Асистент - преподавател: д-р Христиан Даскалов

Лекции № 8, 9: Основни развойни и управленски концепции в отворена среда - Agile Development; Scrum & Extreme Programming

Курсов ръководител: проф. д-р инж. Огнян Андреев

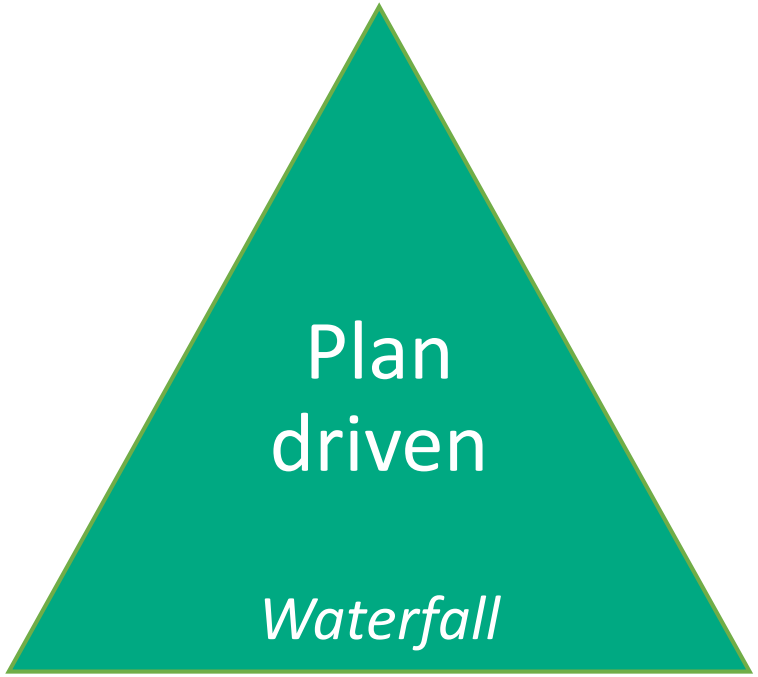
	Ключови процеси	Критични фактори за успех	Съставни компоненти, свързани с проектите с отворен код
Група: Инициране и планиране	**Идентифициране на заинтересованите страни**	**Контекстуално управление**	Преглед и анализ на съществуващите проекти
			Изграждане на устойчива коалиция
			Разбиране на маркетинговите аспекти
			Оценка на финансовите нужди и източници
		Обследване на заинтересованите	Обследване на заинтересованите
	Планиране на управлението на заинтересованите страни	**Стратегия за ангажиране на заинтересованите**	Проблемно-ориентирано проектно начало
			Изчистена и резонираща проектна мисия
			Възлагане на отговорности
			Стабилност чрез привличане към общността
		Отчитане на цялостния продуктов жизнен цикъл	Поглед върху всички области на системната разработка
			Планиране и дизайн преди иницииране
			Отчитане на социо-техническата еволюция на проекта
			Потребителско опосредстване и подкрепа, клиентска поддръжка
		Технологична съгласуваност	Отчитане на технологичните ограничения
			Равновесие между техническото ниво на проекта и участниците
			Развитие на обкръжаващата екосистема
			Развитие чрез модулна организация на работата
Група: Изпълнение и контрол	**Управление на участието на заинтересованите**	**Изграждане на общност**	Подхранване на общностната идентичност
			Менторство и подкрепа в общността
			Провеждане на общностни събития
			Възможности за развитие
		Стратегически управленски подход	Посвещаване на проектната стратегия
			Модел за управление и вземане на решения
			Прилагане на доказани проектни практики
			Проактивна мрежова стратегия
		Техническо управление, насърчаващо участието	Динамично развитие на нови версии
			Управление на комплексността
			Формализирано управление на изискванията и приносите
			Систематичност на заявките за отстраняване на проблеми
	Контрол по управлението на заинтересованите	**Превенция на конфликти от тех. характер**	Прилагане на лицензионни споразумения
			Качествено-ориентиран развоен процес
			Отворени срещи по техническото управление
		Оценка и оптимизация на представянето	Оценка и оптимизация на представянето

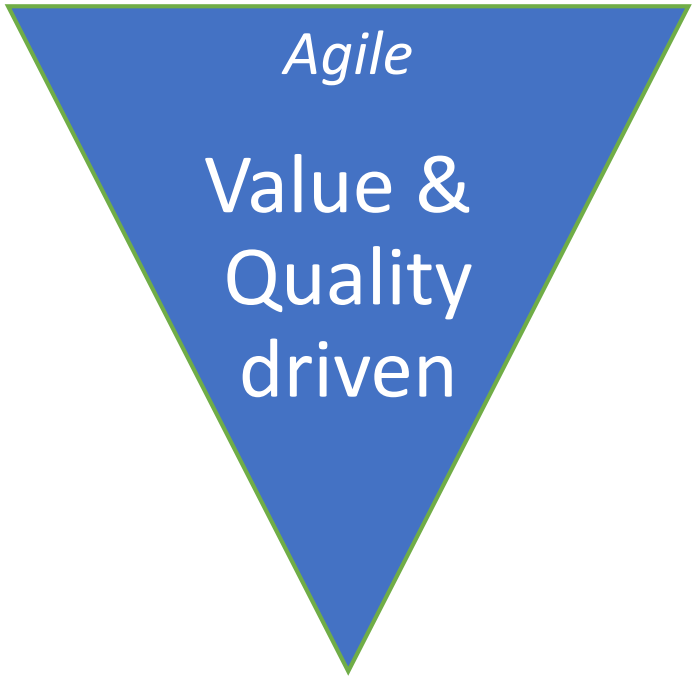
Agile Development

ТЕМА II, МОДУЛ II «СПЕЦИФИКА ПРИ УПРАВЛЕНИЕТО НА СОФТУЕРНИ ПРОЕКТИ»

Agile обръща разработката „наопаки“

Ограничения	Изисквания / Requirements	Цена / Cost	График / Schedule





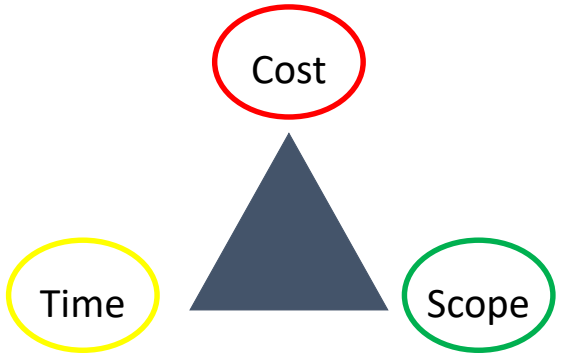
Estimates	Цена / Cost	График / Schedule	Features

The PLAN creates cost/schedule estimates

The VISION creates feature estimates

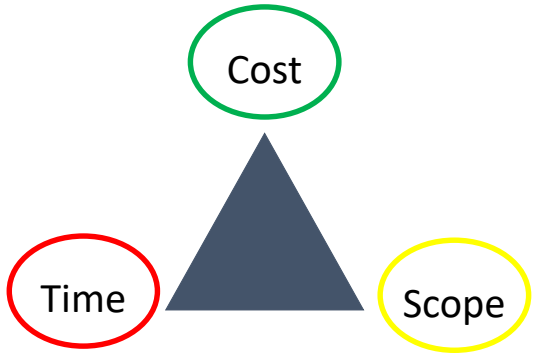
Допълнителни бележки

Waterfall



- **Scope fixed** with potentially Time/Cost varying
- **Planning**
  - Demand agreed by business
  - Requirements defined by business analysts
  - Requirements approved by stakeholders
  - Project budget agreed and approved
  - Business hand-over to project
- **Progress**
  - Phase gates defined by project/PM4U
  - Milestones, work products & cost estimated
  - Defects found and corrected
  - Project hand-over to operations
- **End-user availability feedback**
  - Deployment hand-over
  - Operations failures and incidents
  - Hand-over to business – value realization
- SLA's based on delivered items
- Value measured on delivered items based on specification and initial estimates

Agile

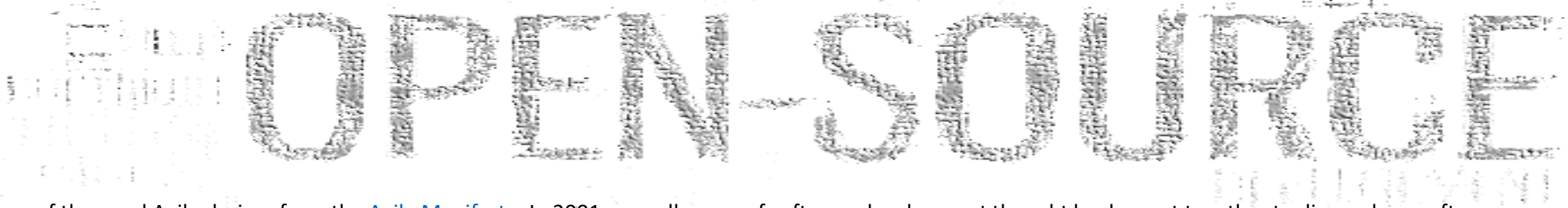


- **Cost fixed** with potentially Scope/Time varying
- **Planning**
  - Demand identified, outlined, prioritized on Product Backlog
  - Demand broken down and sized into Epics/User stories
  - Business and Product Owners involved in estimations and planning for deploy on Release Backlog
  - Product Owners and teams estimating implementation effort needed on Sprint Backlog
  - Team pulls functionality to be in sprint using previous velocity as guide
- **Progress**
  - Burn-down/Burn-up charts showing how much effort is still needed to be done and re-evaluated each day
  - Team velocity tracked and established
  - Daily stand-ups inviting all stakeholders
- **End-user availability feedback**
  - Continuous deployment and metrics to show
  - Operations failures and incidents
- Instant transparency keeping to business involved SLA's based on delivered capacity
- Value measured on continuously re-evaluated functionality made available to customer

Какво е Agile?

