **Мало времени** : если тестовая документация написана, но времени на прохождение тестов уже нет, нужно выбирать наиболее критичные области приложения, которые реально протестировать за имеющееся время. Составить чек-лист с идеями и тестировать вокруг них.
- **Сложности с требованиями**: требований нет, они не полны или устарели и нет возможности их актуализировать.
- **Небольшой проект**: продукт маленький, и разработка тестовых сценариев займет больше времени, чем сам процесс тестирования.

Использование исследовательского тестирование в дополнение к сценарному тестированию

- Тестировщики постоянно проходят одни и те же тестовые сценарии
- При многократном прохождении одних и тех же тестов, например, при регрессионном тестировании. Тестировщики теряют концентрацию и начинают пропускать дефекты. В этом случае исследовательское тестирование помогает взглянуть на проект под новым углом и найти пропущенные дефекты.
- Тестировщик отвлекается от шаблонных действий и чувствует себя в большей степени обычным пользователем. Это помогает найти дефекты, сильнее влияющие на конечного потребителя разрабатываемого продукта.

Использование исследовательского тестирование в дополнение к сценарному тестированию (продолжение)

- Пришел внезапный запрос на изменения
  Времени на разработку новых сценариев нет, так как все заняты другими запланированными задачами или изменения потребуют переработать большую часть документации. В этой ситуации тестирование исследовательским методом может быть наиболее оптимальным.
- Когда хочется перестраховаться
  Продукт уже протестирован по сценариям, но всё еще хочется убедиться в том, что ничего не было упущено.

Когда одним исследовательским тестированием не обойтись

- Приложение стандартизованное: Приложения, работающие по стандартам и гостам, а также системы, для которых малейшее отклонение может быть критичным. Это могут быть приложения, отвечающие за полеты ракет или проводящие финансовые операции.
- Проводится интеграционное тестирование. В этом случае исследовательское тестирование возможно, например, при тестировании API. Но обычно интеграционное тестирование проводится для проверки взаимодействия внутренних компонентов приложений. Эта работа хорошо покрыта документацией и часто автоматизируется

Когда одним исследовательским тестированием не обойтись (продолжение)

- Тестовые сценарии отдаются на в стороннюю организацию.
  Контролировать поставленную задачу и процент ее выполнения проще по формализованным сценариям.
- Длительный проект. Тестировщики могут быть подключены к проекту на время определенной фазы, а после, пока разработчики реализовывают новый функционал, заниматься другими проектами. Если долго не тестировать конкретную функциональность, то ее специфика забывается

Системное сочетание исследовательского и сценарного тестирования

Исследовательское тестирование — не означает полное отсутствие документации и хаос, а является мощным инструментом.

Используя ранжирование типов тестирования от полностью исследовательского до полностью сценарного, можно подобрать оптимальный уровень документации для вашего проекта и сэкономить время.

Сценарное и исследовательское тестирование являются полностью совместимыми и компенсируют недостатки друг друга. Можно покрыть детальными тестами сложные технические аспекты проекта и написать поверхностные чек-листы для пользовательского интерфейса.


Regression
Testing

Регрессионное тестирование
(Regression testing)

Регрессионное тестирование - это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, web сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде.

Регрессионными могут быть как функциональные так и нефункциональные тесты.

Как правило, **для регрессионного тестирования используются тест кейсы, написанные на ранних стадиях разработки и тестирования**. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность.

Рекомендуется делать автоматизацию регрессионных тестов, для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения.

Содержание регрессионного тестирования

Регрессионное тестирование **(regression testing)** – это механизм проверки, который направлен на обнаружение различных проблем в уже проверенных участках программ. Делается это не для окончательного убеждения в отсутствии неработающих участков кода, а чтобы найти и исправить регрессионные ошибки. Под ними понимают баги, которые появляются не во время написания программы, а при добавлении новых участков кода или исправлении допущенных ранее промахов в синтаксисе кода.

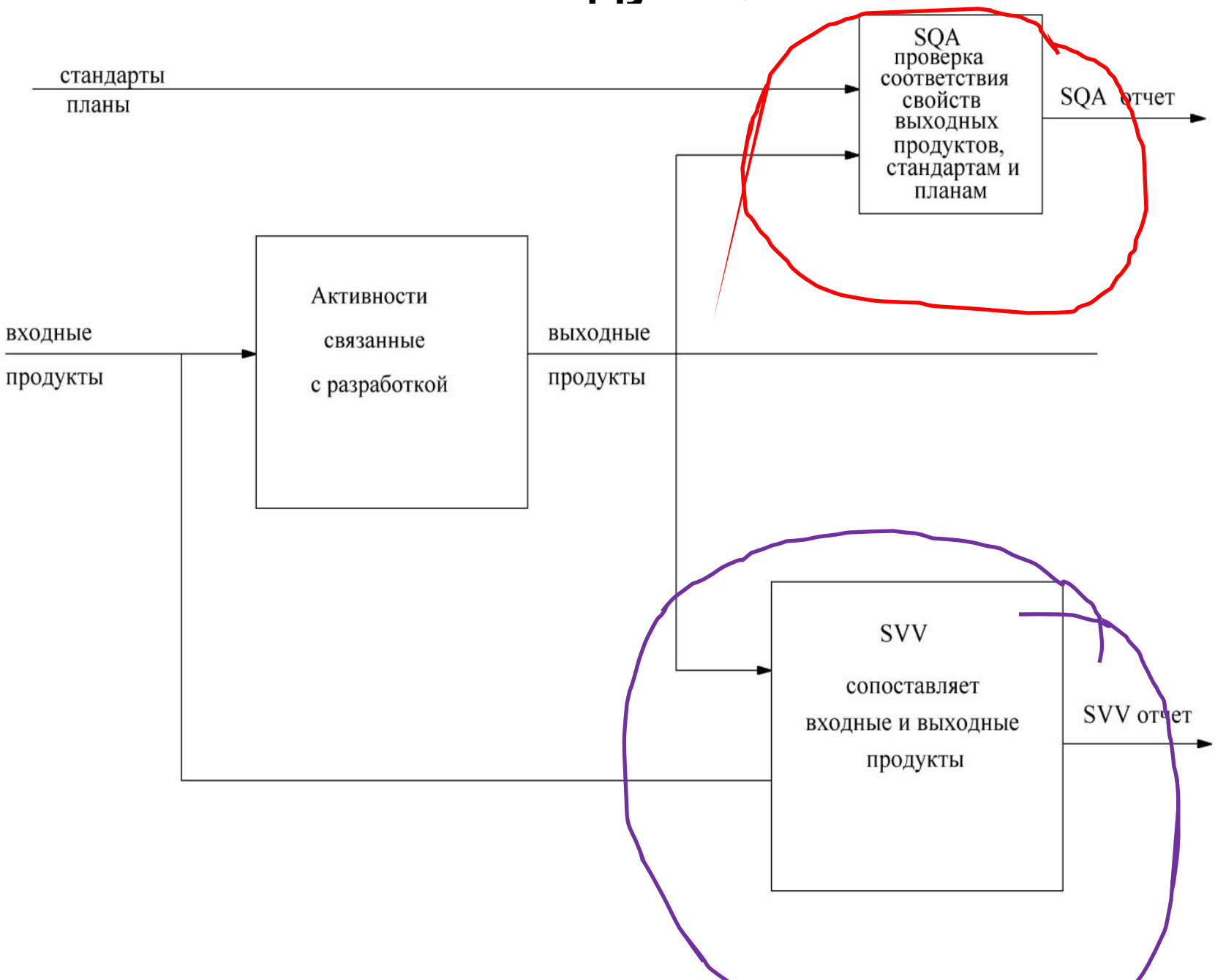
Тема : Стандартизация – путь к успеху программных проектов

