

SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI



Управление на софтуерните проекти

ЛЕКЦИЯ, КУРС: УПРАВЛЕНИЕ НА ПРОЕКТИ, СИ 2021

ИЗГОТВИЛ: ДИЛЯН ГЕОРГИЕВ, ДОКТОРАНТ „УПРАВЛЕНИЕ НА ЗНАНИЯ“

Съдържание

- **ВЪВЕДЕНИЕ**
- **МОДЕЛИ ЗА РАЗРАБОТКА НА СОФТУЕР**
- **РОЛЯТА НА МЕТОДОЛОГИИТЕ**
- **СОФТУЕРНА ДОКУМЕНТАЦИЯ**
- **ВЪПРОСИ И ДИСКУСИИ**

Необходимостта от управление в ИТ

С нарастването на мащаба, в която индустрията се развива, нарастват и трудностите, пред които се изправят екипите, участващи в разработването на софтуер. От гледна точка на мениджмънт, тези трудности резултат в загуби, некачествени продукти и/или недоволни клиенти.

През 1994 година компанията The Standish Group да направи проучване сред 365 ИТ компании, според което 31% от проектите не завършват и биват спирани преждевременно. 53% пък биват завършвани с надвишен бюджет или не навреме. Естествено се взима предвид големината на проекта, сложността на функционалностите и възвращаемостта

Необходимостта от управление в ИТ

<i>Company Size</i>	<i>Average Cost of Development</i>	<i>Average Cost Overruns</i>	<i>Average Schedule Overrun</i>	<i>Original Features and Functions Included</i>	<i>Successful Projects^a</i>	<i>Challenged Projects^b</i>	<i>Impaired Projects^c</i>
Large	\$2,322,000	178%	230%	42%	9%	61.5%	29.5%
Medium	\$1,331,000	182%	202%	65%	16.2%	46.7%	37.1%
Small	\$ 434,000	214%	239%	74%	28%	50.4%	21.6%

^a Completed on-time and on-budget

^b Completed, but over-budget, over schedule, and includes fewer features and functions than originally envisioned

^c Cancelled before completion

SOURCE: Adapted from The Standish Group, *CHAOS* (West Yarmouth, MA: 1995), <http://www.standishgroup.com/visitor/chaos.htm>.

Промените 1994-2015

Може да се каже, че провалените проекти намаляват и преминават към завършените такива с надвишен бюджет, време или човешки ресурси.

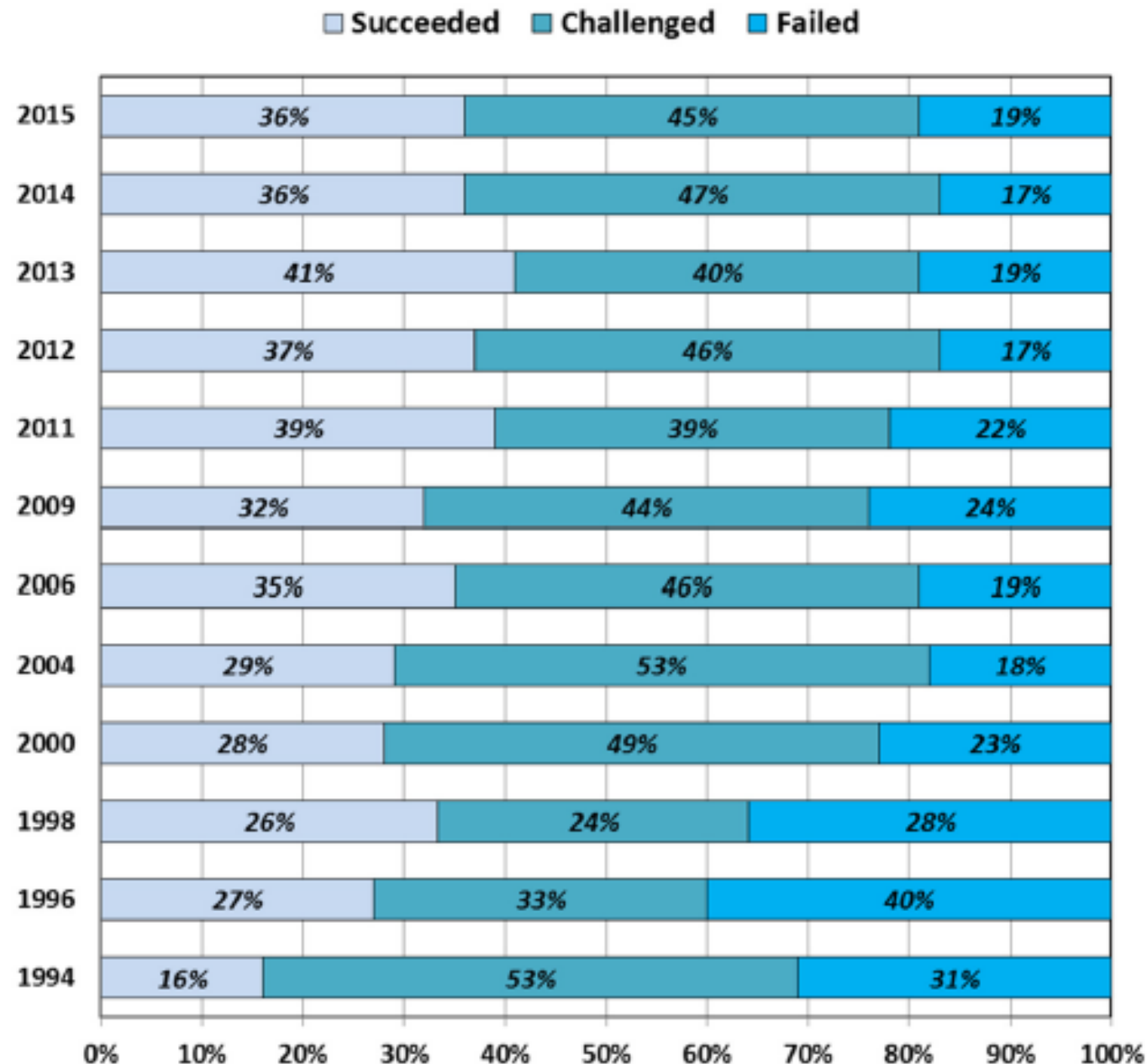
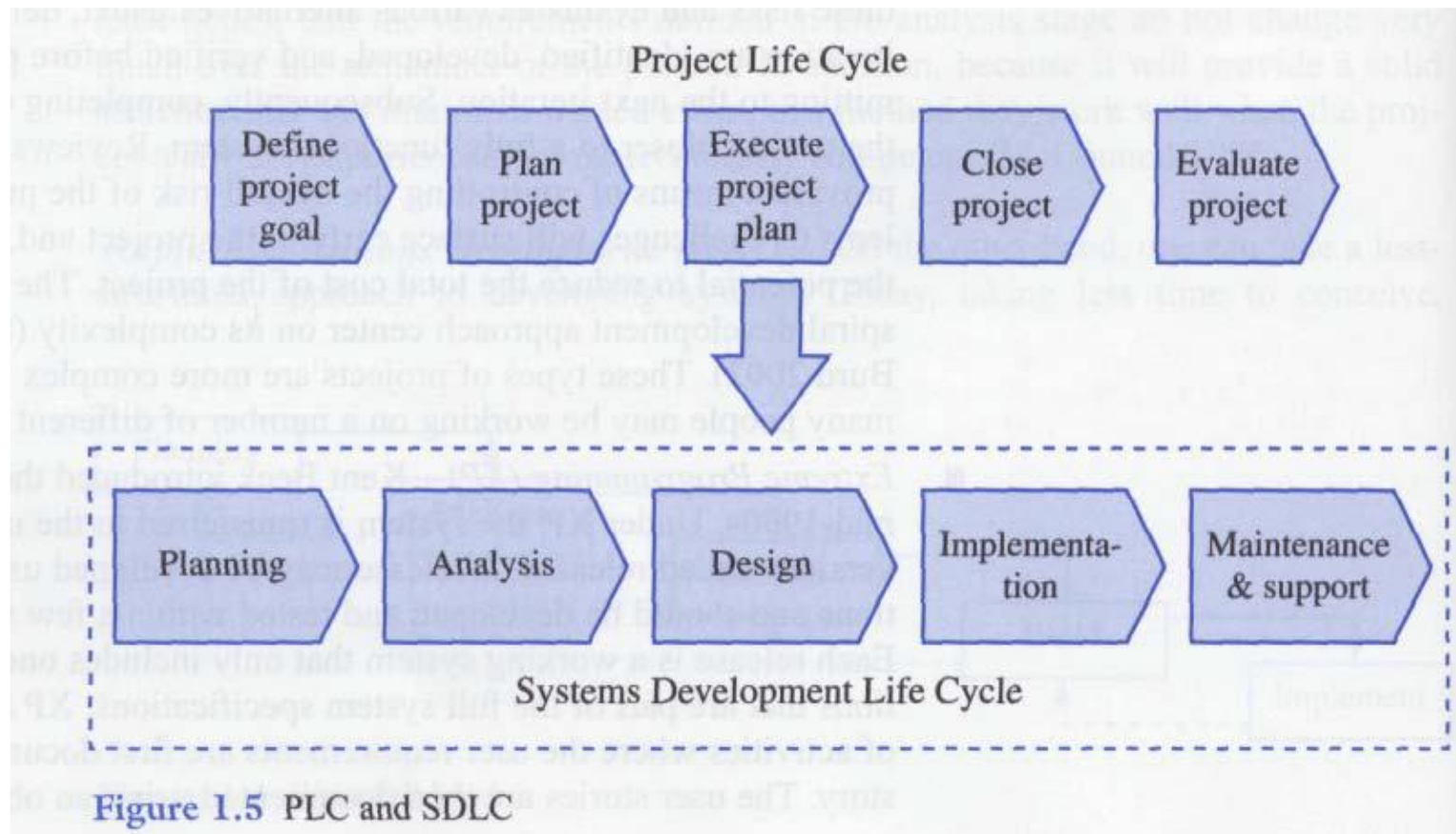


Figure 36 – Standish Group CHAOS Report Project Outcome Results 1994-2015

Success Factor	1994	1999	2000	2015
	Factor Importance Score			
User Involvement	19	20	16	15
Executive Management Support/ Executive Sponsorship	16	15	18	15
Emotional Maturity (Managing Expectations, Gaining Consensus)				15
Optimisation (Clarify Objective, Divide Larger Projects Into Multiple Smaller Projects)				15
Clear Statement of Requirements	15			
Firm Basic Requirements		5	6	
Clear Vision and Objectives/Clear Business Objectives	3	15	12	4
Proper Planning	11	5		
Reliable Estimates			5	
Realistic Expectations	10			
Smaller Project Milestones	9	10		
Minimised Scope			10	
Modest Execution				6
Standard Software Infrastructure			8	
Standard Architecture				8
Formal Methodology			6	
Agile Process				7
Competent Staff/Skilled Resources	8	5		10
Experienced Project Manager/Project Management Expertise		15	14	5
Ownership	6	5		
Hard-Working, Focused Staff	3			
Other		5	5	
Total Success Factor Score	100	100	100	100

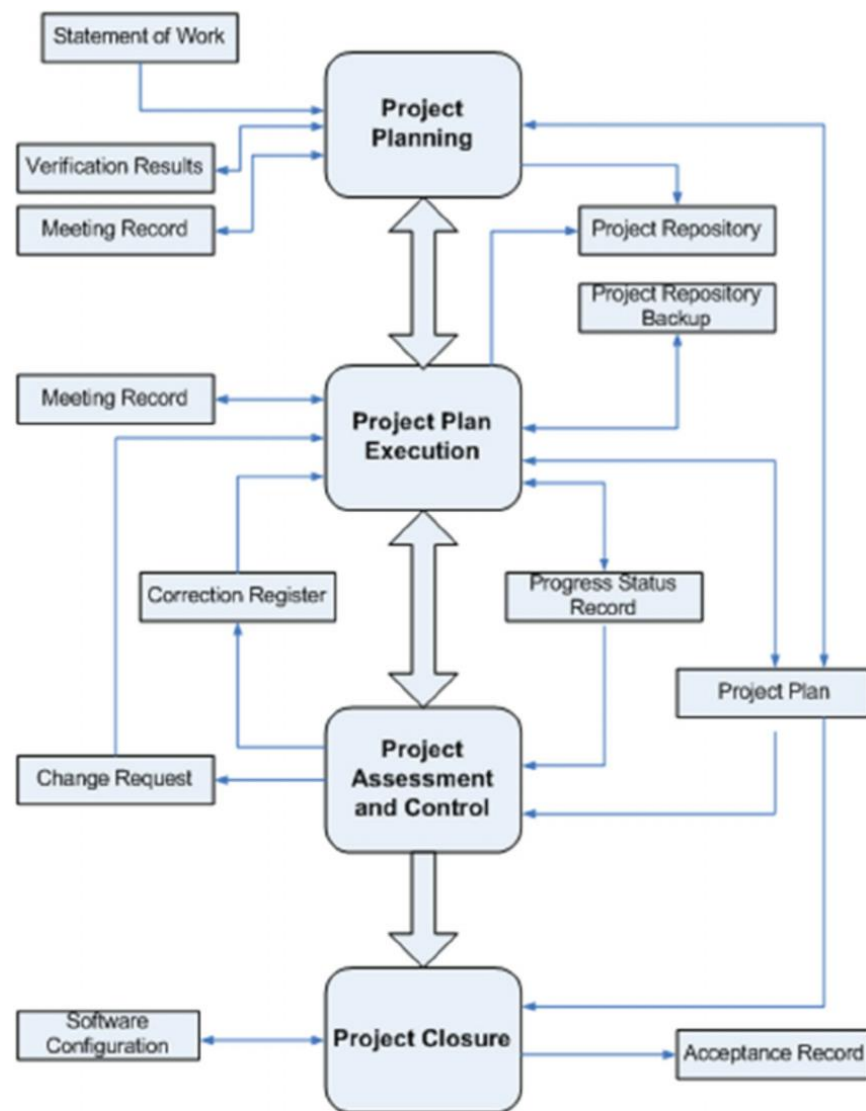
Table 12 – Standish Group Project Success Factors Over Time



Къде ще се
фокусираме
ние?