Има много начини по които Agile може да се обясни – ето няколко:

1. *'Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams'* – **Wikipedia**
2. *'The ability to create and respond to change in order to succeed in an uncertain and turbulent environment'* – **Agile Alliance**
3. *'Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.'* – **Agile In a Nutshell**
4. *'Agile is a term used to describe a general approach to software development. All agile methods, including Scrum, emphasize teamwork, frequent deliveries of working software, close customer collaboration, and the ability to respond quickly to change.'* – **Mountain Goat Software**



The use of the word Agile derives from the Agile Manifesto. In 2001, a small group of software development thought leaders got together to discuss how software development projects might be managed better. The resulting **Manifesto for Agile Software Development** formed the beginning of the Agile movement.

Manifesto for Agile Software

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

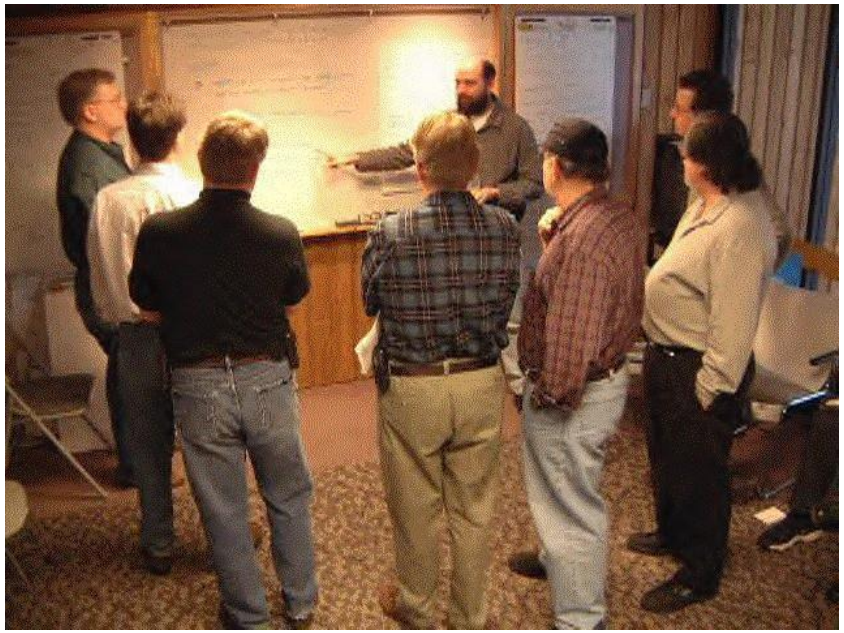
Individuals and interactions *over processes and tools*

Working software *over comprehensive documentation*

Customer collaboration *over contract negotiation*

Responding to change *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.



Agile has as much to do with values and culture as it does software methodology.

Agile развойна организация 101

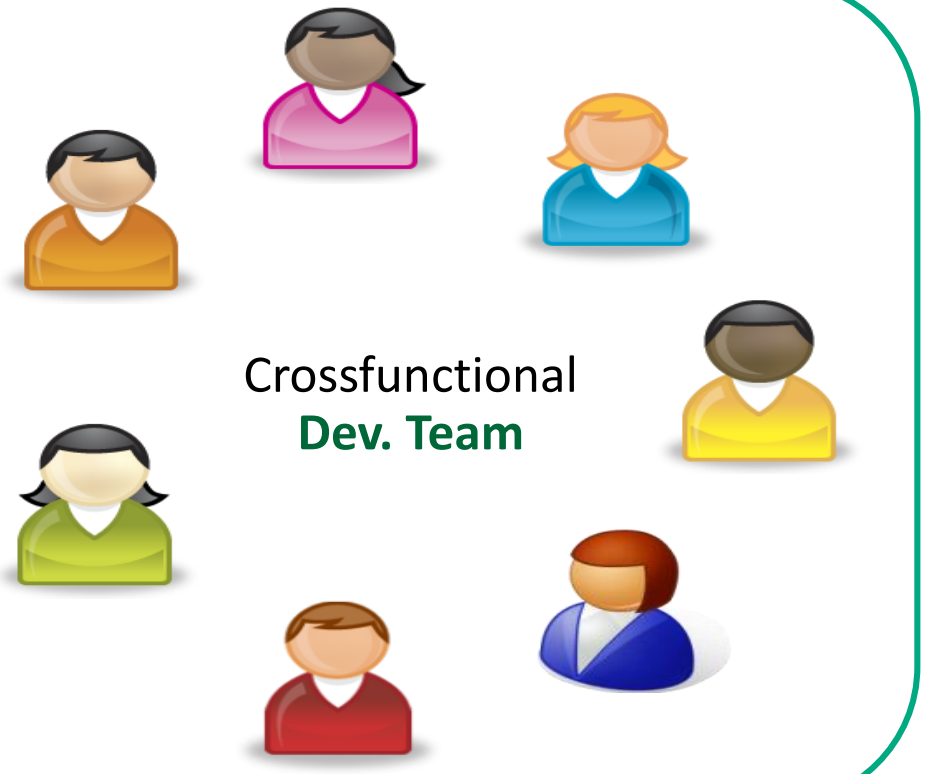
Ефективно екипно взаимодействие

Ефективно лидерство

Product Owner

Scrum Master

User stories:
‘Vertical’ slices
functionality



Crossfunctional
Dev. Team

Potentially shippable product increment

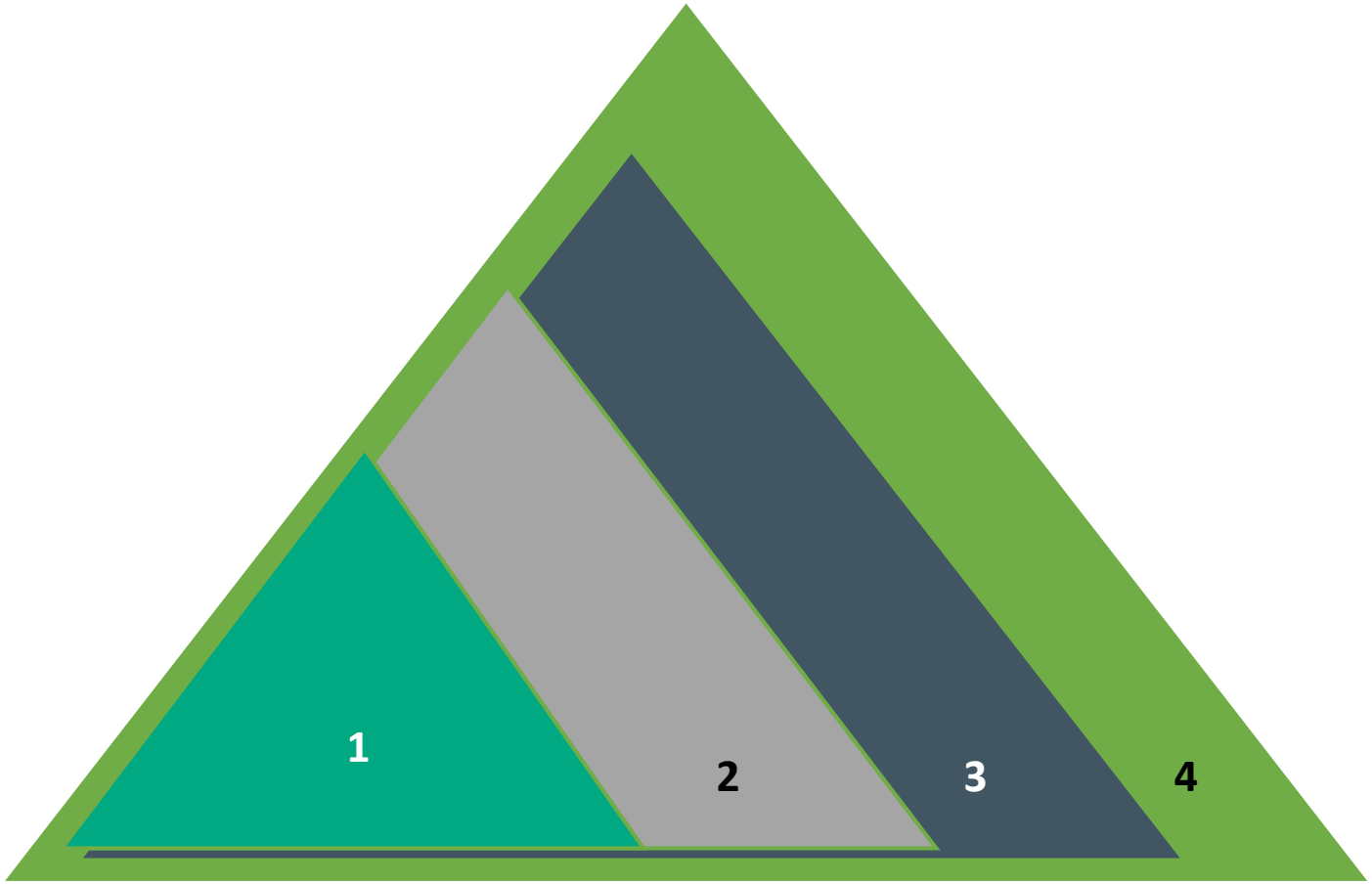
User story

A requirement from a user's perspective

- *„As a parent, I want a comfortable, affordable home so that my spouse and I can raise our family.“*
- *„As the client of a boutique hotel, I want a comfortable room with a personal feel to it so that I have an unexpectedly stimulating night away from home.“*

“As a frequent flyer, [role]
I want to be able to log in, [goal]
so I can access my account information” [business value]