ESA PSS-05-11 Issue 1 Revision 1
March 1995

Guide to software quality assurance

Prepared by:
ESA Board for Software Standardisation and Control (BSSC)

Различие между SQA и SVV



Вывод:

Следование положениям стандартов обеспечивает «правильную» реализацию программного проекта в случае стабильного состояния объекта управления

Испытания обеспечивают контроль соответствия промежуточных и конечных результатов проекта заранее определенным требованиям

Направления стандартизации ЖЦ СОД и У

1. Организуется и стимулируется Международной организацией по стандартизации (ISO – International Standard Organization) и международной комиссией по электротехнике (IEC – International Electrotechnical Commission). Стандартизации подвергаются наиболее общие технологические процессы, имеющие значение для международной кооперации

Направления стандартизации ЖЦ СОД и У (продолжение)

2. Направление, развиваемое институтом инженеров электротехники и радиоэлектроники (IEEE – Institute of Electrotechnical and Electronics Engineers) совместно с Американским национальным институтом стандартизации (American National Standards Institute – ANSI).

Стандарты ISO\IEC и ANSI / IEEE в основном имеют рекомендательный характер

Направления стандартизации ЖЦ СОД и У (продолжение)

3. Третье направление стимулируется министерством обороны США (Department of Defense – DOD). Стандарты DOD имеют обязательный характер для фирм, работающих по заказу Министерства обороны США.

Структура стандартов ESA PSS-05-XX

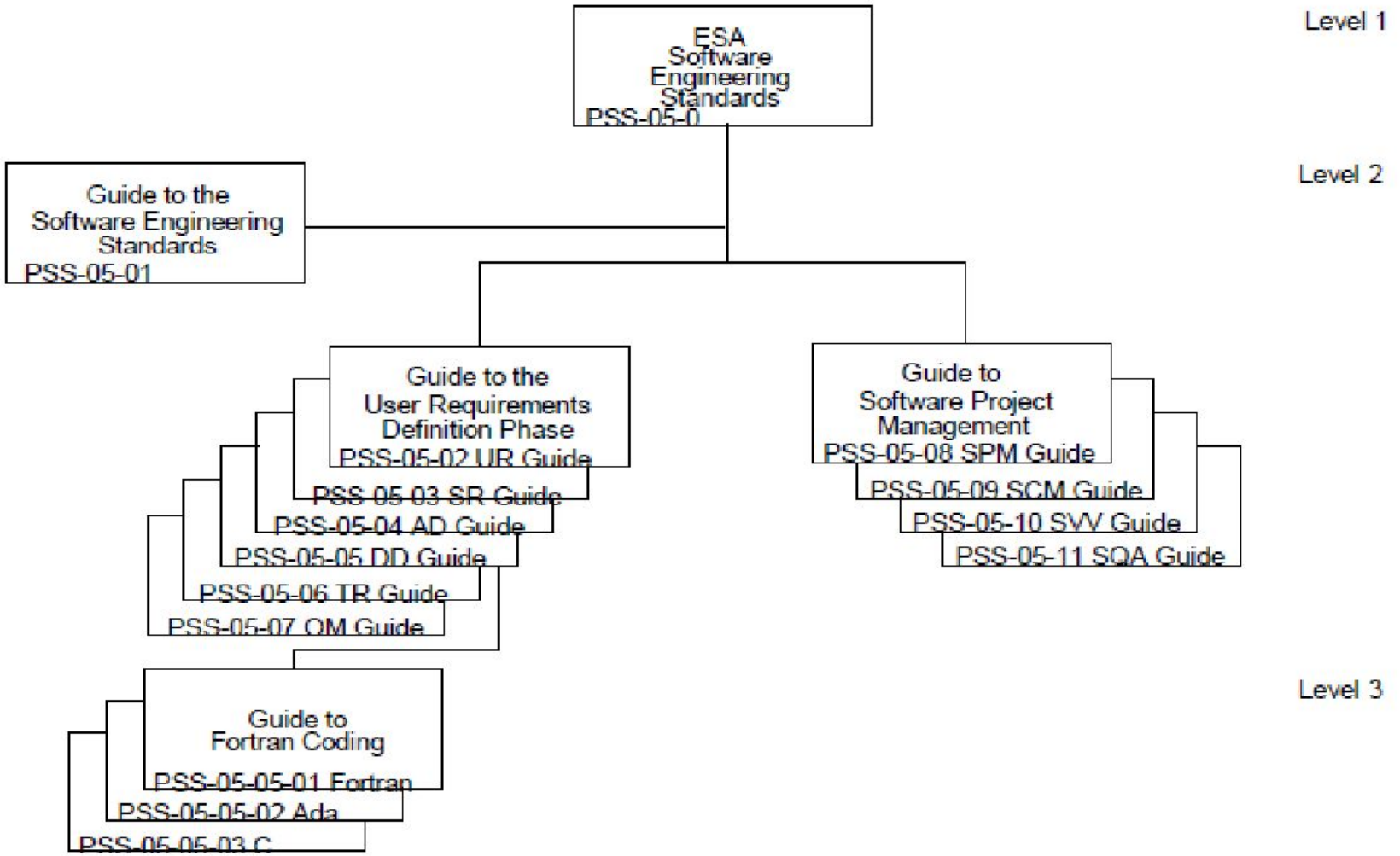
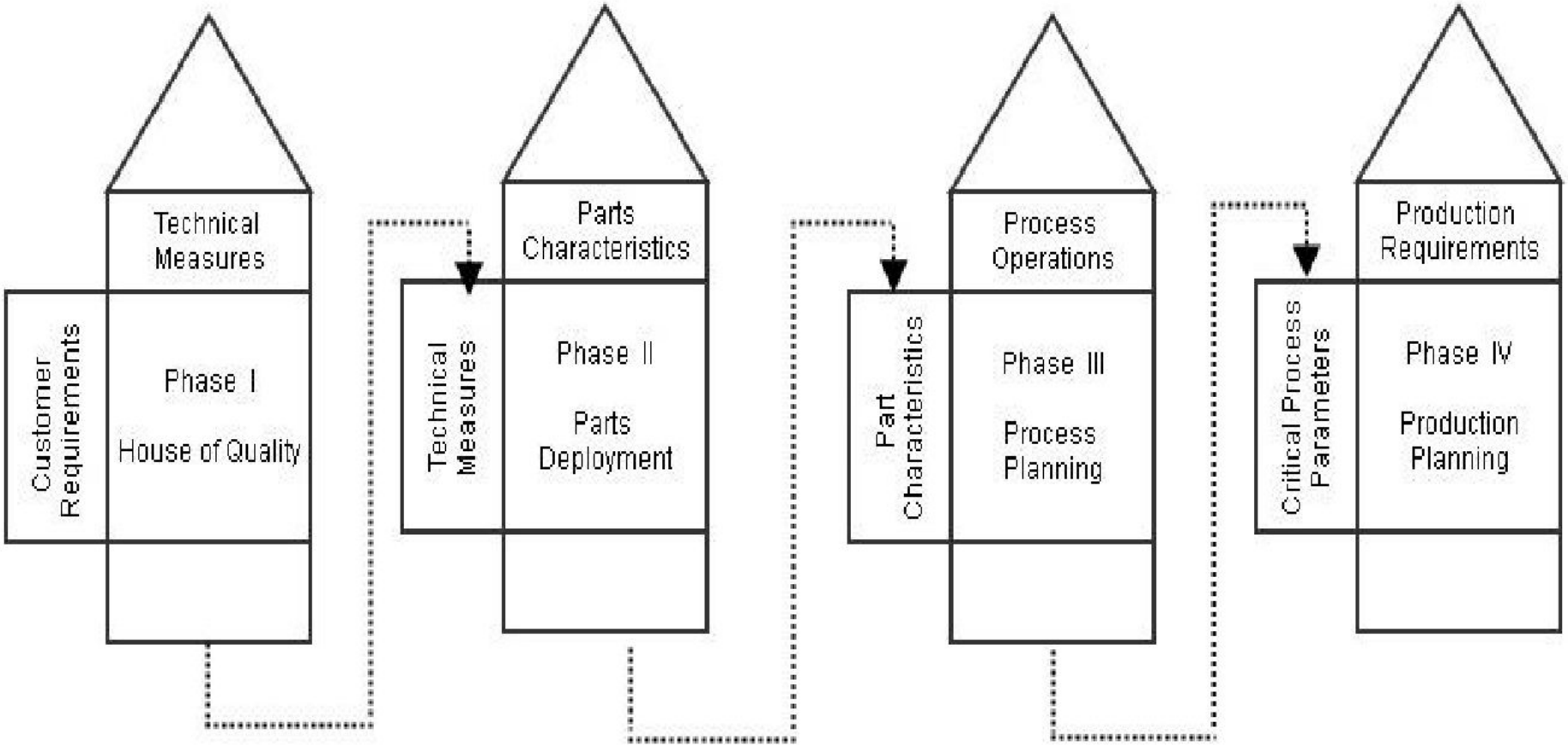