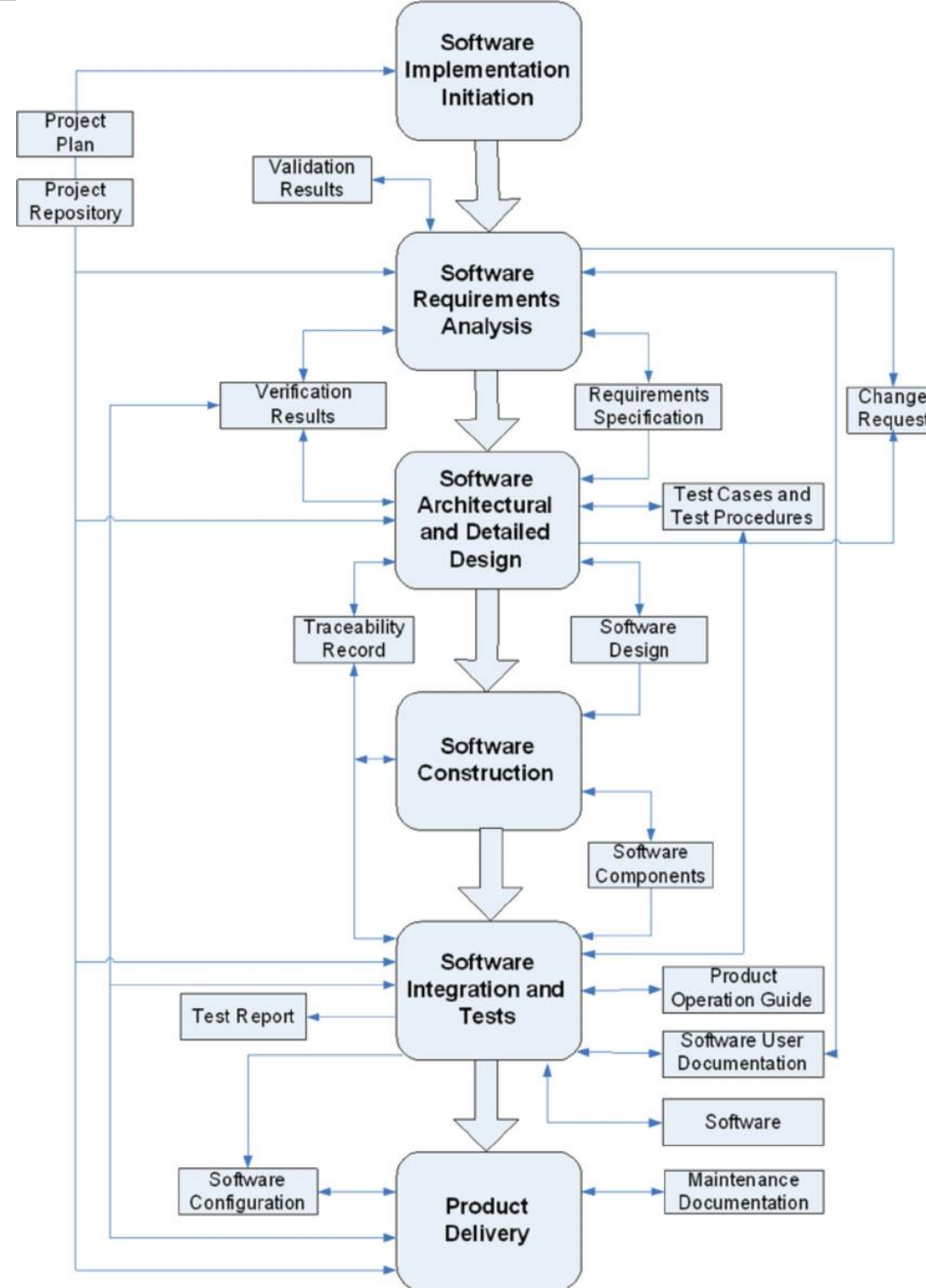
Жизнен цикъл на проектния мениджмънт

► В съществува си всеки един подпроцес разчита на резултатите както на предходните, така и на такива, които се случват след него (особено при итеративните модели на разработка). Това превръща целият жизнен цикъл в една сложна система от постоянно обновяващи се артефакти, което пък от своя страна е предизвикателство за всеки мениджър.



Жизнен цикъл на софтуерния процес

► Сложността на софтуерния процес расте с големината на проекта и е в съответствие със сложността на процеса по проектния мениджмънт.



Роли в проектния мениджмънт

Бизнес Анализатор (Анализатор на изисквания) – стреми се да извлече информация за системата, като изготвя техническа спецификация, която поддържа спрямо промените, произтичащи от клиента

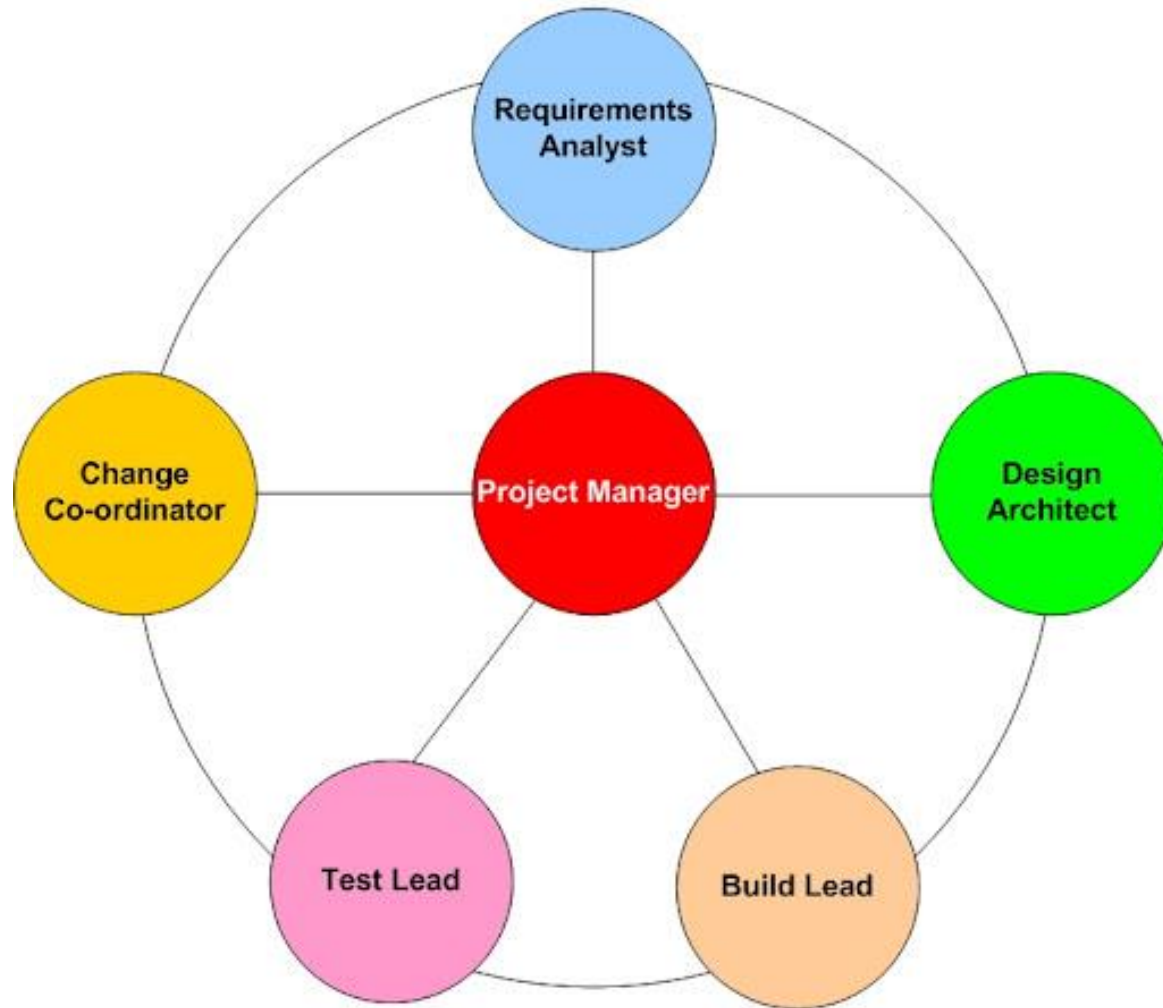
Софтуерен архитект – стреми се да предложи техническо решение/система, което да удовлетворява изискванията, извлечени от Бизнес анализатора

Лидер на екип по разработване – стреми се да имплементира така дефинираната система, като същевременно структурира компонентите на системата по такъв начин, че да са изпълнени добрите практики, както и нефункционални изисквания

Лидер на екип по качеството – стреми се да валидира и верифицира предоставената система според изискванията към системата. Отговаря за различните видове тествания

Координатор по промените – стреми се да отрази промените и да ги приоритизира според обратната връзка на заинтересованите лица

Project Team Roles



**Роли в
проектния
мениджмънт**

Роли в проектния мениджмънт

да се изберат
професионалисти с
технически профил,
отговарящ на
заданието

да се осигури
технически
инфраструктура за
разработка

да се подsigури
средата за
разработване

да се следи
обхватът, бюджетът,
степените на
развитие на
отделните задачи и
качеството им