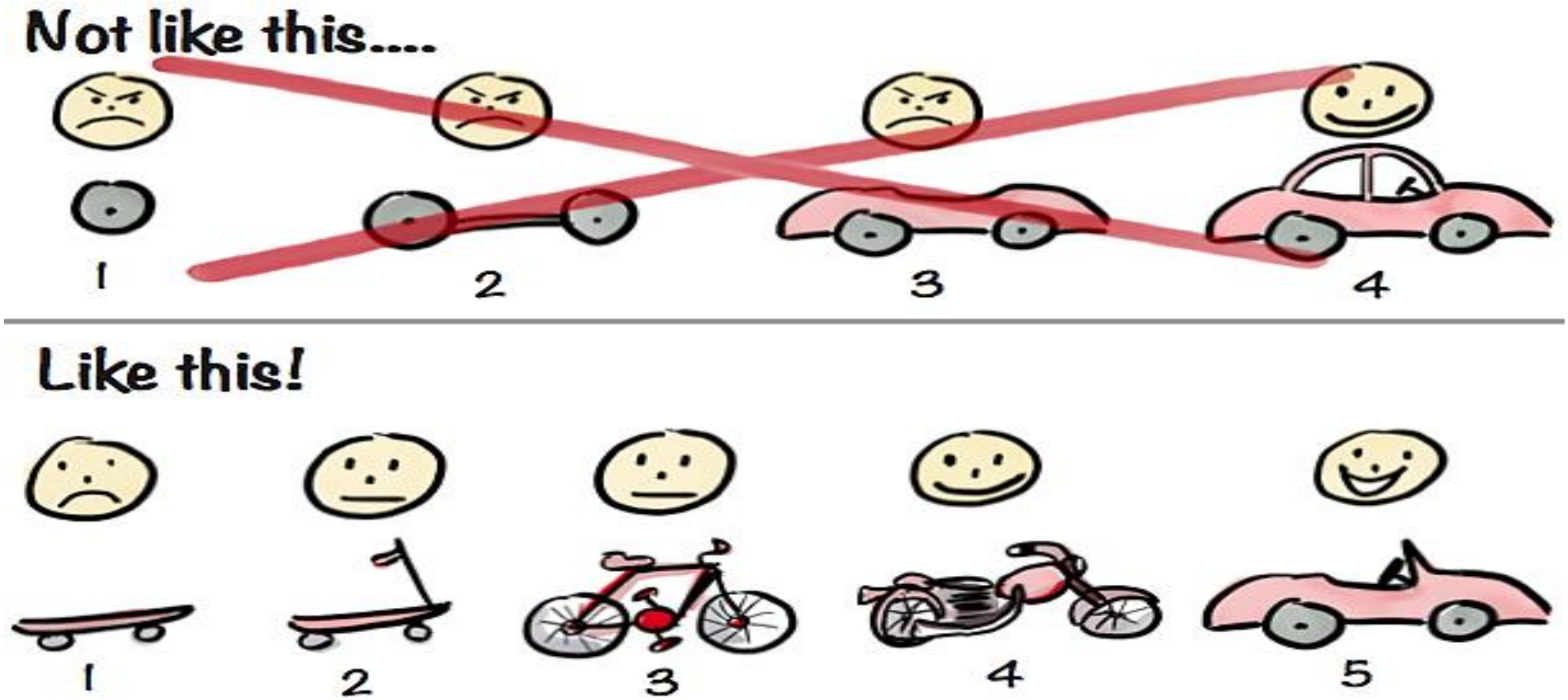
‘Vertical slices’ of end-to-end functionalities



PO decides what is in the next iteration. (roadmap etc.)
The Team commits to deliver as much as possible, without any compromises on quality.

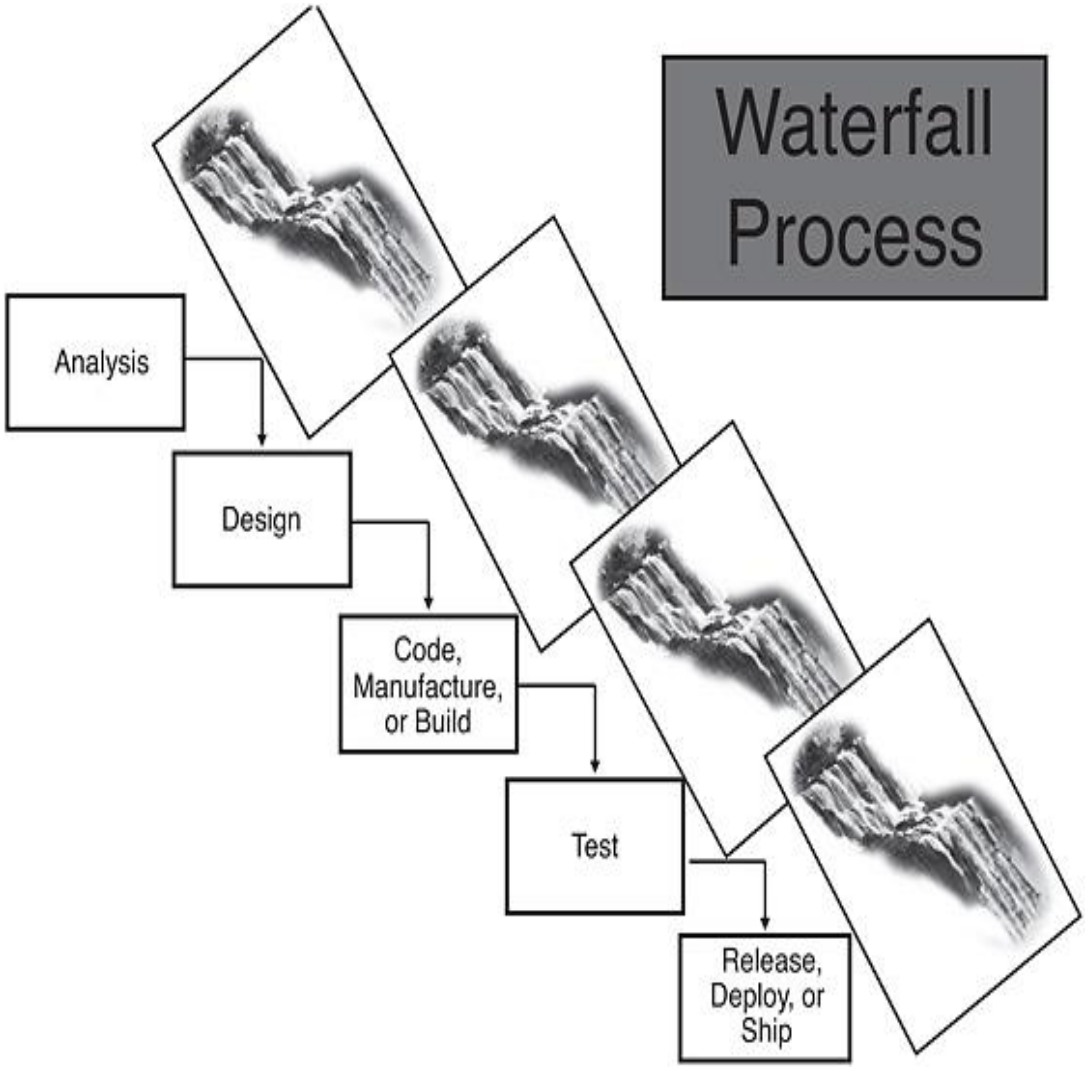
Съпоставка с MVP теорията и практиката

(Minimum Viable Product)



Ключови разлики между Agile и Waterfall

Waterfall



Agile

* Have a dedicated team. (Working on only one project at a time.) Cross-functional and collocated and self-directed.

* Meet as a group and have the customer pick the most important features from a pre-defined list *(Backlog)* to do first.

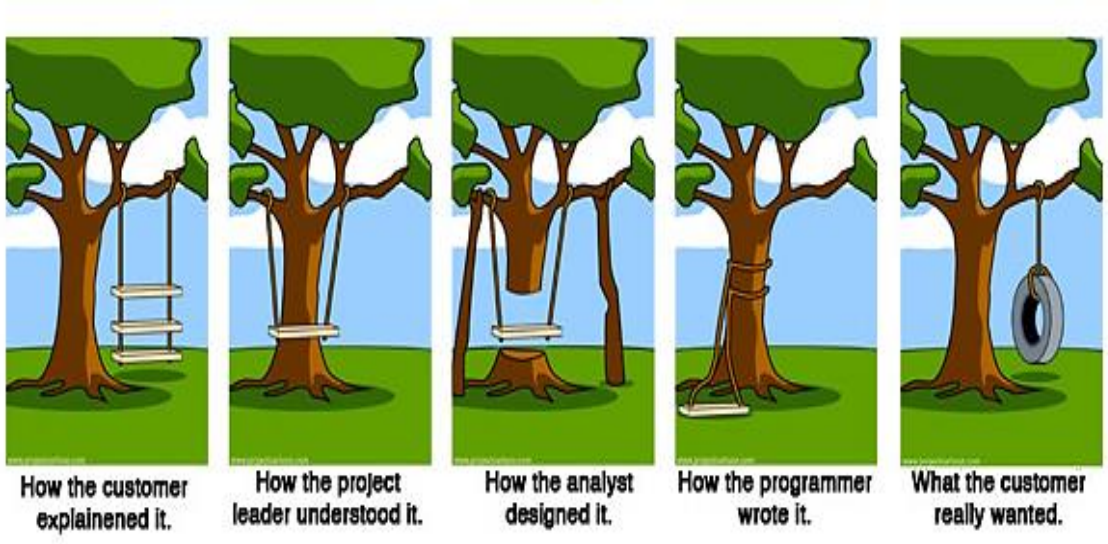
* Estimate how long each item will take; not in duration, but in difficulty.

* Commit to creating the number of features that will fit into the time period of your Iteration, usually 1-2 weeks. (During this time, there will be **no** customer changes).

* Display the stories or features you will create during an Iteration on sticky notes in a highly visible area.