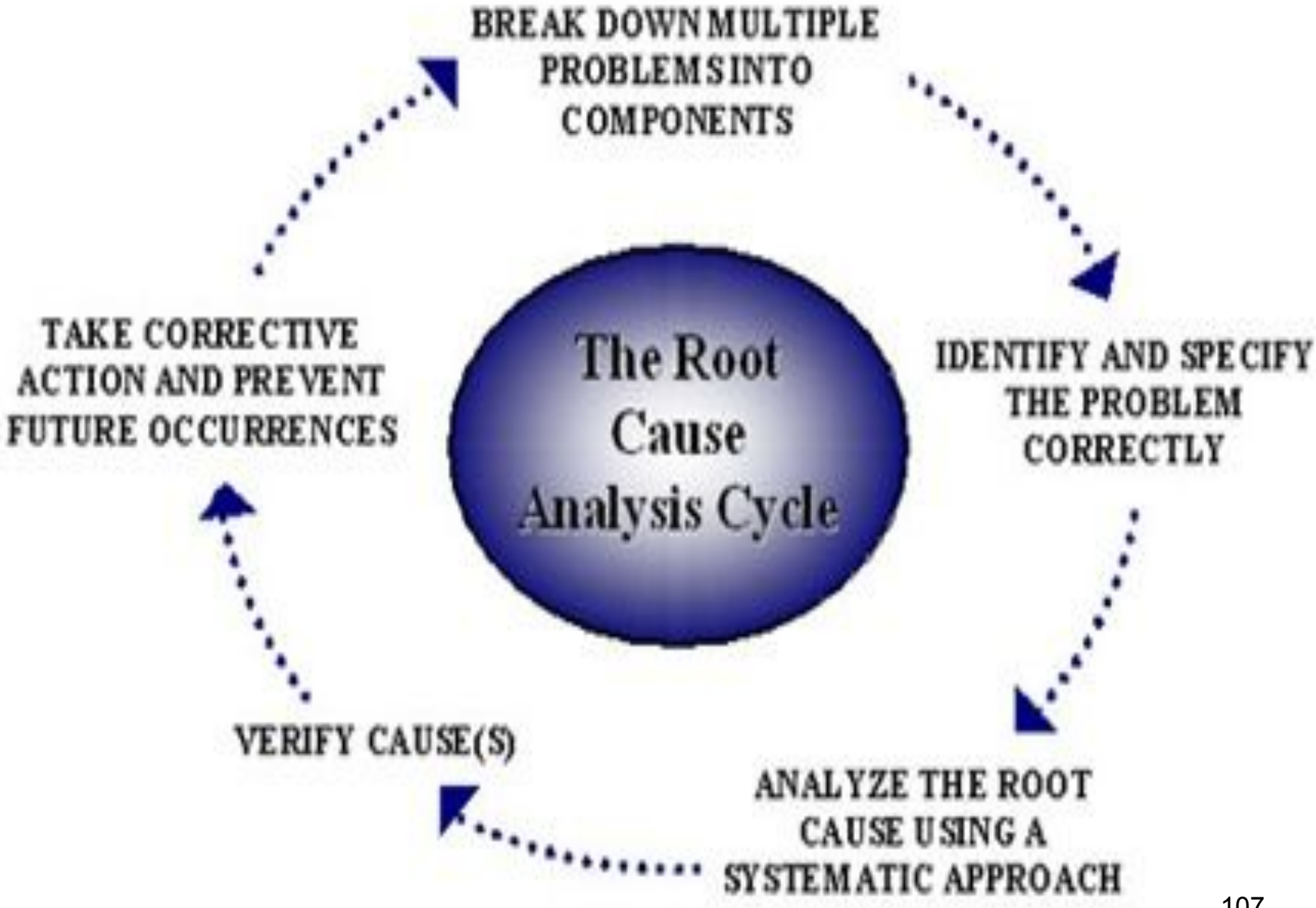
Figure 1: The ESA PSS-05-0 Document Tree