да се управляват
рисковете

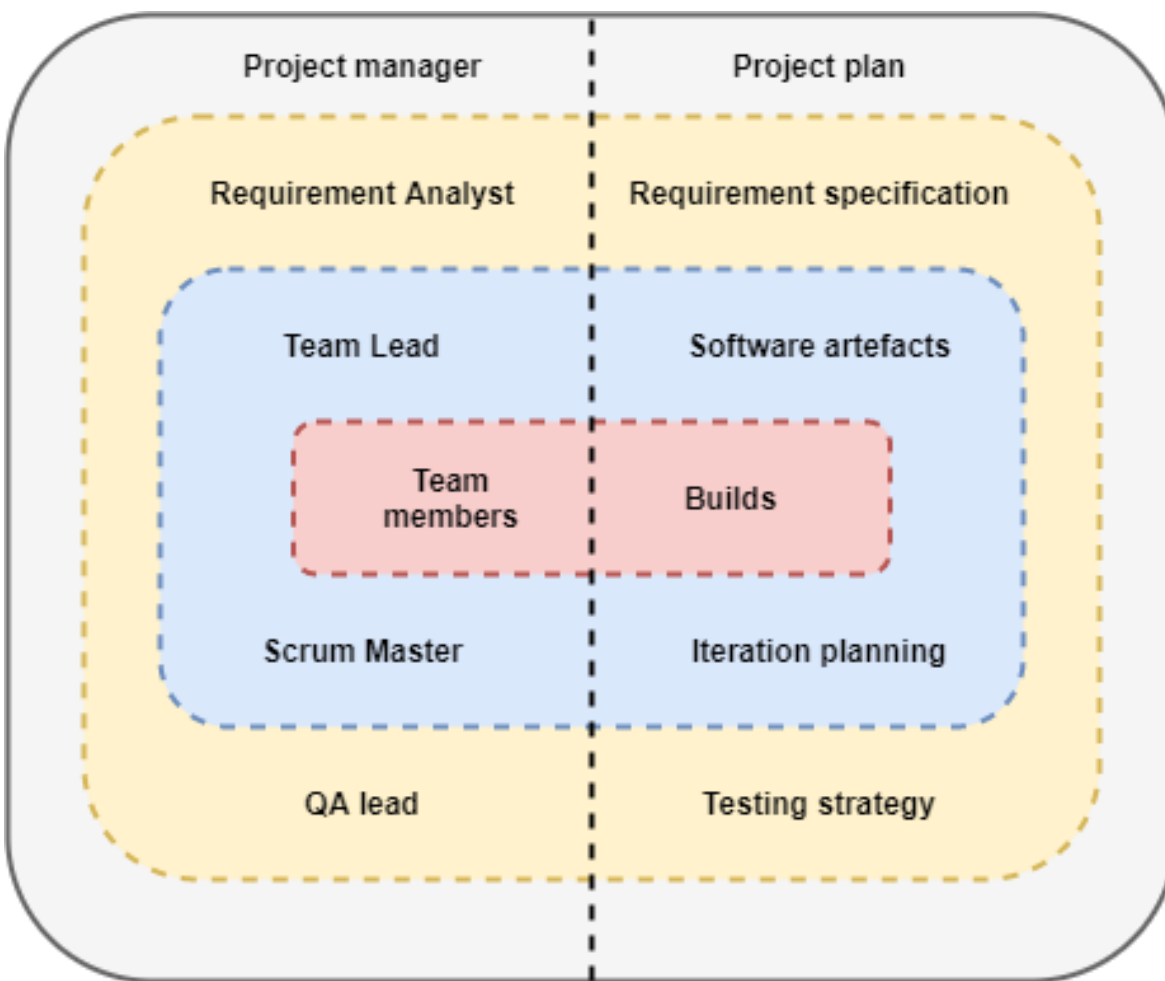
да се следи
предоставянето на
стойност

да се следи
управлението на
качеството на ниво
проект

да се следи
управлението на
промените

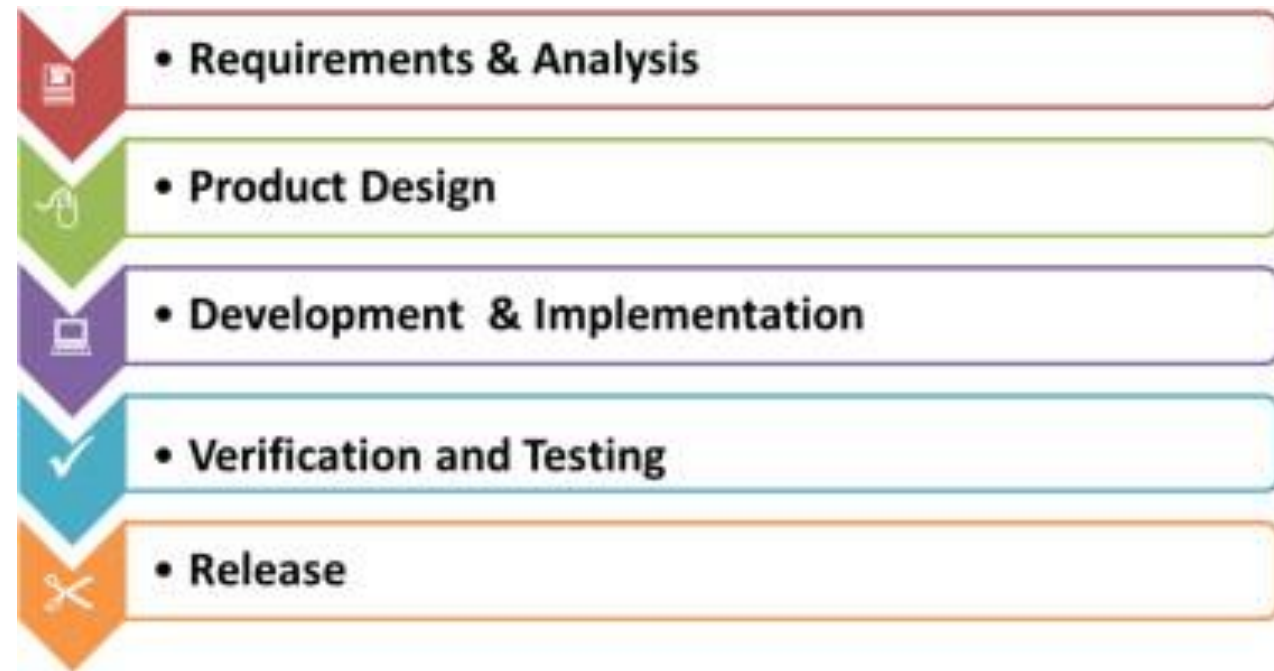
да се следи
комуникацията

да се следи
мотивацията в екипа

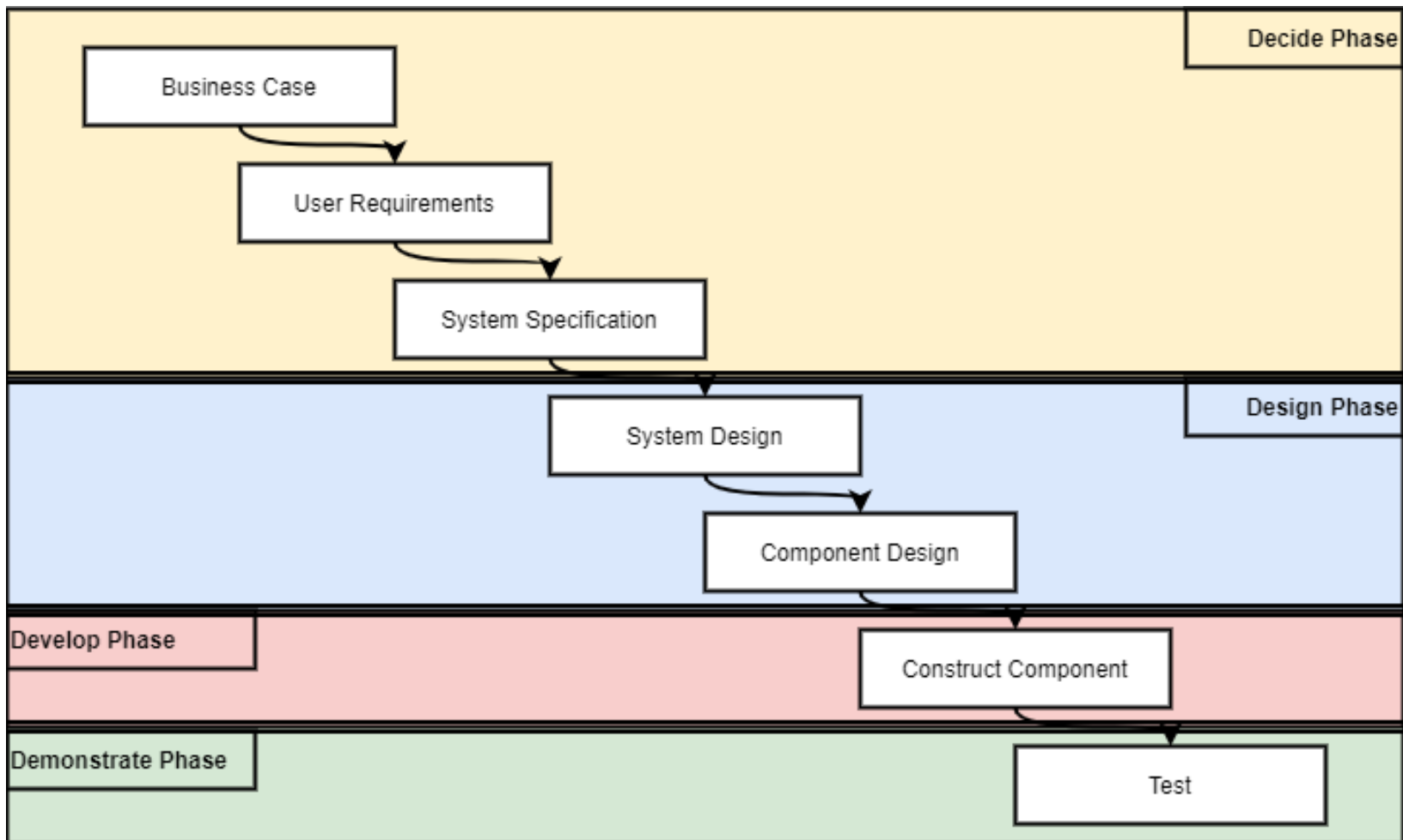


Роли в проектния мениджмънт

Модели на разработка на софтуер



Waterfall



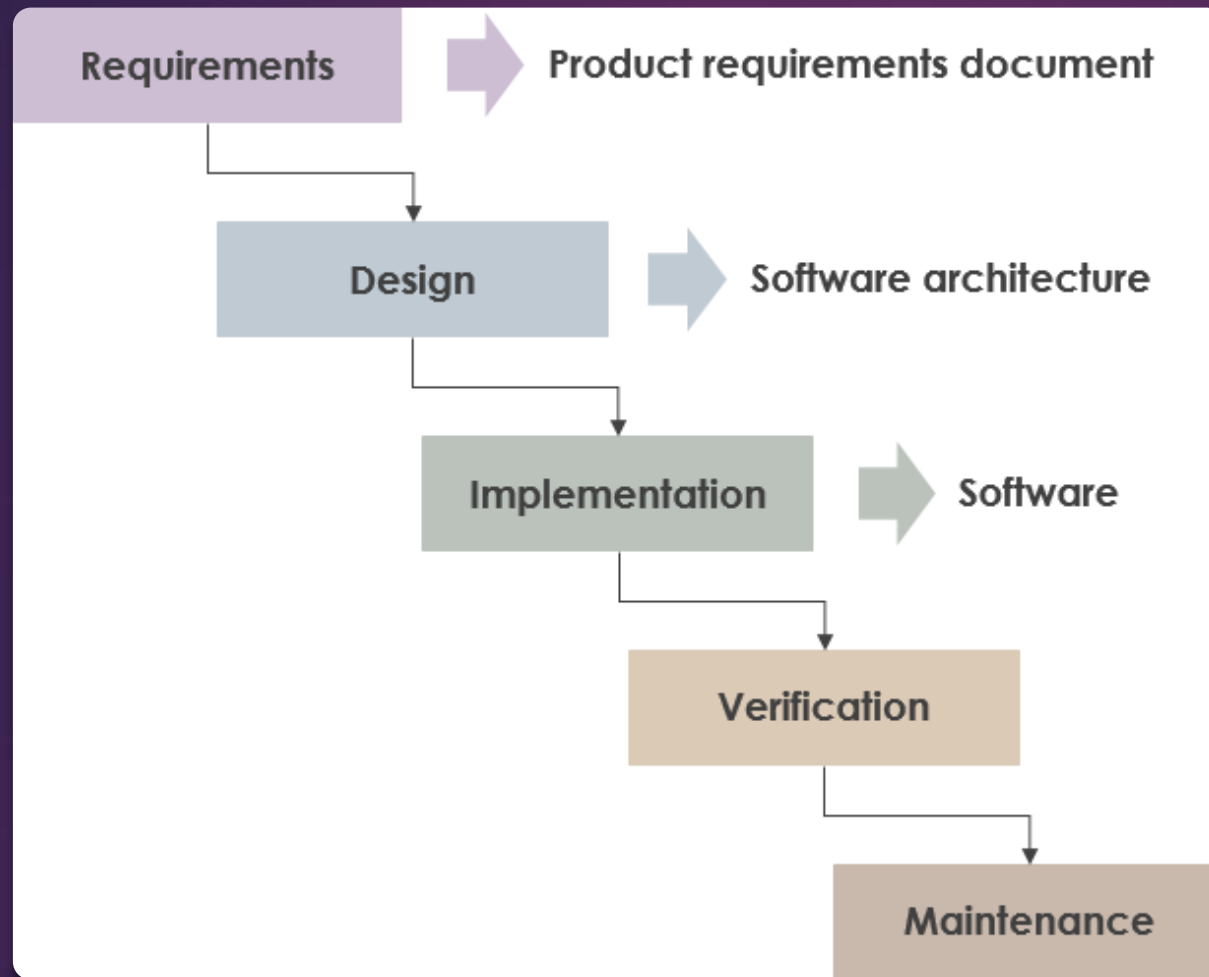
Decide phase

- ▶ Изисквания за екип – credibility, alignment with organizational goals, access to real costs;
- ▶ Изисквания към MOV (Measurable Organizational Value) – да е измерима стойност, да бъде от полза за организацията, да бъде съгласувана и възприета от членовете на екипа и да бъде валидируема;
- ▶ Идентифициране на алтернативните решения – разглеждане както на разработвани решения в средата, така и в самата компания;
- ▶ Изисквания за изпълнимост и оценка на риска - груба оценка цена/стойност, техническа изпълнимост, изпълнимост от гледна точка на организационни ресурси и др. Анализът на риска включва идентификация, оценка и стратегия;
- ▶ Изисквания за cost of ownership – началните капитали (хардуер, софтуер, телекомуникационни разходи), директни разходи по време на изпълнението и индиректни такива.
- ▶ Изисквания към общите привилегии – подобряване на качеството на възпроизвеждания продукт, подобряване на точността и ефикасността, подобряване на вземането на решения, подобряване на работата с клиенти.
- ▶ Анализ на алтернативите – възвращаемост;
- ▶ Изготвяне на предложение и защита на предложението (пред клиента или мениджмънт от по-високо ниво).