Ключови разлики между Agile и Waterfall

Waterfall



"Колко вероятно е да препоръчате компанията, услугата или продукта на колега или приятел?"

% Promoters - % Detractors = % Net promoters

Agile

DO:

* Each day, have a stand-up meeting where each person says what they completed yesterday and what they will do today and what is in their way.
* As you complete each task, you immediately test it or check it for quality. Then you move the sticky note to a "completed area".
* If there is still time in the day or the Iteration, choose another task or offer to help someone who is behind.

CHECK:

* Do all quality checks and testing during this 2 to 4 week iteration. Usually it's more like this…you create something and you immediately test it…because if your work in incorrect it could cause a problem for someone else.
* At the end, demonstrate or show the customer the features, functions, or items you have completed and get their feedback.

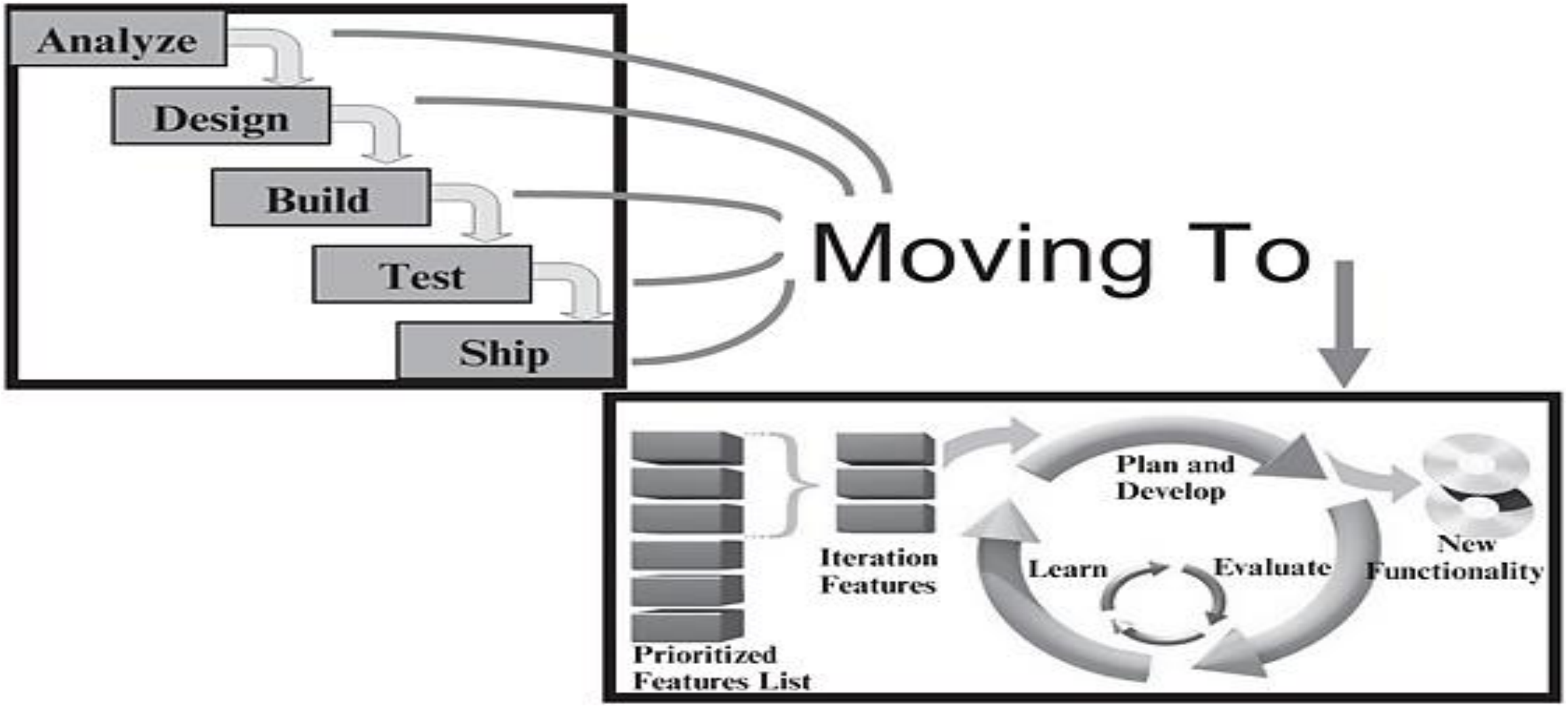
At the end of each Iteration, place any unfinished items or work discovered into the Backlog (pool of features and stories yet to be done).

* Check your velocity (speed with which the team has completed features or stories), and adjust your work for the next Iteration. unfinished items or work discovered into the Backlog (pool of features and stories yet to be done).

* Do a Retrospective, a look back to see what worked and what needs to be changed for the next Iteration. (Why wait until the end of the project, when it's too late.)

Ключови разлики между Agile и Waterfall

Waterfall	Agile



Ключови разлики между Agile и Waterfall

Waterfall
Принципи и практики

- **Processes & tools**
- Comprehensive Documentation
- Contract Negotiation
- Following a Plan

Agile
Agile Manifesto ценности

- **Individuals & Interactions**
- Working Software
- Customer Collaboration
- Responding to Change

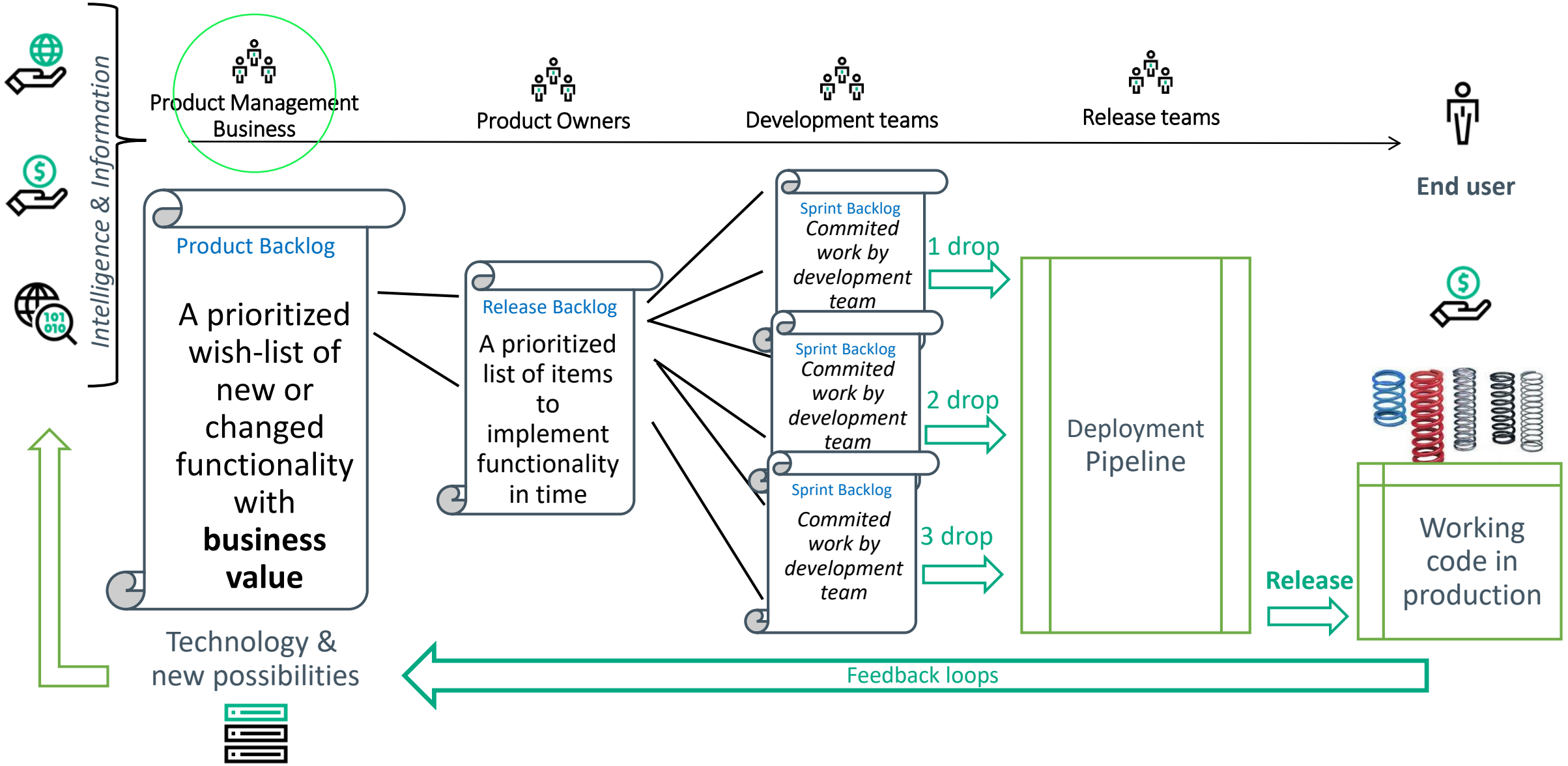
Силни страни

- *Speed (Time) to Market*
- *Product Availability and Delivery*
- *Customer Involvement*
- *Ability to Change Requirements*
- *Team Member Structure*
- *Team Member Interactions*
- *Processes*
- *Customer Acceptance*
- *Managing Changing Priorities*
- *Project Visibility*
- *Team Productivity*
- *Identifying and Resolving Project Issues*