Методология «развертывание функций качества» - Quality Function Deployment-QFD

Четырехфазная модель American Supplier Institute-ASI



Анализ коренных причин (Root Cause Analysis)


BREAK DOWN MULTIPLE PROBLEMS INTO COMPONENTS
IDENTIFY AND SPECIFY THE PROBLEM CORRECTLY
ANALYZE THE ROOT CAUSE USING A SYSTEMATIC APPROACH
VERIFY CAUSE(S)
TAKE CORRECTIVE ACTION AND PREVENT FUTURE OCCURRENCES
The Root Cause Analysis Cycle


What ?
Where ?
Root
Cause
How ?
When ?
Who ?
Why ?

Принципы SMART

Решение задач в рамках RCA основано на принципах SMART :

Specific :учет специфики проблемы;

Measurable: измеримость;

Action Oriented: ориентация на действия;

Realistic: реалистичность;

Time Constrained: ограниченная продолжительность

Краткое описание содержания задач RCA

1. **Определение проблемы**.

Корректное определение проблемной ситуации является критически важным фактором успеха RCA. Если проблема корректно не определена, использование RCA завершится провалом. Результатом решения этого класса задач является план проекта урегулирования проблемной ситуации.

Краткое описание содержания задач RCA (продолжение)

2. Понимание проблемы.

Основой решения этого класса задач является структуризация проблемы, например, за счет построения совокупности когнитивных карт (моделей). Результатом решения этого класса задач является ясное понимание существа проблемы . Для решения задач этого класса могут использоваться различные инструменты, такие, например, как анализ причинно – следственных связей; мозговой штурм и т.д.

Краткое описание содержания задач RCA (продолжение)

3. **Немедленное действие**.

Фокусом этой задачи является незамедлительная реализация мер по урегулированию проблемной ситуации. При этом исходят из того, что чем дальше решение от источника проблемы, тем меньше шансов, что деятельность будет эффективной.

Краткое описание содержания задач RCA (продолжение)

4. **Корректирующие действия**.

Определение и ранжирование наиболее возможных причин проблемы. Фундаментальной предпосылкой является то, что временные меры не устраняют коренных причин проблемы. Реализация корректных мер, как минимум смягчает, а как максимум устраняет коренные причины.

Краткое описание содержания задач RCA (продолжение)

5. **Подтверждение правильности решения**. После определения способов воздействия на коренные причины и их реализации, необходимо убедиться в их действенности. Наличие подтверждения успешности предлагаемых решений создает основу реализации действий, предупреждающих возникновения проблемы в последующем.

Базовые положения RCA

1. RCA должен выполняться на систематической основе; допущения, возможные причины и заключение должны подтверждаться данными и документально оформленными событиями.

2. Эффективность RCA обеспечивается выявлением и анализом всех доступных связей между коренными причинами и выделенной проблемой.

3. В большинстве случаев каждой проблеме может быть поставлена в соответствии множество коренных причин.