Design phase

System design - процесът по идентифициране на отделните модули и интерфейсите, чрез които те си комуникират. По този начин софтуерната архитектура придобива плътност.

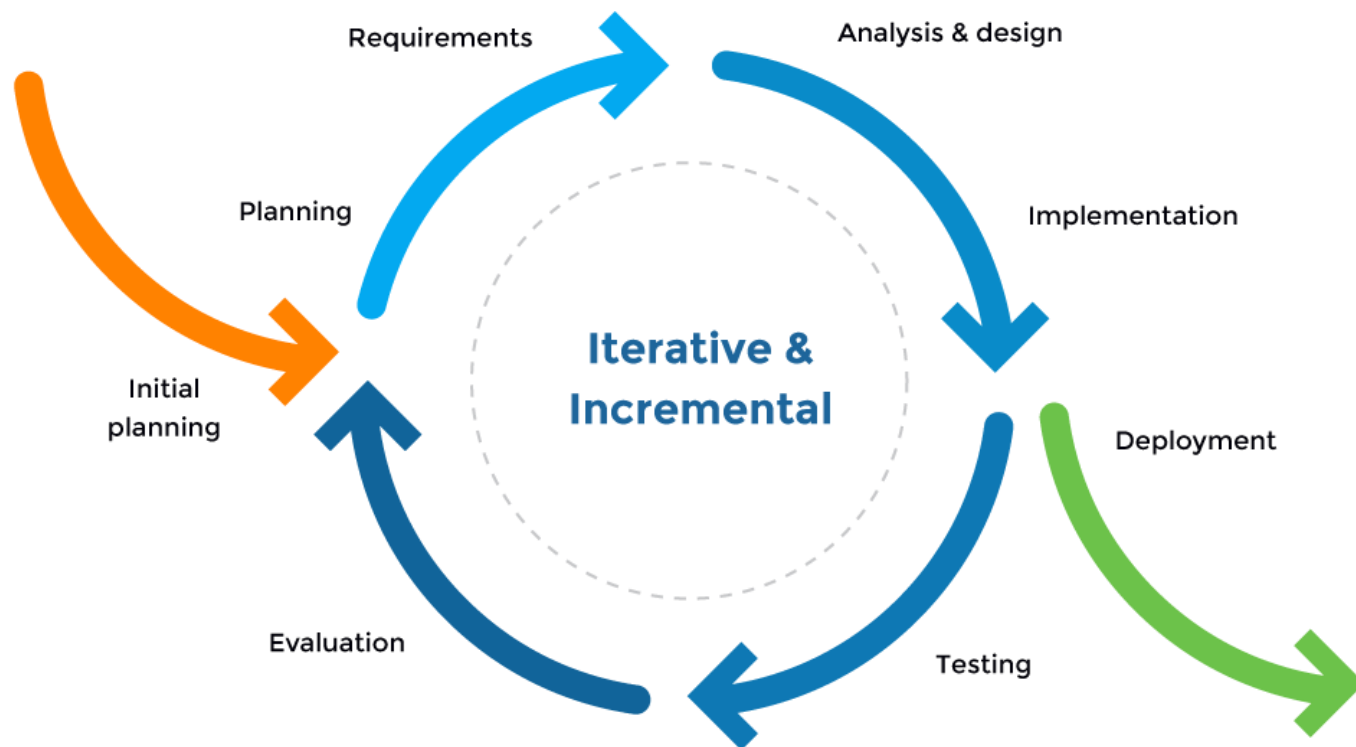
Component design - проектирането на ниво компонент спомага дефинирането на структури от данни (на ниво архитектура), алгоритми, методи на комуникация и протоколи, които да установят базови функционални и нефункционални изисквания. Целта тук е да се избере подходящ подход за функциониране на системата, като не се задълбава в детайли, а се описва на ниво абстракция.



**Develop phase –
Waterfall as it is**

Demonstration phase

- ▶ В случая внедряването на проекта в реална среда изисква и неговата поддръжка – не се планира версификация.
- ▶ Предимства от гледна точка на Проектния мениджмънт – лесен за управление, ефективен за малки проекти с ясни изисквания, малко и лесни за идентифициране рискове.
- ▶ Недостатъци от гледна точка на Проектния мениджмънт – необходима е опитност при определянето на бюджет, време и ресурси. Няма финален продукт до самия край на проекта, което поражда рискове. Неподходящ при големи проекти.



Итеративен модел

Итеративен МОДЕЛ

- ▶ Итеративният модел в Софтуерното инженерство се основава на итерации, като с всяка итерация се предоставя част от системата или се правят промени в системата на база обратна връзка или промени при заинтересованите лица. За да се внедри тази парадигма в Проектния мениджмънт, трябва да се приложи подход, достатъчно адаптивен по отношение на обхвата, ресурсите и качеството на софтуерния продукт
- ▶ За да се разбере какво представлява итеративния модел, трябва да се разберат две понятия:
 - ▶ Дефиниция: итеративен е подходът при който разработката на софтуерния продукт е разделена на времеви интервали (итерации).
 - ▶ Дефиниция: инкрементален е подходът, при който продуктът е предвиден да бъде предоставен порционално, като след всяка итерация се предоставя част от функционалност или цяла функционалност на клиента.

Планиране на итерациите: всяка итерация се планира като набор от цели, които трябва да се изпълнят в период от 2-6 седмици.

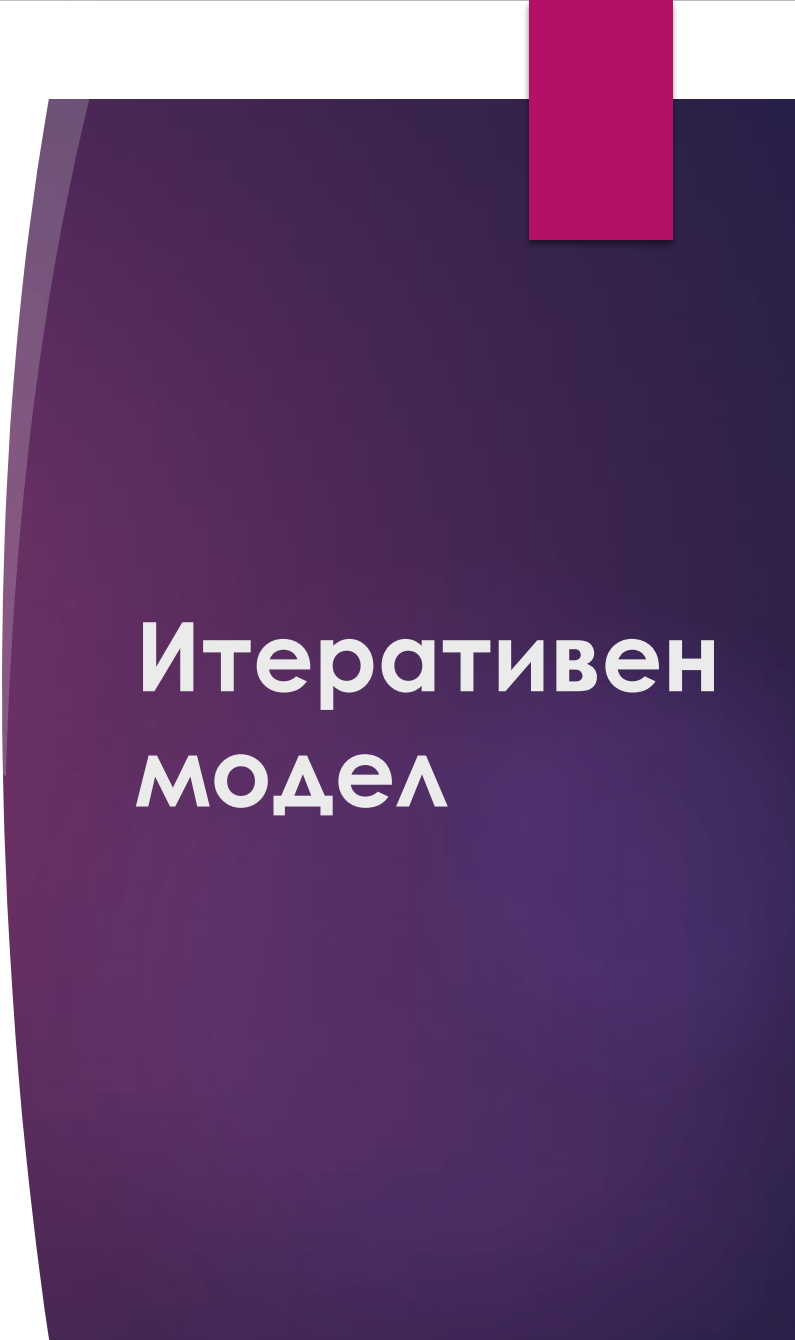
Проектният мениджър има отговорността да анализира и управлява риска при протичането на итерацията.

Важно е да се вземе предвид големината на екипа и да се съпостави на целевата му производителност.

Важно е и да има документалност на процеса, за да се постигне желаната проследимост на процеса – документът трябва да съдържа ясно дефинирани целите на итерацията, както и приоритетите на отделните задачи.

Планиране на отделните порции: всяка порция представлява основна част от продукта, която би била разработена, интегрирана и версифицирана.

Проекти с над година време за изпълнение, с високи технологични рискове или със стойност, зависеща от динамична обратна връзка, са подходящи за подобно планиране.



Итеративен модел

Итеративен модел

Предимства от гледна точка на ПМ:

- генерира приемлива за клиента стойност бързо;
- спомага ефективността на обратната връзка;
- лесно за тестване и управление на риска.