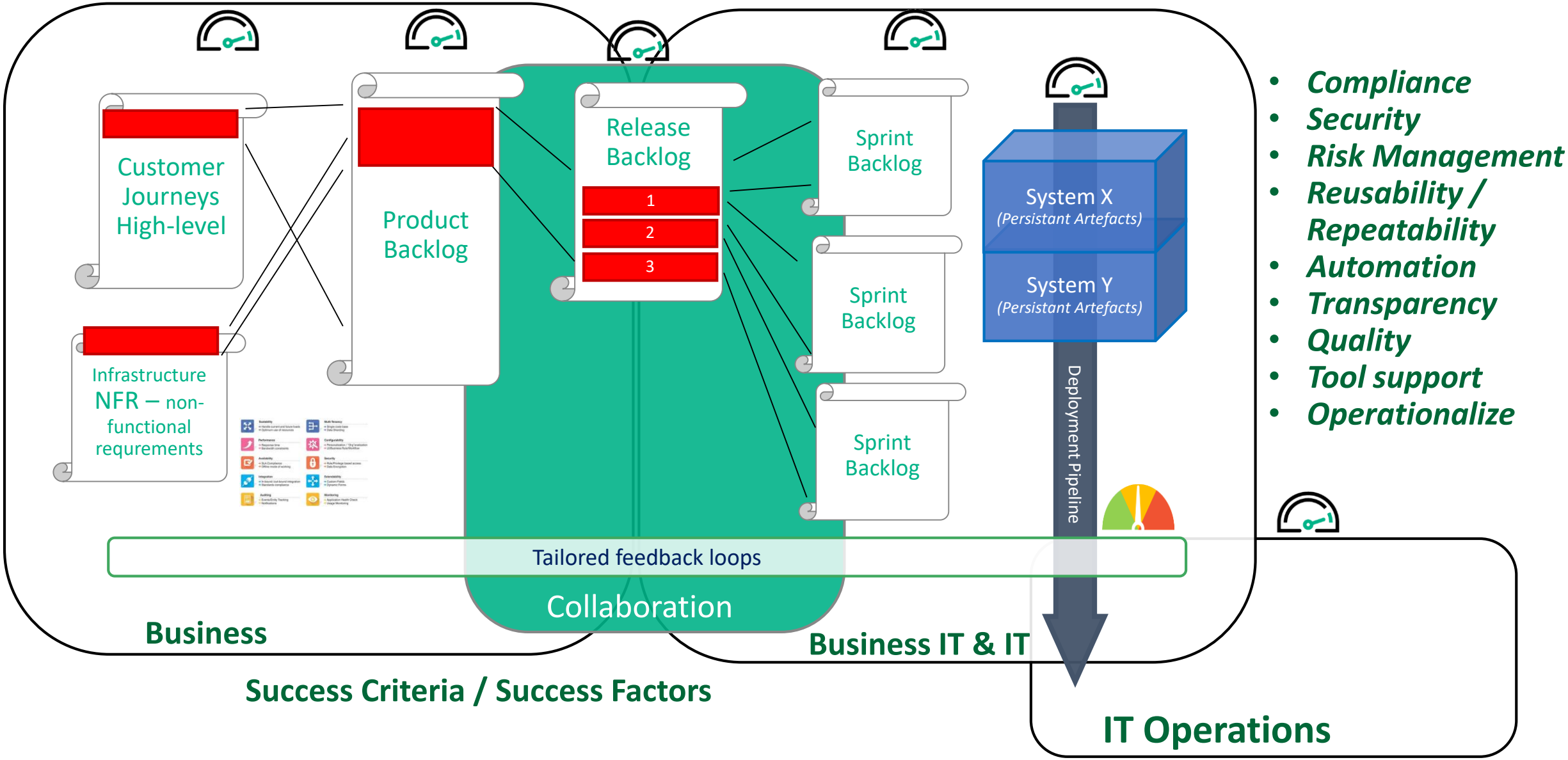
Agile Management / Governance

ТЕМА II, МОДУЛ II «СПЕЦИФИКА ПРИ УПРАВЛЕНИЕТО НА СОФТУЕРНИ ПРОЕКТИ»

Agile Product Development Core



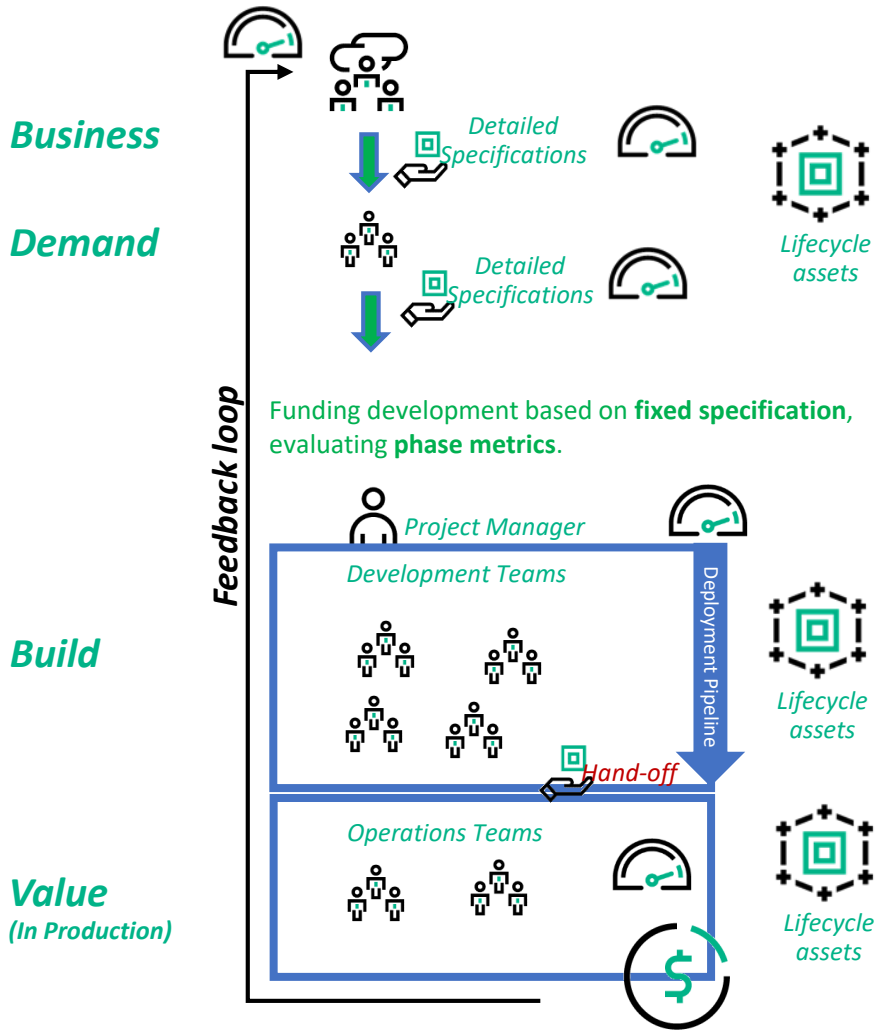
Agile Product Development Expanded



- ***Compliance***
- ***Security***
- ***Risk Management***
- ***Reusability / Repeatability***
- ***Automation***
- ***Transparency***
- ***Quality***
- ***Tool support***
- ***Operationalize***

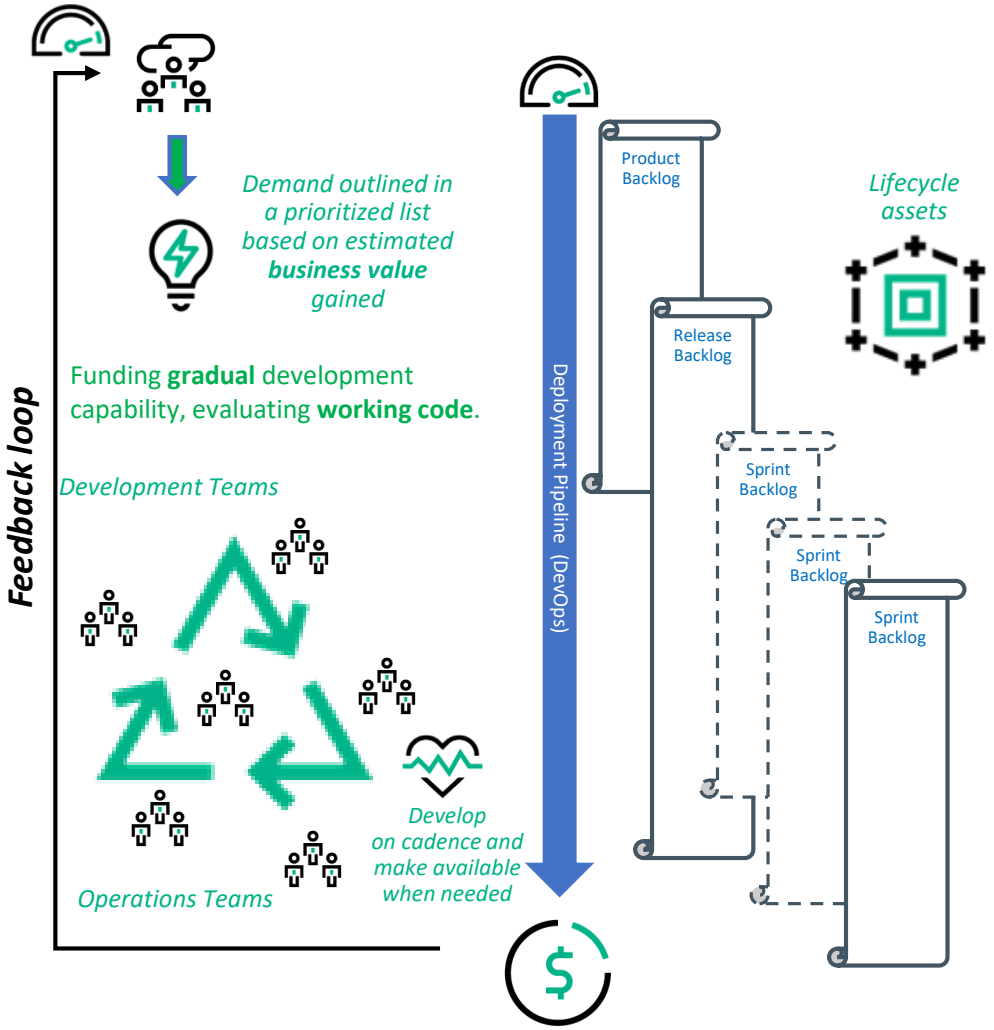
Agile Project Delivery Management

Waterfall Model



Vs.