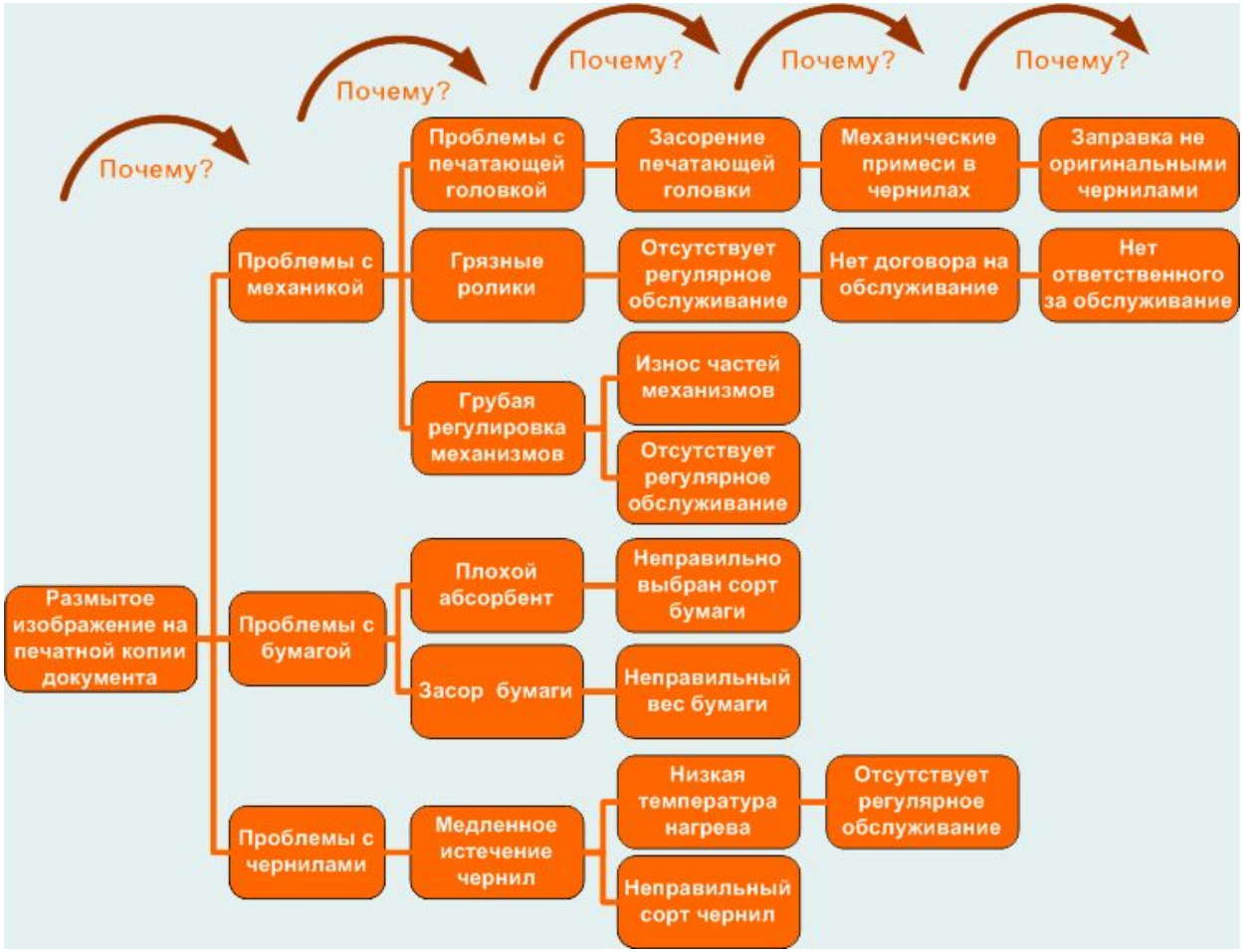
Базовые положения RCA

4. Должны быть либо понятны отношения между коренными причинами. Выявление этих отношений позволит понять как они влияют друг на друга, а также может послужить предметом дискуссии экспертов, что, в свою очередь, является основой построения онтологической модели ситуации.

5. Потенциальные варианты решения проблемы должны быть проанализированы с двух позиций: насколько в долговременной перспективе они решают проблемы; в какие экономические затраты для организации выливается реализация

Инструментарий и технологии RCA.

«Пять Почему?»


Почему?
Почему?
Почему?
Почему?
Почему?
Размытое изображение на печатной копии документа
Проблемы с механикой
Проблемы с бумагой
Проблемы с чернилами
Проблемы с печатающей головкой
Грязные ролики
Грубая регулировка механизмов
Засорение печатающей головки
Отсутствует регулярное обслуживание
Износ частей механизмов
Отсутствует регулярное обслуживание
Механические примеси в чернилах
Нет договора на обслуживание
Заправка не оригинальными чернилами
Нет ответственного за обслуживание
Плохой абсорбент
Засор бумаги
Неправильно выбран сорт бумаги
Неправильный вес бумаги
Медленное истечение чернил
Низкая температура нагрева
Неправильный сорт чернил
Отсутствует регулярное обслуживание