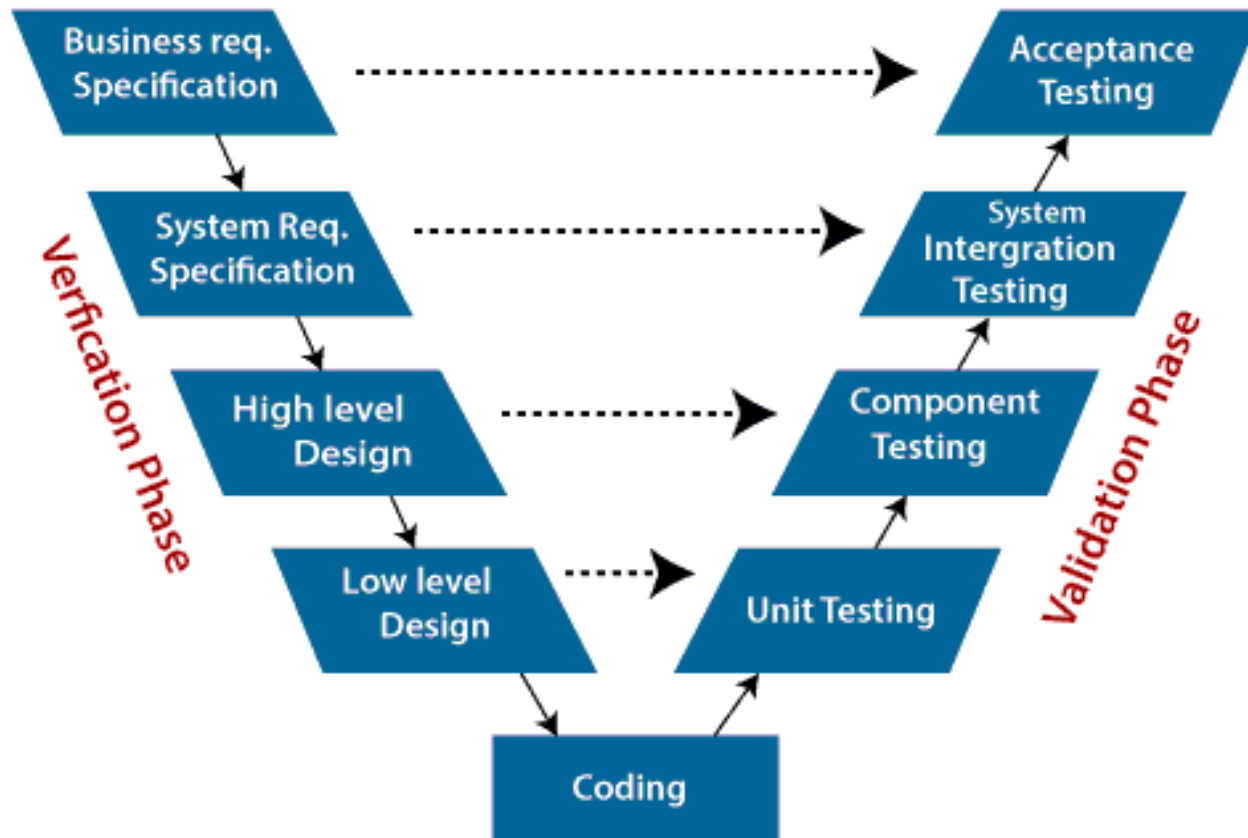
Недостатъци от гледна точка на ПМ:

- необходима е опитност за ефективното управление на итерациите;
- липса на ясна представа какъв е финалният продукт, към който екипът се стреми.

V- Model

Developer's life Cycle

Tester's Life Cycle



V-model

V-модел

- ▶ V-моделът следва последователния подход на изпълнение, като се придава тежест върху тестването в две части – верификация и валидация.
- ▶ За да се разбере какво представлява V-моделът, трябва да се разберат две понятия:
 - ▶ Дефиниция: верификация е процес на изкрystalизиране на целите, подходите и изискванията, който стартира още със започването на проекта и приключва окончателно към средата на неговото изпълнение. В общия случай целта е да се отговори на въпроса „Правим ли правилното нещо?“
 - ▶ Дефиниция: валидацията е процес на изчистване на резултатите, получени след прилагането на архитектура, технология, подход, като целта е да се повиши качеството на дефинираният след верификацията продукт. Отговаряме на въпроса „Правим ли нещата правилно?“

V-модел

Процесът започва с специфициране на софтуерните изисквания (както при водопадния модел), но се добавят допълнителни стъпки за дизайн на високо и ниско ниво, като успоредно се дефинира тестовият план, който ще валидира до колко произведеният продукт „среща“ първоначално въведените изисквания.

Дизайнът от високо ниво се фокусира върху архитектурната част – дефинират се отделните модули, както и начинът, по който те ще се съчетаят за да изпълнят даден потребителски случай. Резултатите са модел на системата и Integration testing plan.

Дизайнът от ниско ниво се фокусира върху компонентите и тяхната структура, като същевременно се изготвя стратегия за тестване на отделните софтуерни единици.

След разработката на проекта се пристъпва към тестването на системата на ниво Unit, след това Component, Integration и накрая Acceptance testing.

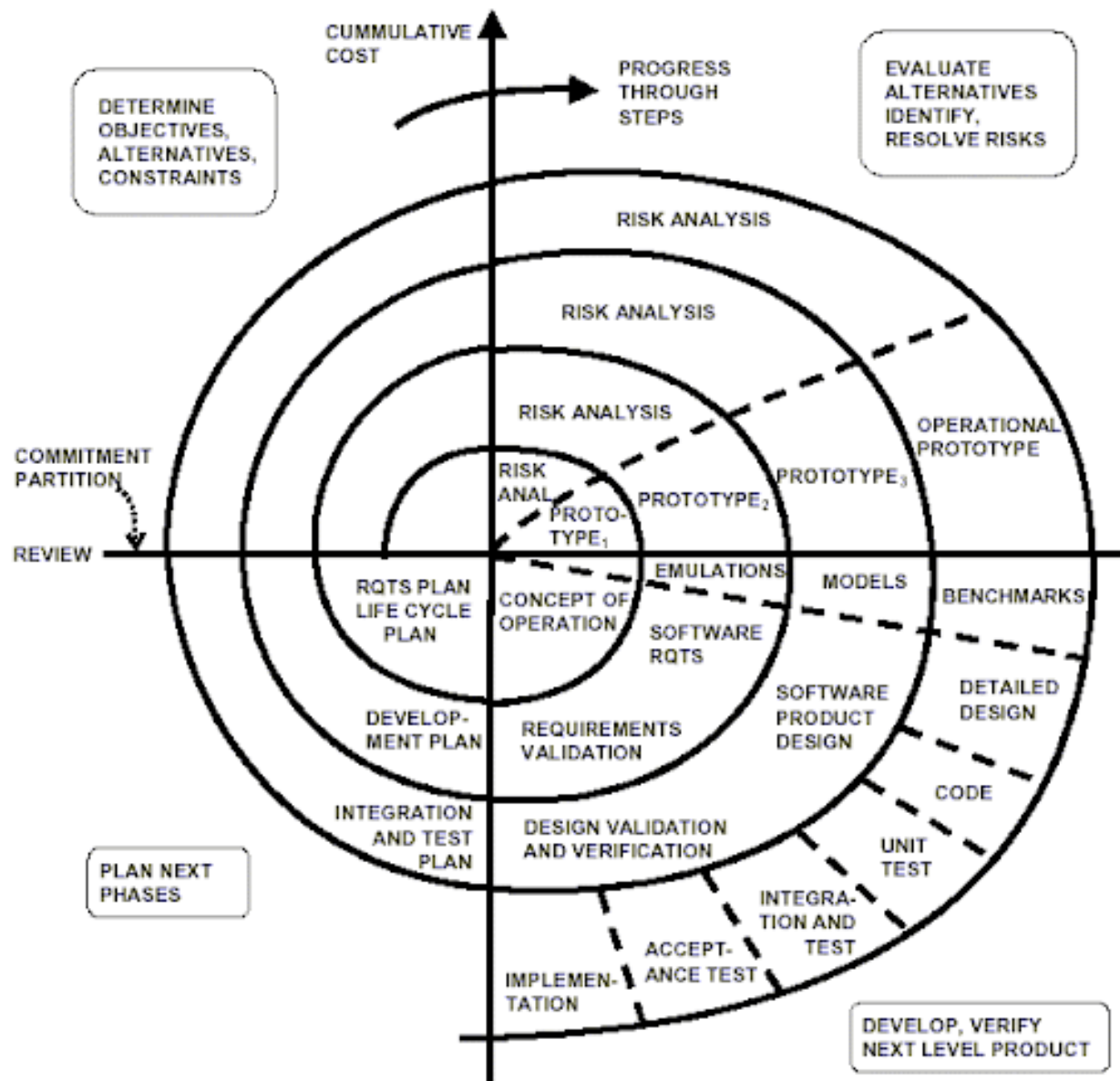
V-модел

Предимства от гледна точка на ПМ:

- лесен за прилагане;
- с ясно дефинирани цели след всеки етап;
- по-голяма ефективност в сравнение с водопадния модел.

Недостатъци от гледна точка на ПМ:

- обхватът на проекта е статичен – т.е. трудно евентуални промени в изискванията могат да бъдат интегрирани успешно.
- Какво се случва ако тестовите планове не са покрили минимума потребителски случаи?



Спираловиден модел

Спираловиден модел

Спираловидният модел следва итеративния подход на изпълнение, като акцент се придава върху рисковете, които могат да възникнат както по време на изпълнението, така и по време на тестването и внедряването на дадена система.

Залага се на прототипния модел и версифициране на системата, т.е. със всяко „завъртане“ на спиралата продуктът преминава в нова версия, която отразява промените в средата. Затова и се приема, че спираловидният модел е подходящ при разработването на големи софтуерни продукти, чиято дейност продължава с години и смяната ѝ с друга система е обвързана с голям риск от загуба на информация, клиенти и ресурси

Round 0: Feasibility study

Предпроектното проучване може да продължи до 3 месеца и си поставя за цел изкрystalизиране на функционалности и ограничения, като се иска балансиране между качество и цена. Обсъждат се алтернативи от технологичен характер, като се взима предвид и потенциалната интеграция с модела на управление на процеса.

Оценката на риска е резултат от анализи на база проведени срещи, интервюта и анкети. Правят се и сравнения с други системи, като акцентът пада върху онова, с което проектът ще допринесе на пазара.

Всички доклади и анализи са добра основа за предстоящия първи рунд от спиралата, където се дава акцент върху конкретни цели в рамките на даден период (например една година), както и оценка на разходите спрямо първоначалната предоставена стойност.

Round 1: Concept of operation

Изясняването на концепцията може да продължи до година и включва много по-подробни и детайлни проучвания, които изискват и по-голяма експертиза. Всичко това резултира в подробен план от сравнителни анализи с технологичен характер – сравнения на софтуерни архитектурни решения и методи на разработка. Поставя се за цел в следващата фаза да се изготви техническата спецификация, планът за управление на риска.

Round 2: Top-Level Requirements Specification

- ▶ Детайлизират се функционалностите, както и компонентите на вече дефинираната софтуерна система. Водещ при извличането на изисквания винаги е рискът.

Next phases

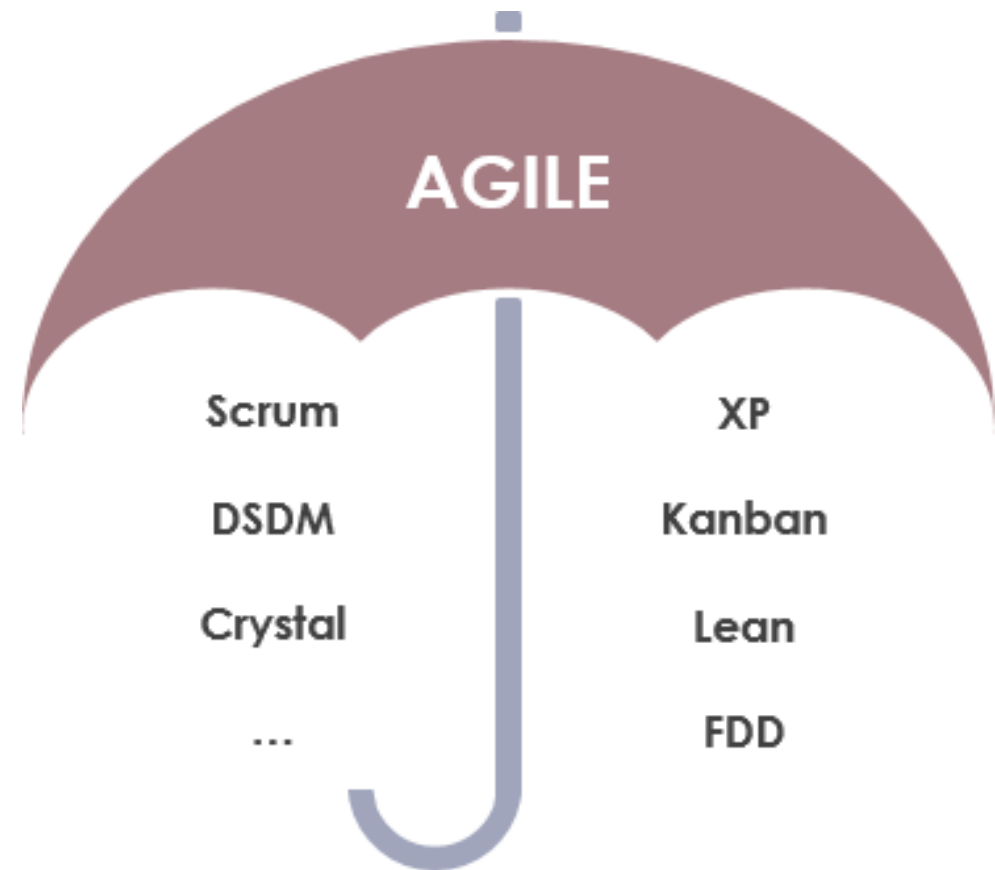
- ▶ Детайлизират се функционалностите, както и компонентите на вече дефинираната софтуерна система. Водещ при извличането на изисквания винаги е рискът.