Agile Model



Допълнителни бележки

Business

Demand

Build

Value
(In Production)

Waterfall Model

Backward focus:

1. Looking at how much **time** have we **spent** fulfilling **specifications** estimates
2. Using previous experienced individuals and expectations to **predict** the size of the **project**
3. Using rather exact estimations to determine the project plan **not expecting change** during development
4. **Lifecycle artefacts** are **rarely re-used** between projects lessens the benefits of automation
5. Financial criterias based on the projects expectated needs, making detailed change. **Success is: delivering on time and under budget**

Measurement mechanics:

1. Using **estimations upfront** to determine final outcome using project plan to visualize progress
2. **Sizing** is done in **absolute** terms tracking hours and money spent
3. Testing activities to track progress
    1. Test against test cases defined based on upfront specifications
    2. Testing is done when something is testable which most often is late i the process
4. **Sequential development** with many **hand-overs** between groups
5. **Metrics** are generally **easy to retrieve** and to aggregate into models

Trust:

1. Relying on inital expectations and clear requirement specifications to define what work is to be done
2. Celebrating hand-overs as achieved goals, putting plan before the ever changing reality measuring the parts and not the whole

Agile Model

Forward focus:

1. Looking at how much time do we estimate we will need to spend develop **working code**, to prove the point driving the **collaboration** forward.
2. Using the historical **velocity** of the team to estimate how much change to feed into a sprint, including stretch targets.
3. Outlining demand in epics/stories **expecting change** during development
4. **Lifecycle artefacts** are **implemented** for future **re-use** based on automation and continous integration
5. Financial criterias based on maintaining a capability and retaining business value created. **Success is: delivering increased business value**

Measurement mechanics:

1. Using **daily team estimations** of remaining work (burndown charts) providing working code to have a continous dialogue about the final outcome
2. **Sizing** is done in **relative** terms using story points tracking time to completion and functionality implemented
3. Testing activities to track progress
    1. Testing is used in-sprint to help developers understand when they are done based on acceptance criterias for each specific user story.
    2. Testing out-of-sprint is done to satisfy business stakeholders need to trust the outcome of deploying new code into production.
4. **Gradual development** of features starting with a skeleton and gradually adding functionality in close **collaboration** with business to prioritize value as early as possible
5. **Metrics** are often soft and **hard** to define, understand and **retrieve**

Trust:

1. Outling demand relying on all involved parties to contribute with their knowledge to deliver the best result possible
2. Removing hand-overs and expecting collaboration by all, including financial criterias, to create real business value

Delivery models - *Разлики*

	Waterfall	Agile
Business	• Wrong design decisions will affect a full development cycle. • Approval decisions based on estimations of the future need. • Funding is based on estimations of resource need	• Faulty design decisions will affect the sprint • Approval decisions based on delivered working code • Funding is based on maintaining a capability
Demand	• Demand is captured in dialogue and finalized before breaking down to development requirements • Requirements are defined upfront in detail and agreed on by all stakeholders for development teams to realize	• Demand is captured on a high level and outlining epics/user stories to guide development activities. • Requirements continously worked on together with stakeholders expecting stakeholder involvement
Build	• *Tasks pushed to team members* • *Perceived control based on estimates* • *Direct management* • *Fixed metrics and phase gates*	• *Tasks pulled by team members* • *Control based on measurable delivered working software* • *Servant leadership* • *Relative metrics and working software*
Value *(In Production)*	• *SLA's based on delivered items* • *Value measured on delivered items based on specification and initial estimates*	• *SLA's based on delivered capacity* • *Value measured on continuously re-evaluated functionality made available to customer*

When you fully and correctly can predict the future need, then Waterfall can offer proper transparency and control. Otherwise Agile is more suited to let business stakeholders drive the functionality to what really create business value for them.

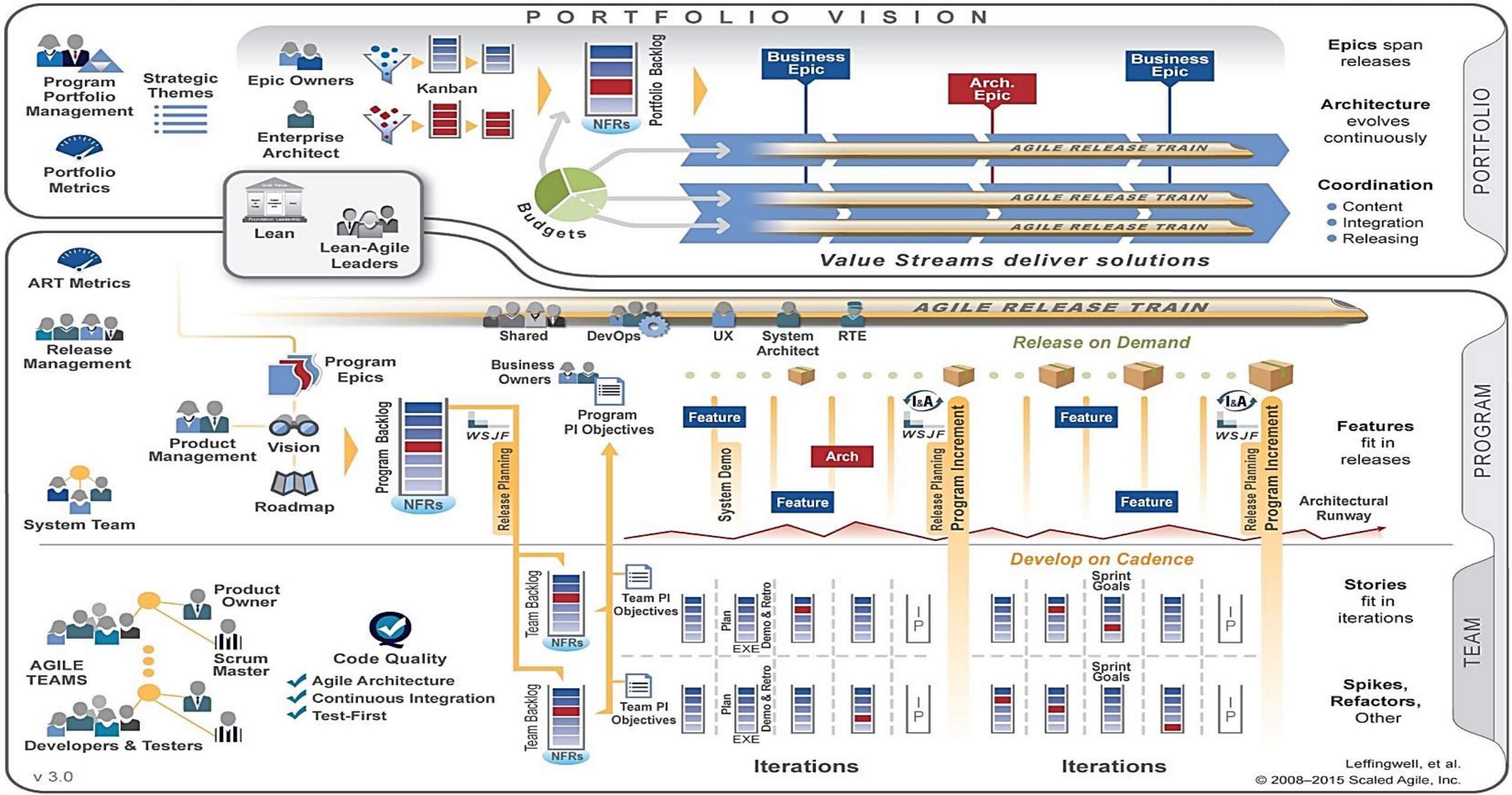
Outsourcing models – *разлики*

	Staff Augmentation	Managed Service	**Agile Factory**
Business	• Resource provisioning • Rate card negotiations • Resource availability	• Service definition • Contract negotiation • Future value estimated upfront	• Capability definition • Contract collaboration • Future value gradually estimated and re-evaluated in a controlled way
Demand	• Client defined	• Short backlogs / specifications • Detailed requirements / user stories • Hand-over with stakeholder approval • Scope set • Expectations based on estimates	• Extensive backlogs prioritized • High-level requirements / user stories • Stakeholder involvement • Scope continuously re-evaluated • Expectations based on real progress
Build	• Client defined	• Tasks pushed to team members • Perceived control based on estimates • Direct management • Fixed metrics and phase gates	• Tasks pulled by team members • Control based on measurable delivered working software • Servant leadership • Relative metrics and working software
Value ***(In Production)***	• Client defined	• SLA's based on delivered items • Value measured on delivered items based on specification and initial estimates	• SLA's based on delivered capacity • Value measured on continuously re-evaluated functionality made available to customer

An Agile factory truly establish a long term relationship between the company and the client, based on continuously proven trust of delivery.