Парето – анализ

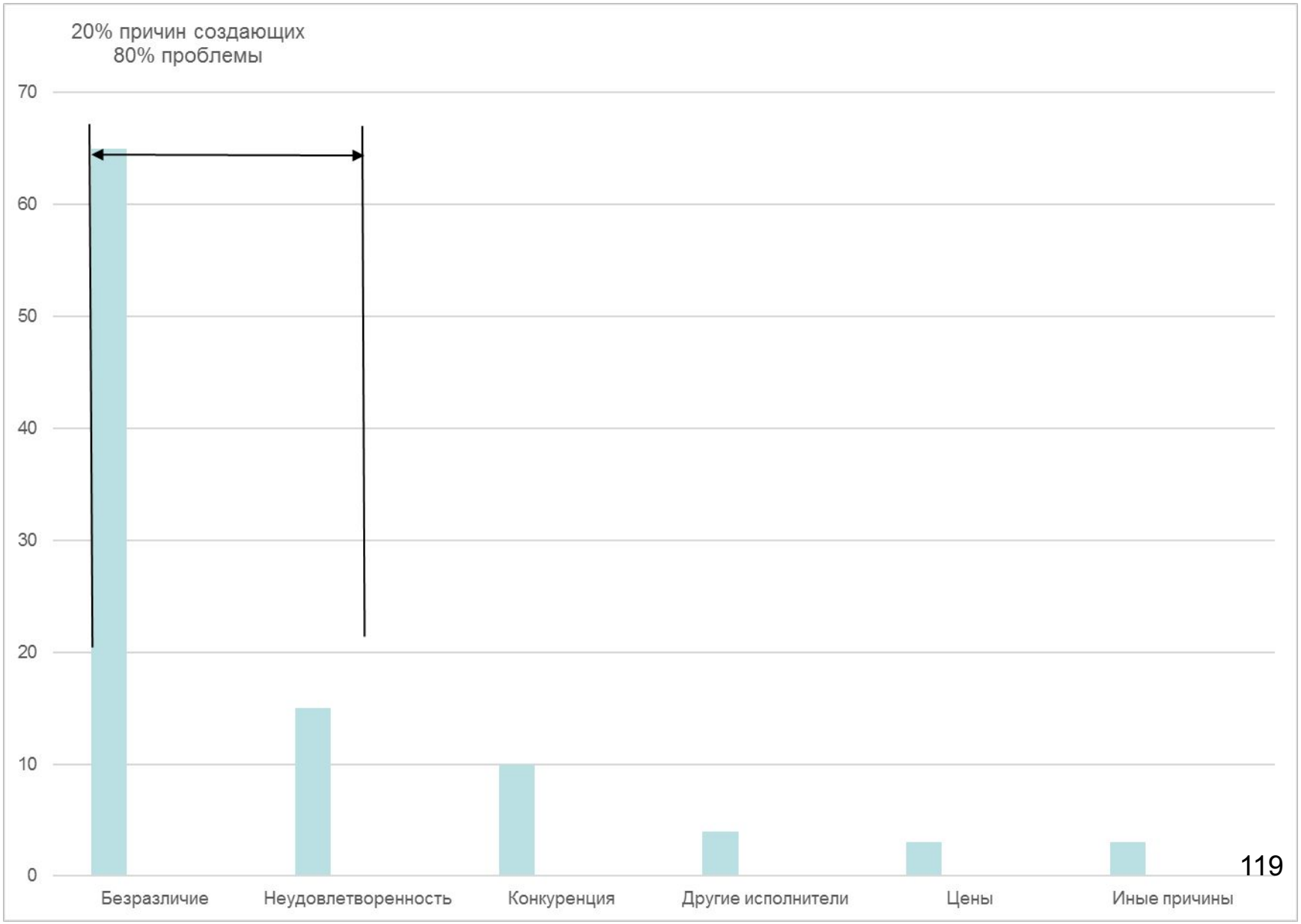
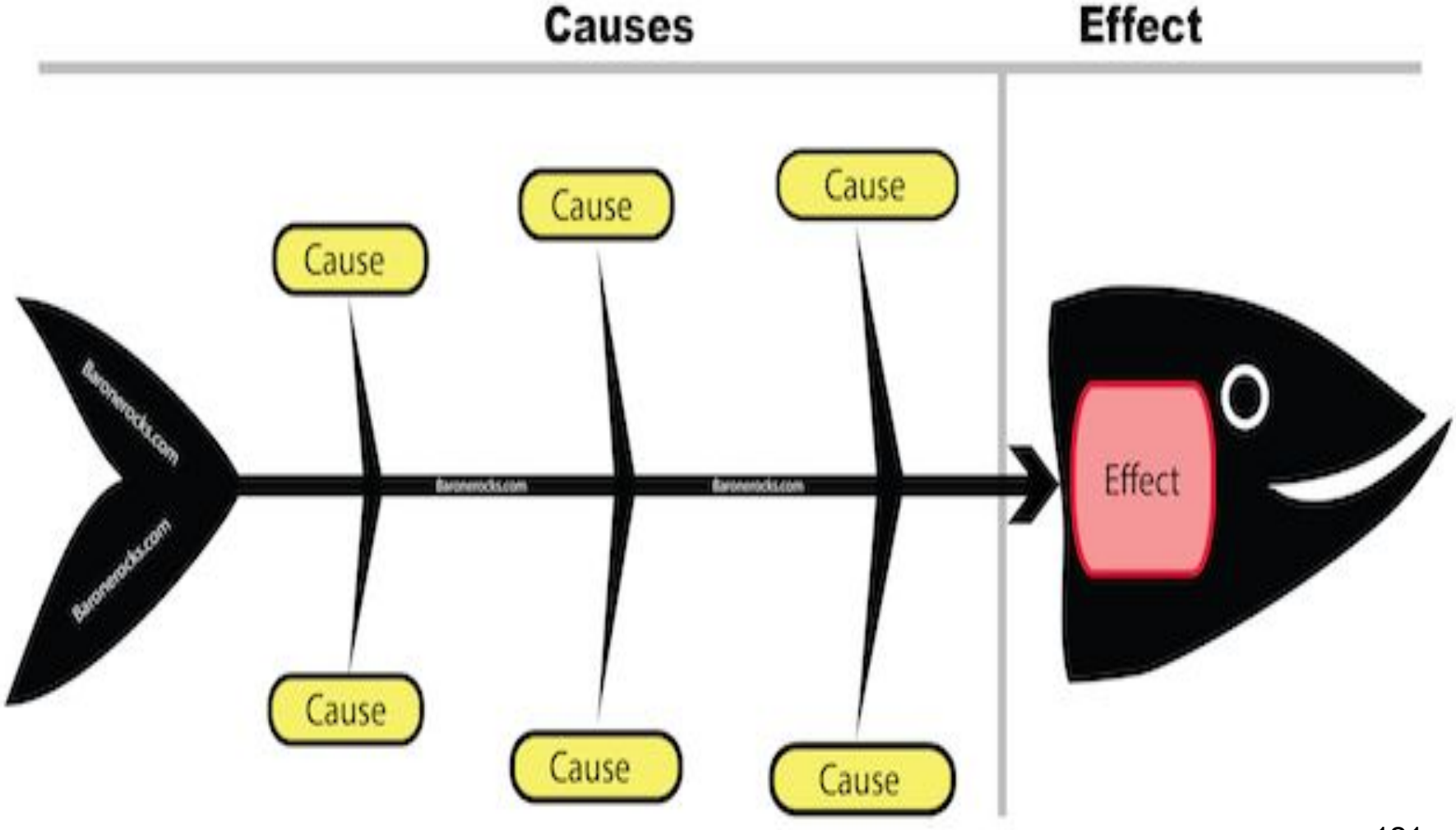
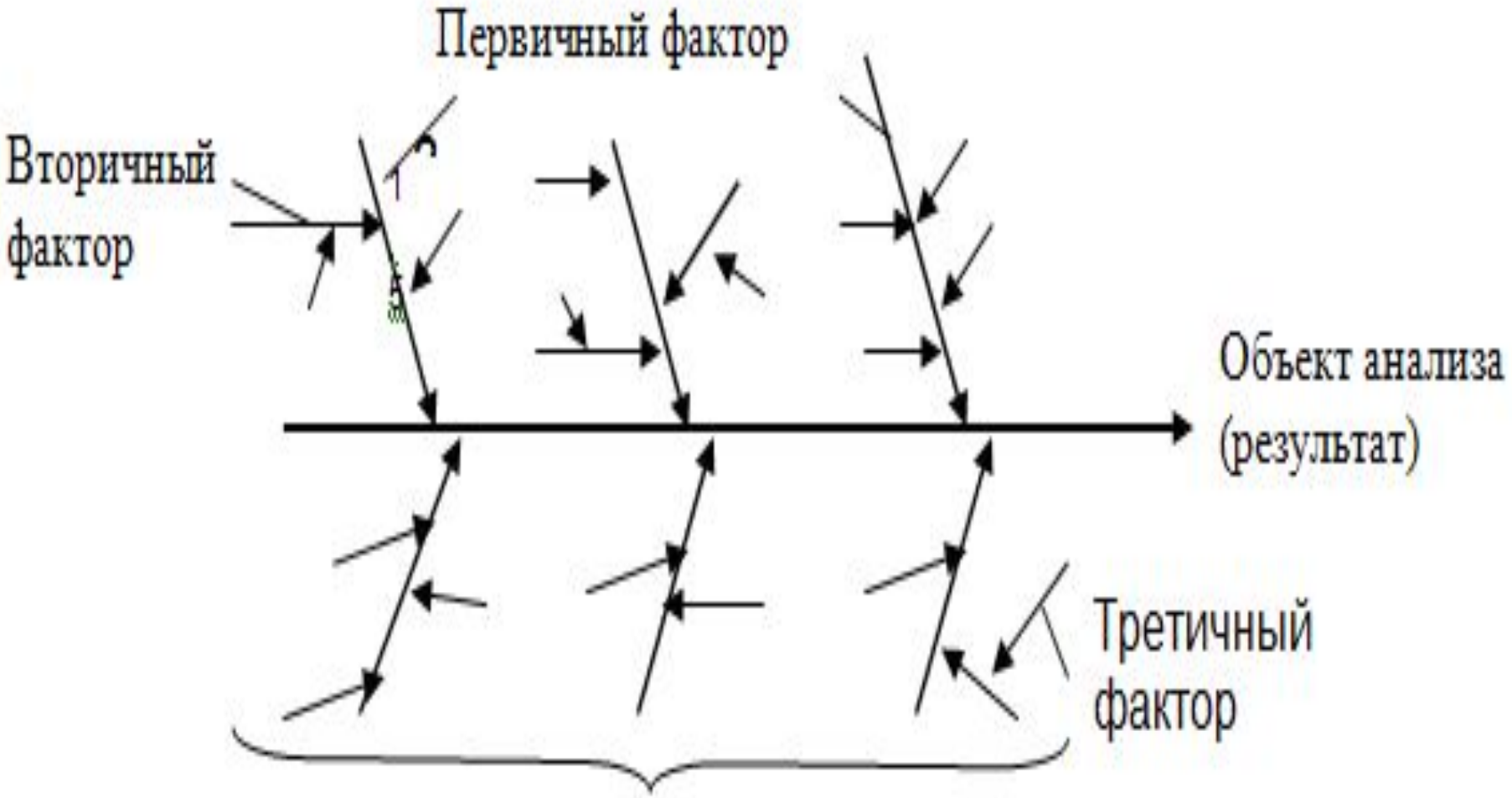