Table 4. A prioritized top-ten list of software risk items

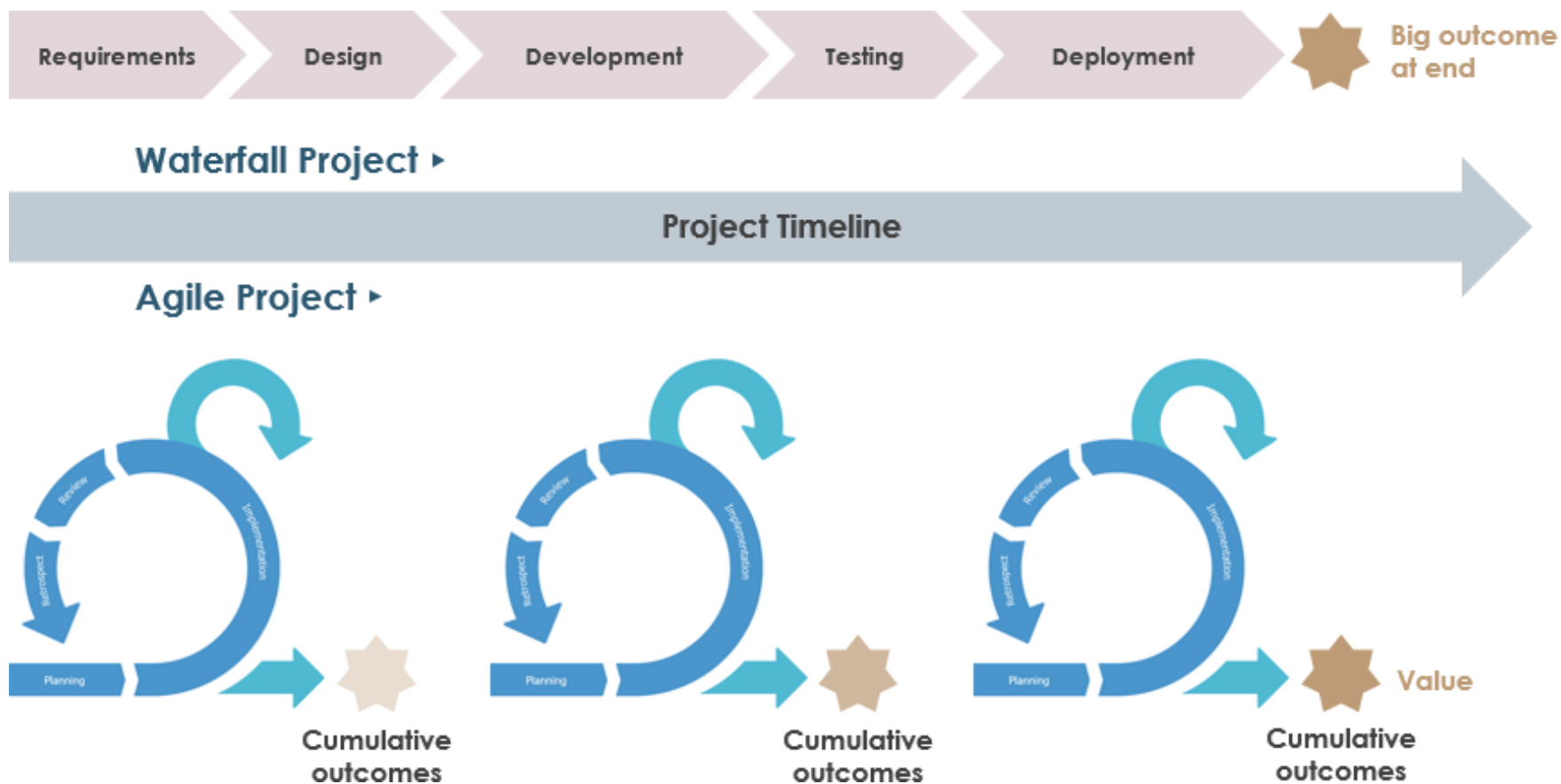
Risk Item	Risk management techniques
1. Personnel shortfalls	Staffing with top talent, job matching; teambuilding; morale building; cross-training; pre-scheduling key people
2. Unrealistic schedules and budgets	Detailed, multisource cost and schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing
3. Developing the wrong software functions	Organization analysis; mission analysis; ops-concept formulation; user surveys; prototyping; early users' manuals
4. Developing the wrong user interface	Task analysis; prototyping; scenarios; user characterization (functionality, style, workload)
5. Gold plating	Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
6. Continuing stream of requirement changes	High change threshold; information hiding; incremental development (defer changes to later increments)
7. Shortfalls in externally furnished components	Benchmarking; inspections; reference checking; compatibility analysis
8. Shortfalls in externally performed tasks	Reference checking; pre-award audits; award-fee contracts; competitive design or prototyping; teambuilding
9. Real-time performance shortfalls	Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
10. Straining computer-science capabilities	Technical analysis; cost—benefit analysis; prototyping; reference checking

**Most
anticipated
risks**

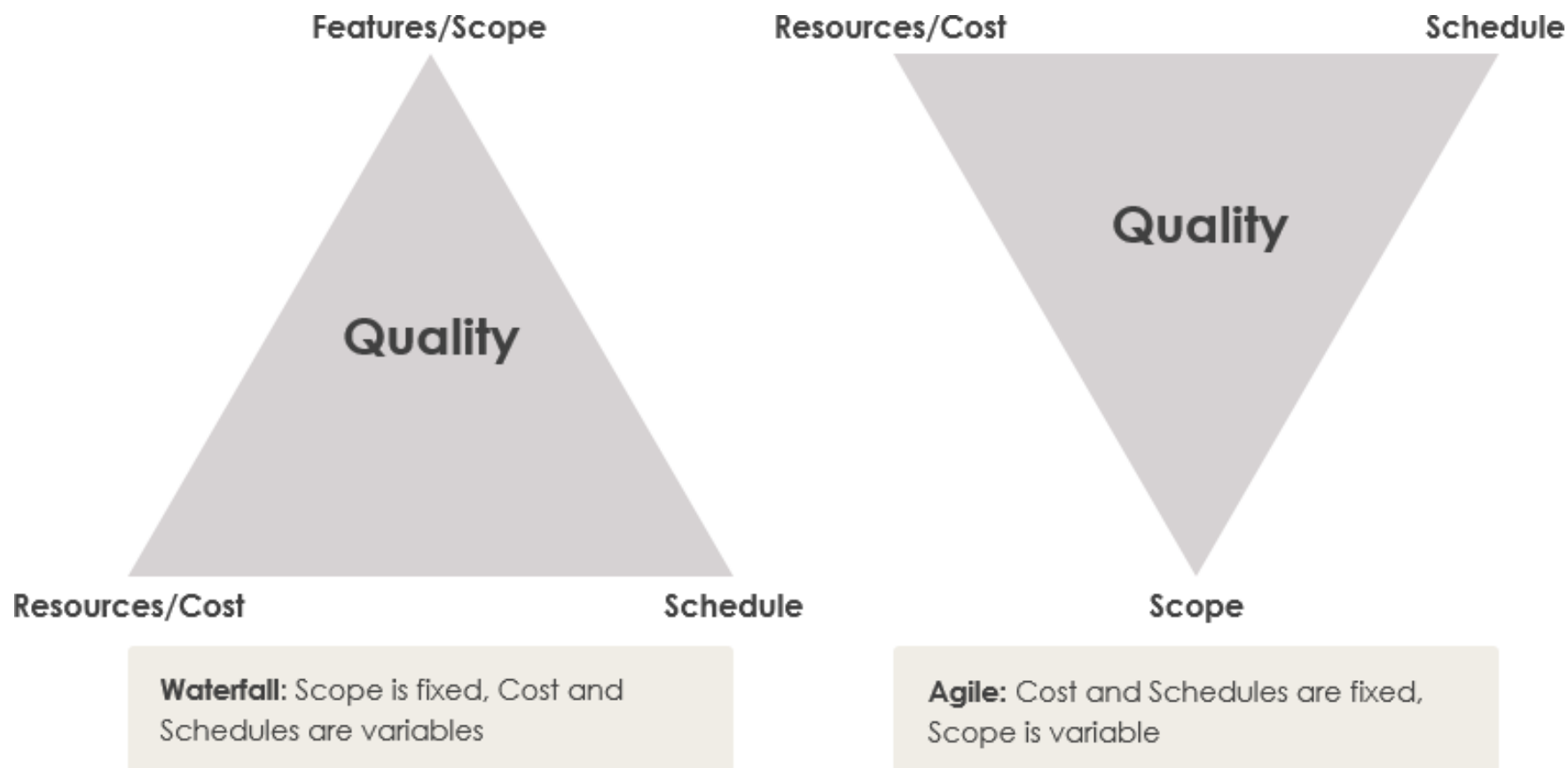
Ролята на методологиите при моделите на разработка



От гледна точка на софтуерния процес



От гледна точка на проектния процес



От гледна точка на резултатите

Method	Successful	Challenged	Failed
Agile	42%	50%	8%
Waterfall	26%	53%	21%

От гледна точка на различните методологии

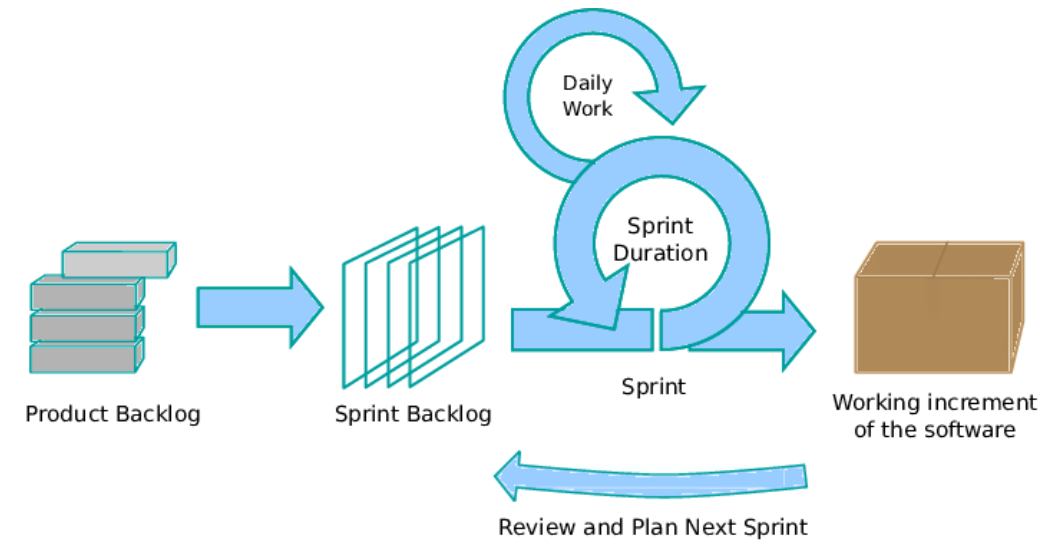
Scrum - акцентът е върху планирането на итерациите и стремежът на екипа да изпълнява своите цели в подходящи времеви рамки. Всяка итерация започва с планиране на отделните задачи, като важно е всяка задача да присъства със своя приоритет и принос към общите цели на проекта.

Lean - акцентът е върху стойността и начина, по който тя се формира. Въз основа на това се изготвя процес по предоставяне на стойност, който цели да се намалят максимално излишните дейности. Този процес по изчистване на процеса се повтаря с всяка итерация.