Scaled Agile Framework® http://www.scaledagileframework.com/

SAFe®



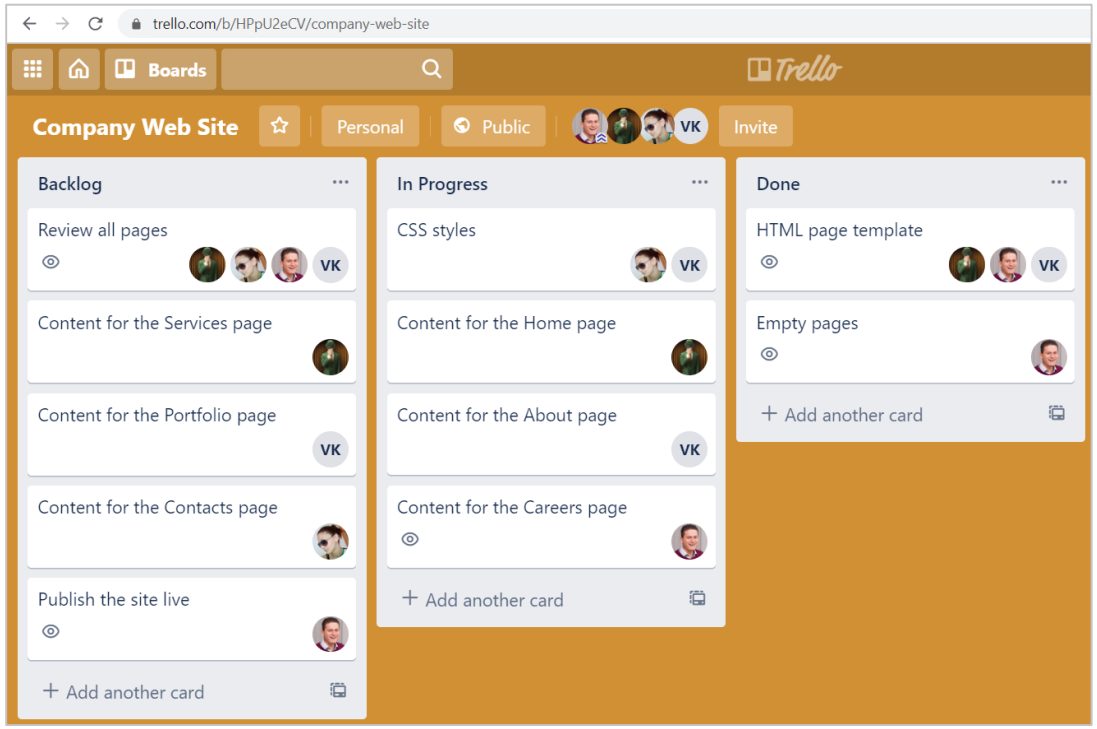
Scrum

ТЕМА III, МОДУЛ II «СПЕЦИФИКА ПРИ УПРАВЛЕНИЕТО НА СОФТУЕРНИ ПРОЕКТИ»

Project Trackers & Kanban бордове

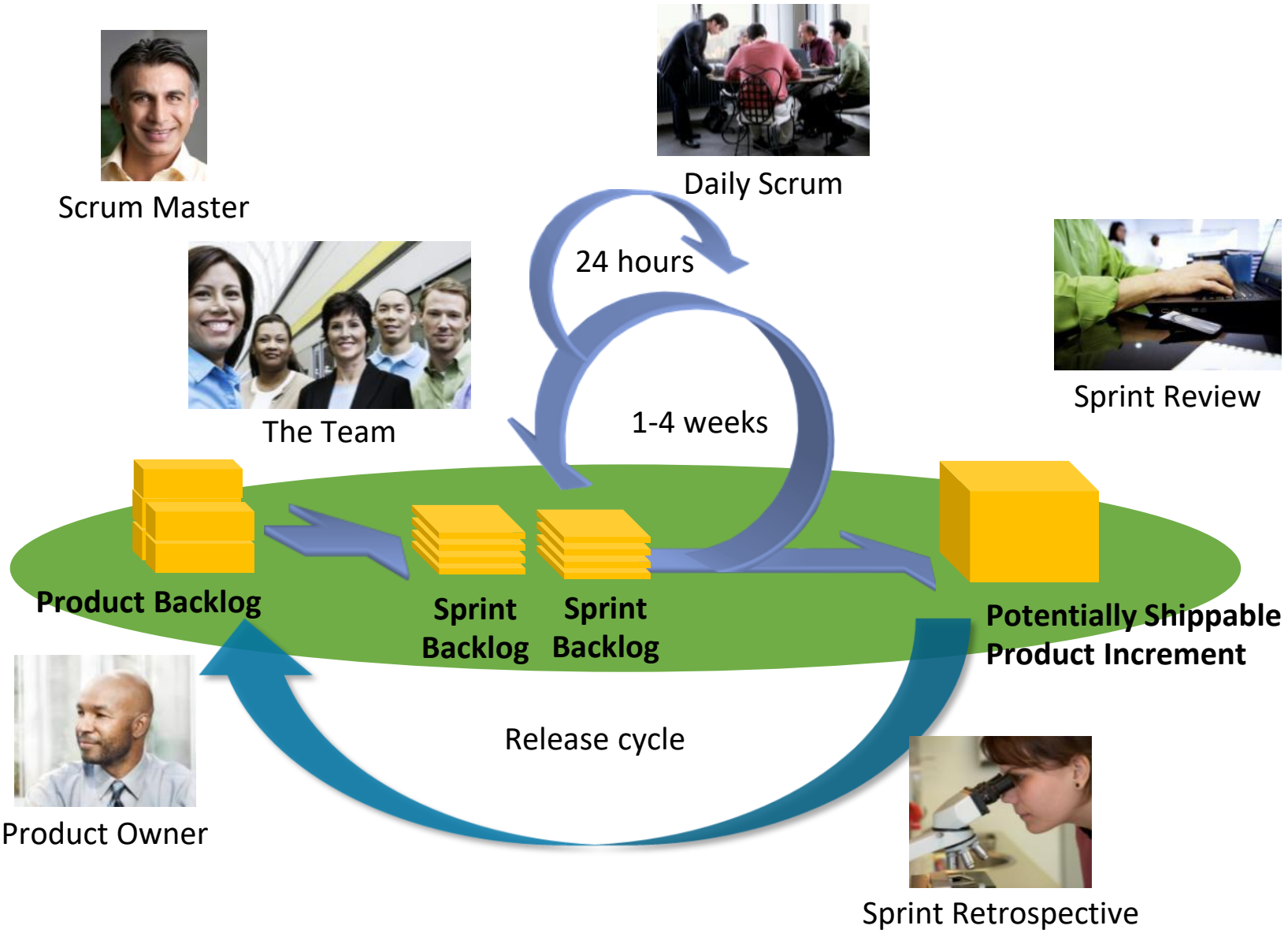
- **Project trackers – организират и следят изпълнението на работните задачи**
  - **Задачите включват:** description, sub-tasks, assigned people, deadline
- **Kanban бордове – визуализират работата върху проекта**
  - Типични секции:
    - Backlog
    - In Progress
    - Done
  - Примери:
    - Trello,
    - GitHub Projects

https://trello.com/b/HPpU2eCV/company-web-site



A **Kanban board** is an agile project management tool, designed to help visualize work, limit work-in-progress, and maximize team's efficiency.

The Scrum Framework



The Product Owner

(http://www.scrumguides.org/scrum-guide.html#team-po)

- The Product Owner is responsible for **maximizing the value** of the product and the work of the Development Team. How this is done may vary widely across organizations, Scrum Teams, and individuals.
- The Product Owner is the sole person responsible for **managing the Product Backlog**. Product Backlog management includes:
  - *Clearly expressing Product Backlog items;*
  - *Ordering the items in the Product Backlog to best achieve goals and missions;*
  - *Optimizing the value of the work the Development Team performs;*
  - *Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,*
  - *Ensuring the Development Team understands items in the Product Backlog to the level needed.*
- The Product Owner may do the above work, or have the Development Team do it. However, the Product Owner remains **accountable**.
- The Product Owner is **one person**, not a committee. The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a Product Backlog item's priority must address the Product Owner.
- For the Product Owner to succeed, the entire organization must respect his or her decisions. The Product Owner's decisions are visible in the content and ordering of the Product Backlog. No one is allowed to tell the Development Team to work from a different set of requirements, and the Development Team isn't allowed to act on what anyone else says.

The Scrum Master

(http://www.scrumguides.org/scrum-guide.html#team-sm)

The Scrum Master is responsible for **ensuring Scrum is understood and enacted**. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.

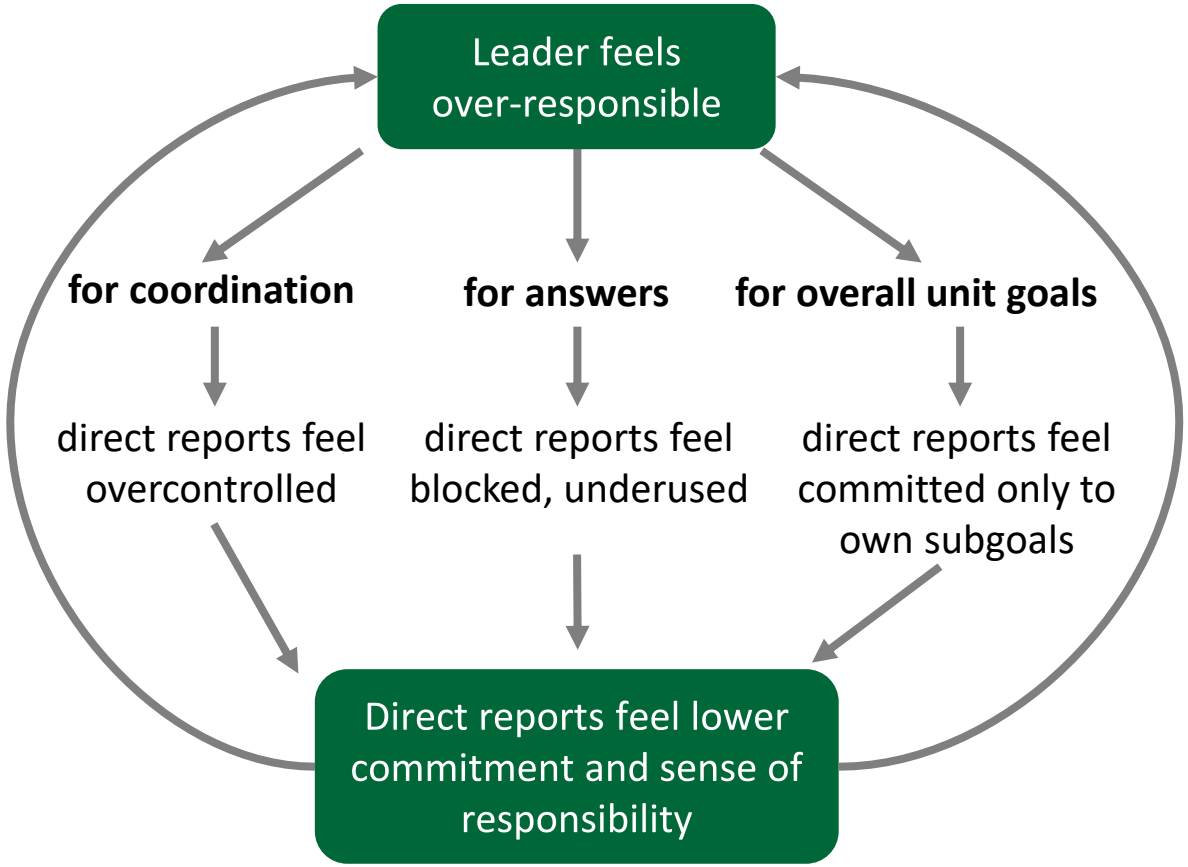
The Scrum Master is a **servant-leader** for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