Диаграмма причинно – следственных связей

Fishbone Diagram

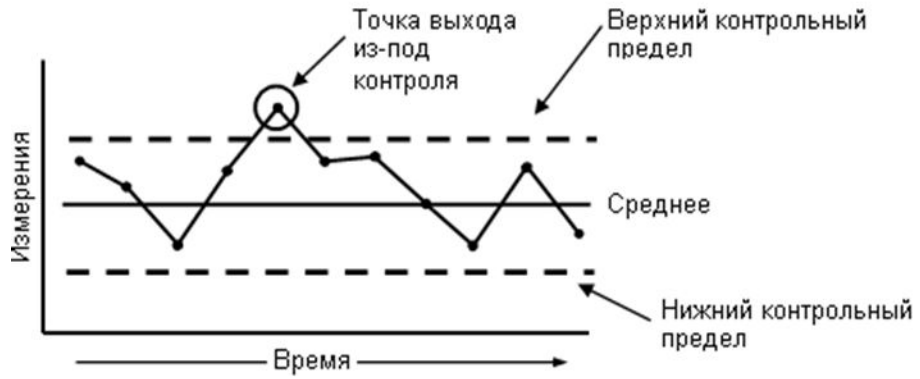




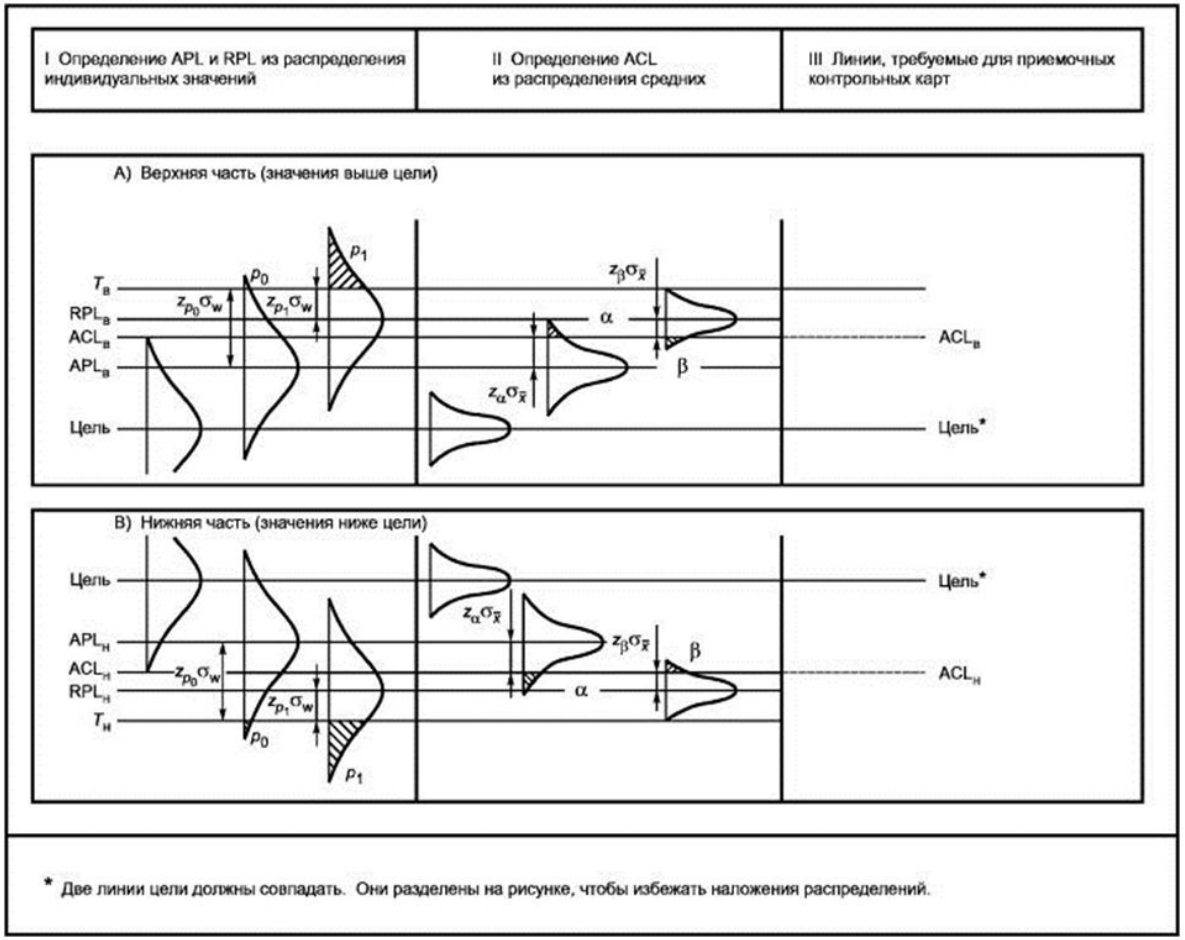
Причинные факторы (характеристики)

Принцип построения диаграммы Исикавы

Контрольные диаграммы (отдельный процесс)



Контрольные диаграммы (совокупность процессов)