Kanban - принципите се базират на това как се възприема дейността по разработването на софтуер с оглед колаборацията между отделните членове на екипа и комуникацията на междуекипно и вътрешноекипно ниво. Също с цел да се оптимизира работата се слага ограничение на започнатите задачи и допълнителното им приоритизиране. На база свършената работа се прави анализ на процеса по разработка, като се цели преход към Continuous Integration процес

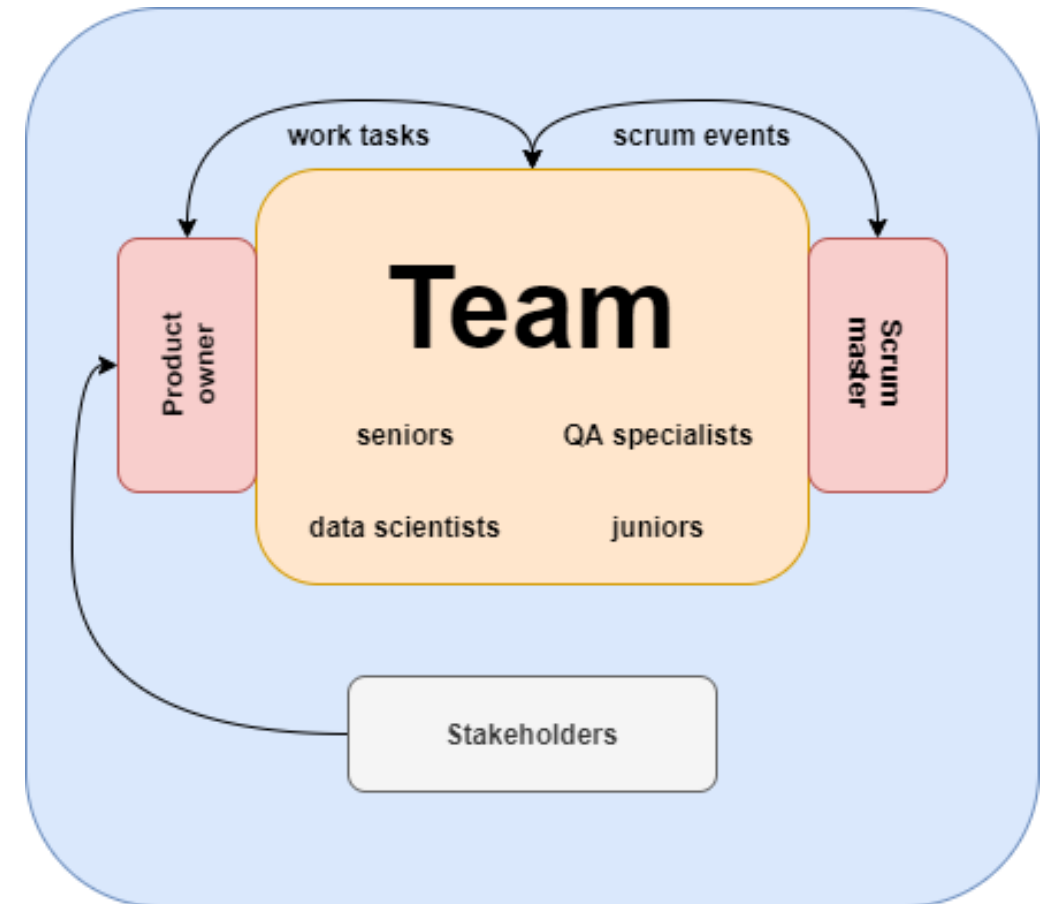
Scrum

- При Scrum акцентът е върху планирането на итерациите и стремежът на екипа да изпълнява своите цели в подходящи времеви рамки. Всяка итерация започва с планиране на отделните задачи, като важно е всяка задача да присъства със своя приоритет и принос към общите цели на проекта.



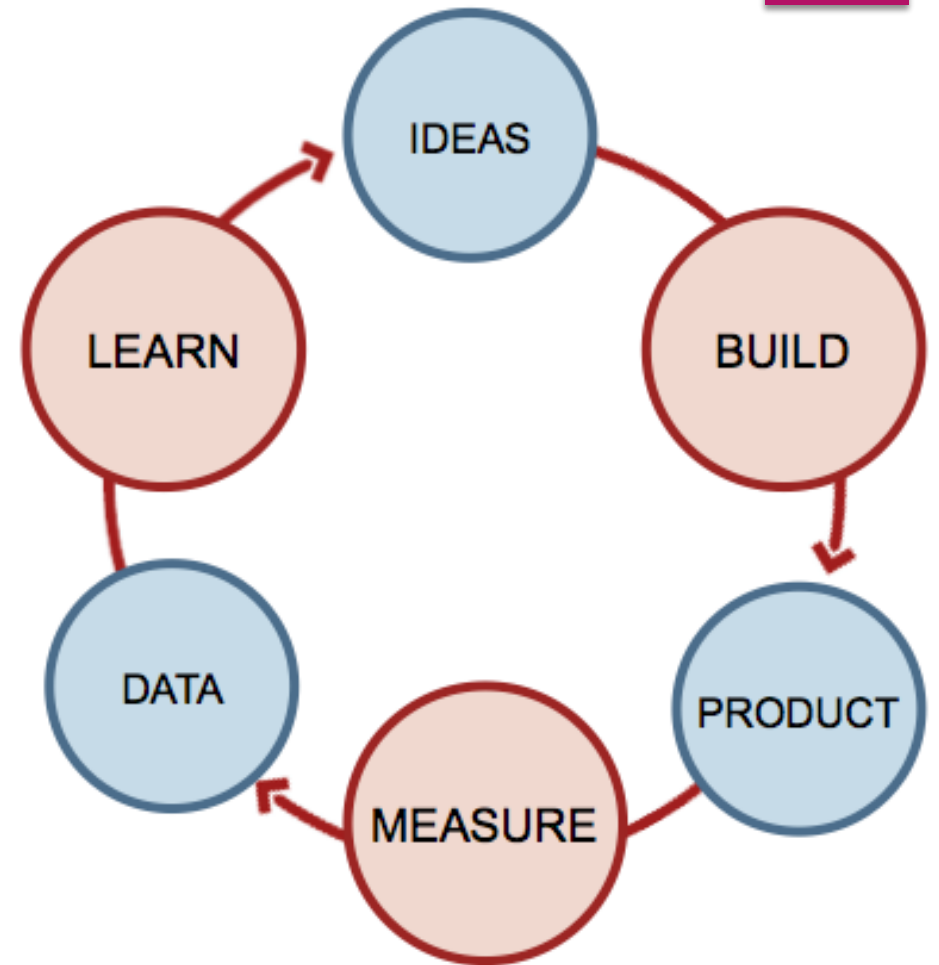
Scrum

- Scrum master е отговорен за протичането на целият Scrum процес и с дейността си допринася за предоставянето на стойност. За тази цел не е необходимо той да контактува с заинтересованите лица – той получава необходимата информация от Product owner, който е отговорен както за приоритизирането на промените, така и за тяхното детайлизиране.



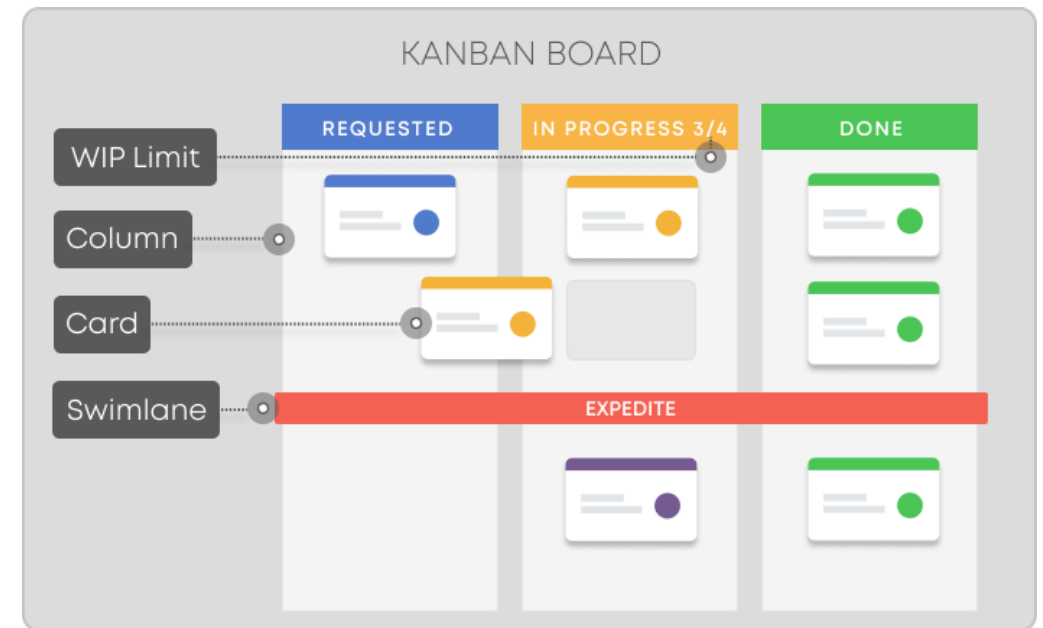
Lean

- ▶ Акцентът е върху стойността и начина, по който тя се формира. Въз основа на това се изготвя процес по предоставяне на стойност, който цели да се намалят максимално излишните дейности. Този процес по изчистване на процеса се повтаря с всяка итерация.



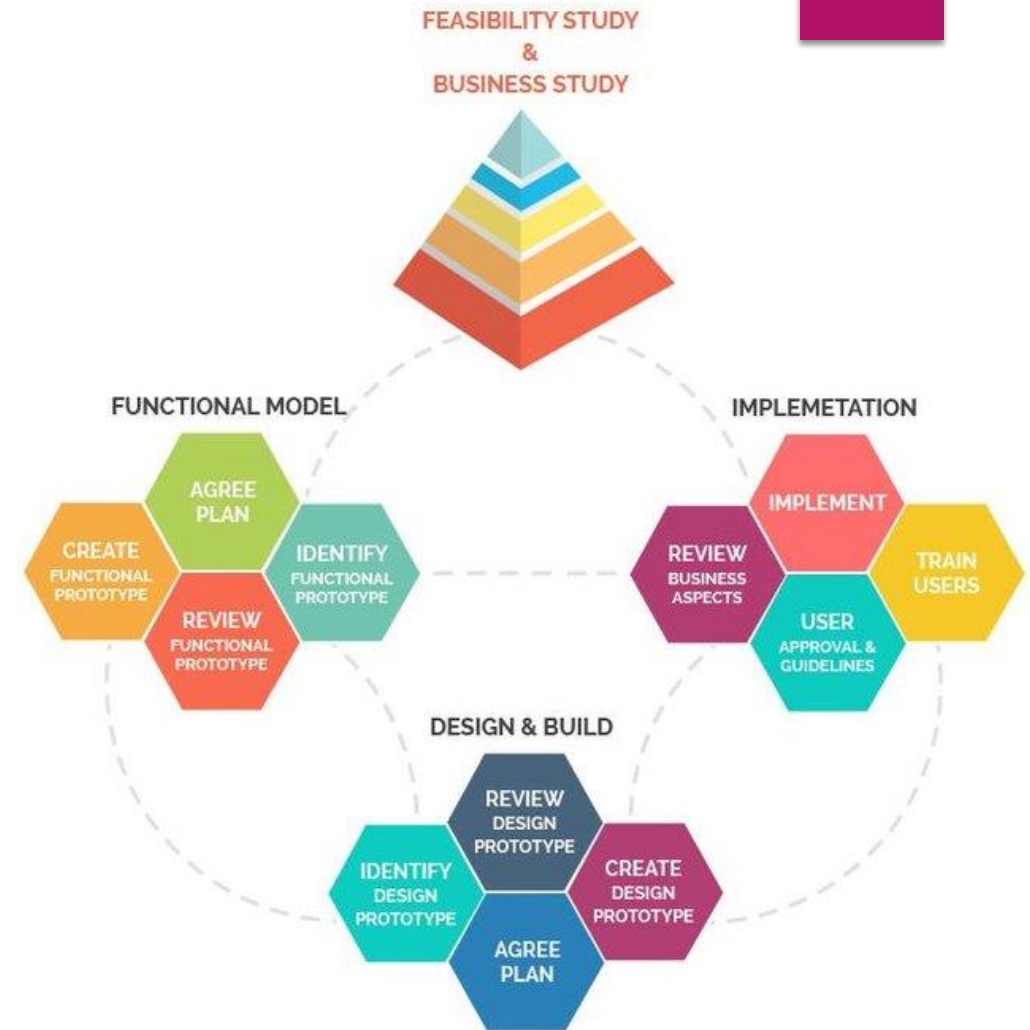
KANBAN

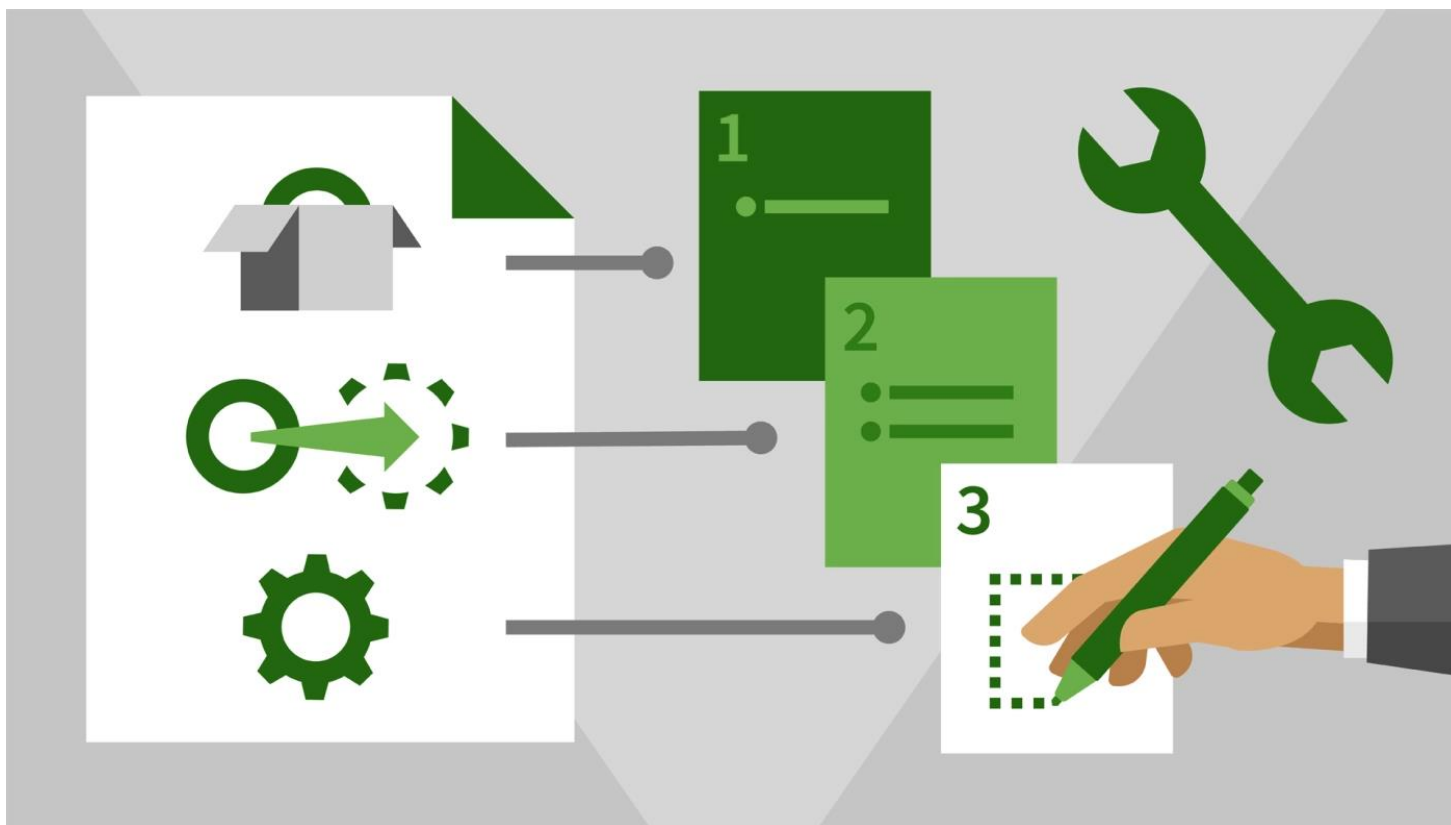
- ▶ Принципите се базират на това как се възприема дейността по разработването на софтуер с оглед колаборацията между отделните членове на екипа и комуникацията на междуетипно и вътрешноекипно ниво. Също с цел да се оптимизира работата се слага ограничение на започнатите задачи и допълнителното им приоритизиране. На база свършената работа се прави анализ на процеса по разработка, като се цели преход към Continuous Integration процес



DSDM

- При Dynamic System Development Method акцентът е върху минимизиране на времето и разработване на онези функционалности, за които има най-много информация и при които вероятността те да бъдат променени е минимална. Залага се на максимално включване на крайните потребители в комуникацията и решението за промените се взима от екипа. Тестването и внедряването протичат през целия процес.





Софтуерна документация

ОСНОВНИ ДОКУМЕНТИ



Communication Management Plan



High – level Business Requirement Specification



Project Status Report



Risk Tracking Log



Scope Change Log



Project Charter



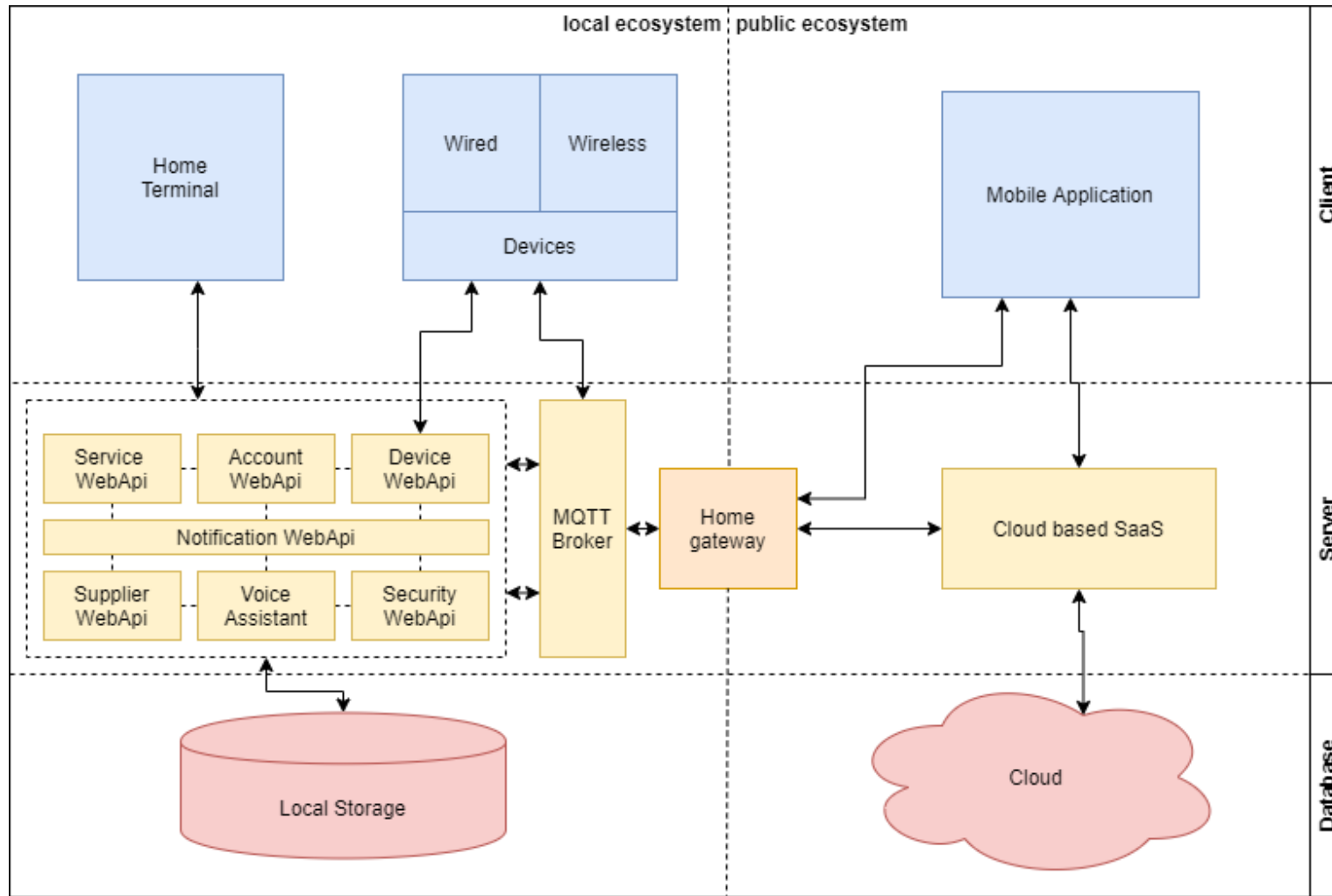
Issues log



Change Requests



Упражнения



Упражнение 1

Упражнение 1

Дадено е следното описание на система:

Home Manager Ultra представлява мулти-компонентна система за управление на ресурсите в умен дом, състояща се от няколко архитектурни слоя, сред които:

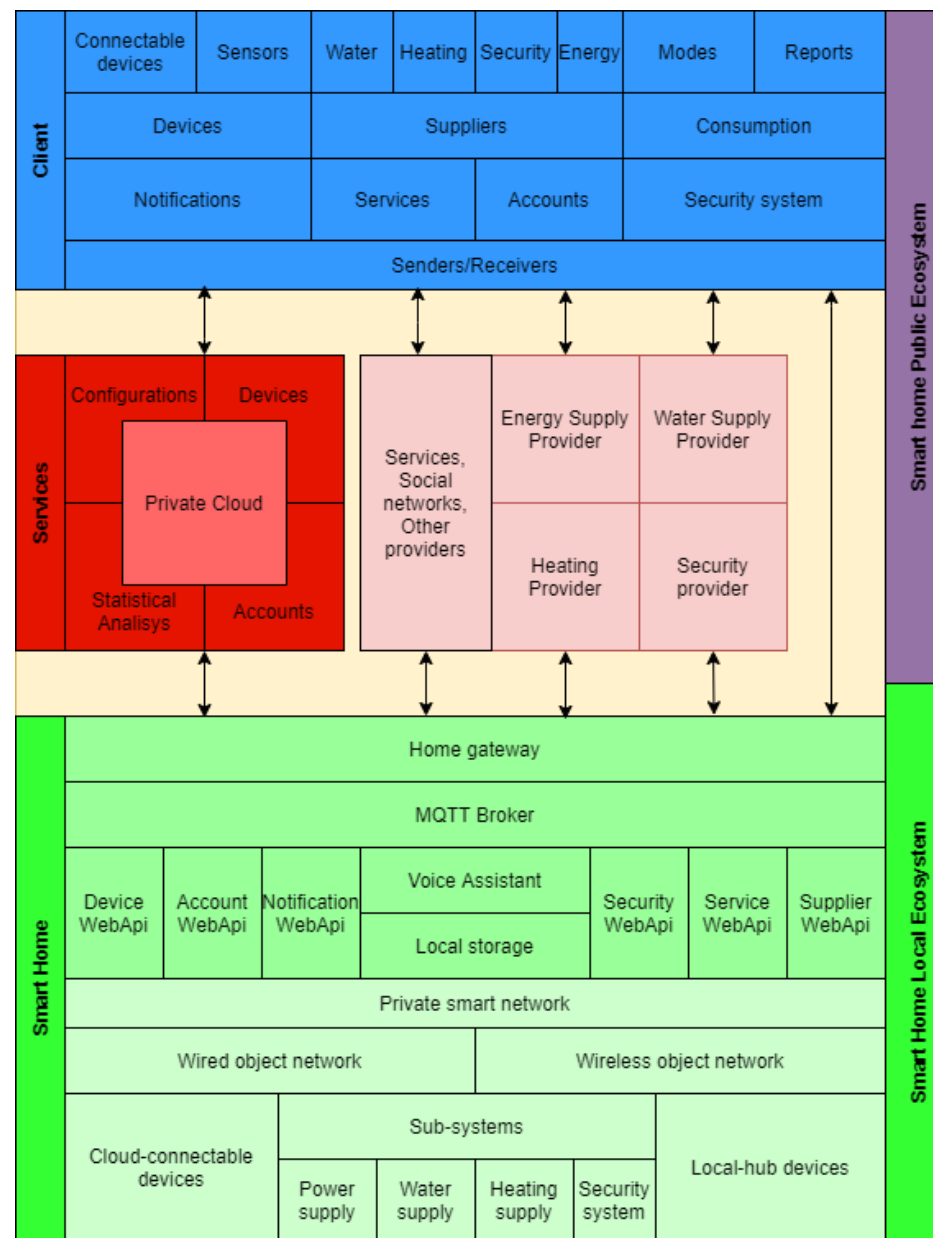
- клиентски слой, в който са поместени умните устройства, които изпращат информация на системата, централен терминал, който се намира на територията на дома и предоставя удобен интерфейс за комуникация и мобилно устройство, което позволява отдалечен мониторинг на устройствата;
- сървърен слой, който обхваща софтуерната част на локално ниво от една страна и на ниво облачни структури, които позволяват свързаност с други компоненти;
- слой на данните, който включва локалното хранилище и облачното такова;

Задача: Да се проведе дискусия на тема – какви модели на разработка и методологии за предоставяне на стойност биха били подходящи при имплементацията на подобна система?

Упражнение 1

Направено е компонентно представяне на системата според слоевете, описани в предишната задача.

- **Задача:** Да се обмислят потенциалните рискове, които биха възникнали и да се направят предложения за тяхното минимизиране.
- **Задача:** Да се обсъдят потенциалните стратегии за тестване на системата.





Въпроси