Тема: Реализация системного подхода при испытаниях программных систем

ESA PSS-05-10 Issue 1 Revision 1
March 1995

Guide to software verification and validation

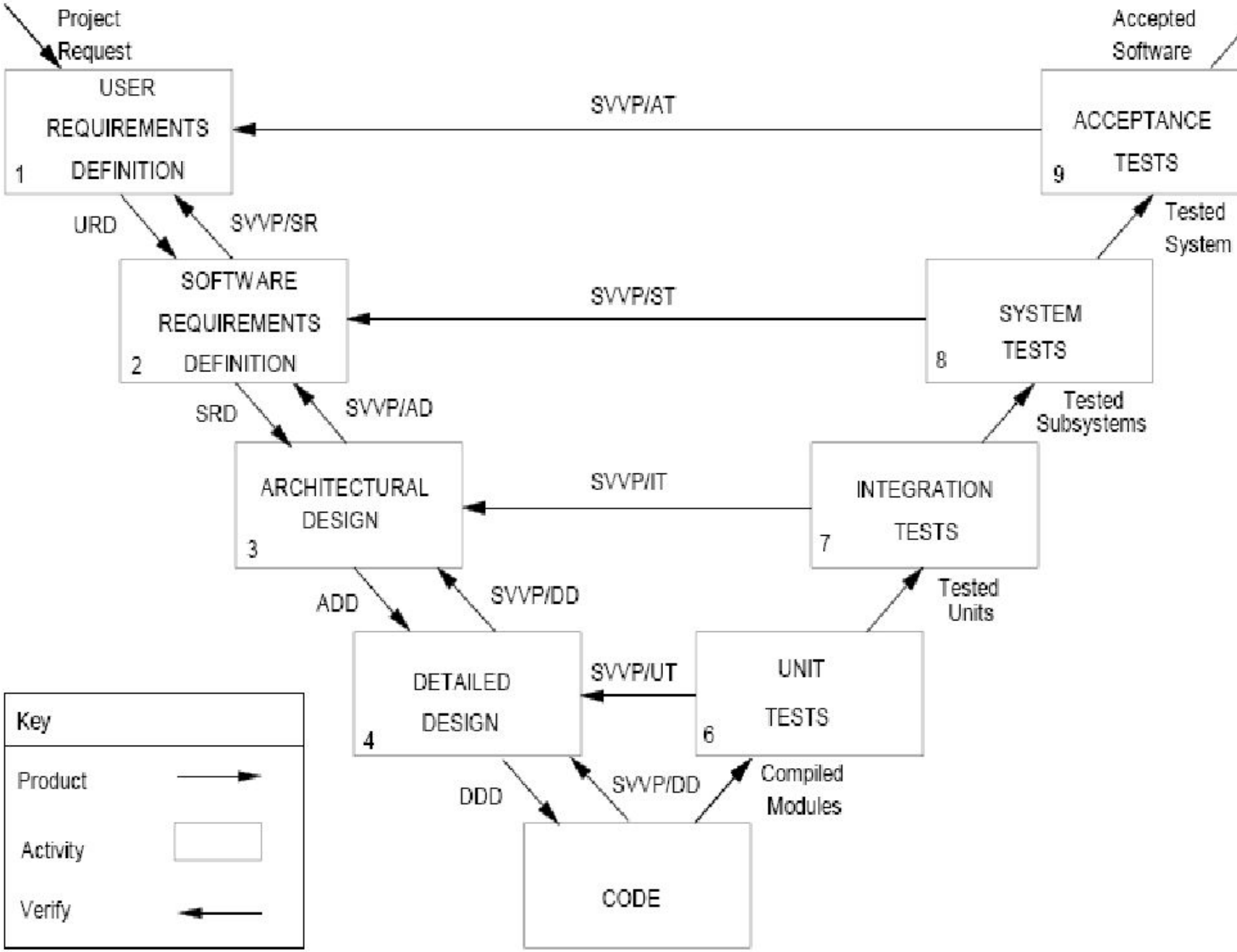
Prepared by:
ESA Board for Software Standardisation and Control (BSSC)

IEEE Std 1012 - 2004

IEEE Standard for Software Verification and Validation

Recognized as an American National Standard (ANSI)

Life cycle verification approach



Структурирование вариантов использования на основе целей пользователей

1. Вариант использования представляет собою текстовое описание шагов, которые выполняет актор
2. Вариант использования не сводится к описанию функциональности системы, что составляет основу работы программистов
3. Вариант использования состоит из двух частей: описание последовательности действий в штатном режиме использования системы и описание того, как ведет себя система когда происходит сбой

Формальные приемы структурного анализа

1. Количество ошибок в программном модуле прямо пропорционально сложности модуля

   Источник: В.В.Липаев Анализ и сокращение рисков проектов сложных программных средств. М.: СИНТЕГ, 2005

2. Цикломатический показатель сложности структуры

   Источник: ESA PSS-05-10

МОДЕЛЬ CMM

Пять уровней зрелости CMM

Начальный уровень

Уровень 1 - означает, что процессы создания ПО на предприятии не формализованы. Они не могут строго планироваться и отслеживаться, их успех носит случайный характер. Результат работы целиком

и полностью зависит от личных качеств отдельных сотрудников. При увольнении таких сотрудников проект останавливается.

повторяемый уровень

Уровень 2. Внедрены внедрить формальные процедуры выполнения основных этапов процесса разработки. Результаты выполнения процесса соответствуют заданным требованиям и стандартам. Основное отличие от уровня 1 состоит в том, что выполнение процесса планируется и контролируется. Применяемые средства

планирования и управления дают возможность повторения ранее достигнутых успехов.

Определенный уровень

Уровень 3. Требует, чтобы все элементы процесса были определены, стандартизованы и задокументированы. Основное отличие от уровня 2 заключается в том, что все этапы процесса уровня 3 планируются и управляются на основе единого стандарта предприятия. Качество разрабатываемого ПО уже не зависит от способностей отдельных личностей.

Управляемый уровень

Уровень 4. На предприятии используются

количественные показатели качества как программных продуктов, так и процесса. Это обеспечивает более точное планирование проекта и контроль качества его результатов. Основное отличие от уровня 3 состоит в более объективной, количественной оценке продукта и процесса.

Оптимизирующий уровень

Уровень 5. Подразумевает, что главной задачей предприятия становится постоянное улучшение и повышение эффективности существующих процессов, ввод новых технологий. Основное отличие от уровня 4 заключается в том, что технологии разработки и сопровождения программных продуктов планомерно и последовательно совершенствуются

КОНЕЦ ЛЕКЦИЙ