Scrum Master services include:

- Finding techniques for effective Product Backlog management;
- Helping the Scrum Team understand the need for clear and concise Product Backlog items;
- Understanding product planning in an empirical environment;
- Facilitating Scrum events as requested or needed.
- Coaching the Development Team in self-organization and cross-functionality;
- Helping the Development Team to create high-value products;
- Removing impediments to the Development Team's progress;
- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact Scrum and empirical product development;
- Coaching the team on continuous performance improvement.

Капанът на добрия *„отговорен лидер“*

Допълнителни бележки



(From *Leading SAFe* course material, summarizing *Managing for Excellence*, David Bradford and Allan Cohen)

Agile лидерът – ментор

Escape the responsibility trap with a post-heroic, lean leadership style

Допълнителни бележки

ПОВЕДЕНИЕ	ПОЛЗИ
▪ Creates a team jointly responsible for success	▪ Increased direct report ownership and responsibility
▪ Asks, *"How can each problem be solved in a way that further develops my people's commitment and capabilities?"*	▪ Increased employee engagement and motivation
▪ Work is developing other's abilities	▪ Allows leader to spend more time managing laterally and upward
	▪ There is no limit to the power of getting things done

(From *Leading SAFe* course material, summarizing *Managing for Excellence*, David Bradford and Allan Cohen)

Agile лидерът - диригент

Допълнителни бележки

ХАРАКТЕРИСТИКИ

- The central decision maker, nerve center, coordinator
- Orchestrates all individual parts of the organization into a harmonious whole
- Subtle and indirect manipulation to their solution
- Manages across individuals, teams, and departments
- Work is coordinating others

ПРЕДИЗВИКАТЕЛСТВА

- Narrows the focus of direct reports to their own areas
- Narrows the focus of direct reports to their own areas
- Use systems and procedures to control work
- Works harder and harder, without realizing full potential

(From *Leading SAFe* course material, summarizing *Managing for Excellence*, David Bradford and Allan Cohen)

Развитие на PM уменията с Agile

Agile предоставя възможности за настройка и развитие на упр. умения

- Servant Style of Leading
- Facilitation Skills
- Sales Skills
- Coordination Skills
- Collaboration Skills
- Team Building
- Conflict Resolution
- Group Decision Making

Допълнителни бележки

ИЗСЛЕДВАНЕ НА КАЗУСИ

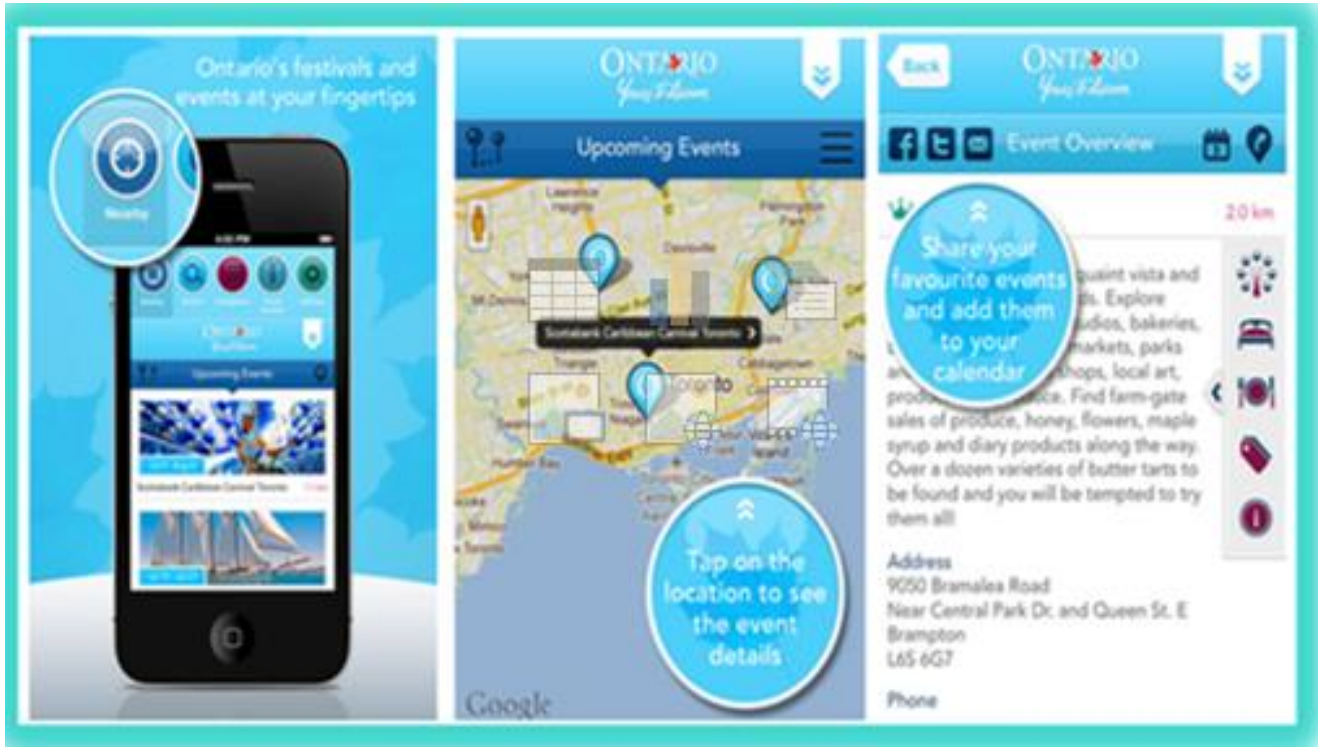
ТЕМА II & III, МОДУЛ II «СПЕЦИФИКА ПРИ УПРАВЛЕНИЕТО НА СОФТУЕРНИ ПРОЕКТИ»

Ontario Tourism Marketing Partnership Corporation (OTMPC)

Case Study

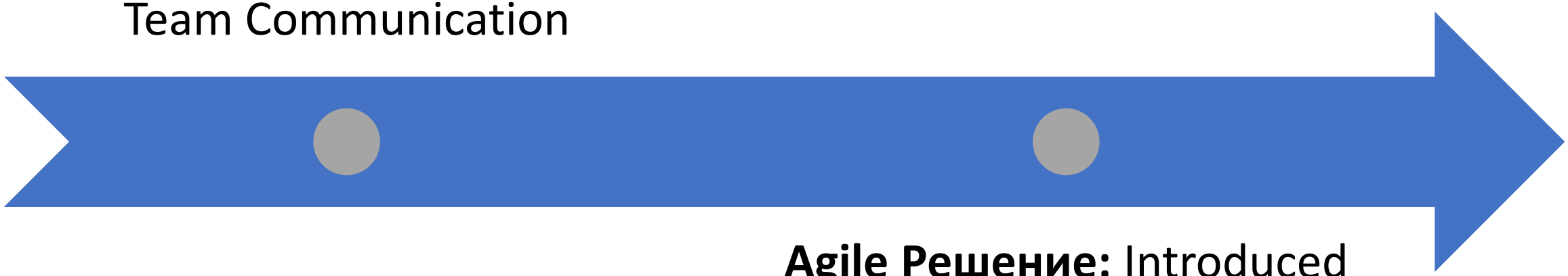
Background:

- *OTMPC vision is to inspire travelers to visit Ontario*
- *Primary information tool to achieve mandate is the Tourism Consumer Information System (TCIS)*
- *Project TCV: $10 Million*



Case Study – OTMPC

Проектно предизвикателство: Improve Team Communication



Agile Решение: Introduced Daily Scrum Meeting

3 ключови Daily Scrum въпроса

1	What did you work on?
2	What are you working on next?
3	What if any impediments?

Обратна връзка от проектния екип

Allowed for efficient communication amongst a dispersed team as each team lead was aware of what other teams were working on and any impediments they were facing. It helped create a more unified team." – **D. Ramkissoon, PM**

"The project involved multiple teams working on completely different stacks e.g .NET, Java etc. The standup meetings were important in helping the teams to identify as a single team. Also, the standup meetings were kept short and that forced the team leads to focus on immediate needs and issues only." - **H. Minhas, Project Architect**

"Incorporating Agile practices with an account / client that mandates the use of a waterfall methodology is quite challenging; being able to 'cherry-pick' certain processes such as the daily stand-up, and the internal breakdown of large scale waterfall tasks into smaller sprints greatly increased our teams' ability to quickly identity / respond to roadblocks, and allowed us to make course corrections much earlier than what normally could be achieved following traditional waterfall practices." – **R. Drenkar, Mobile Architect**

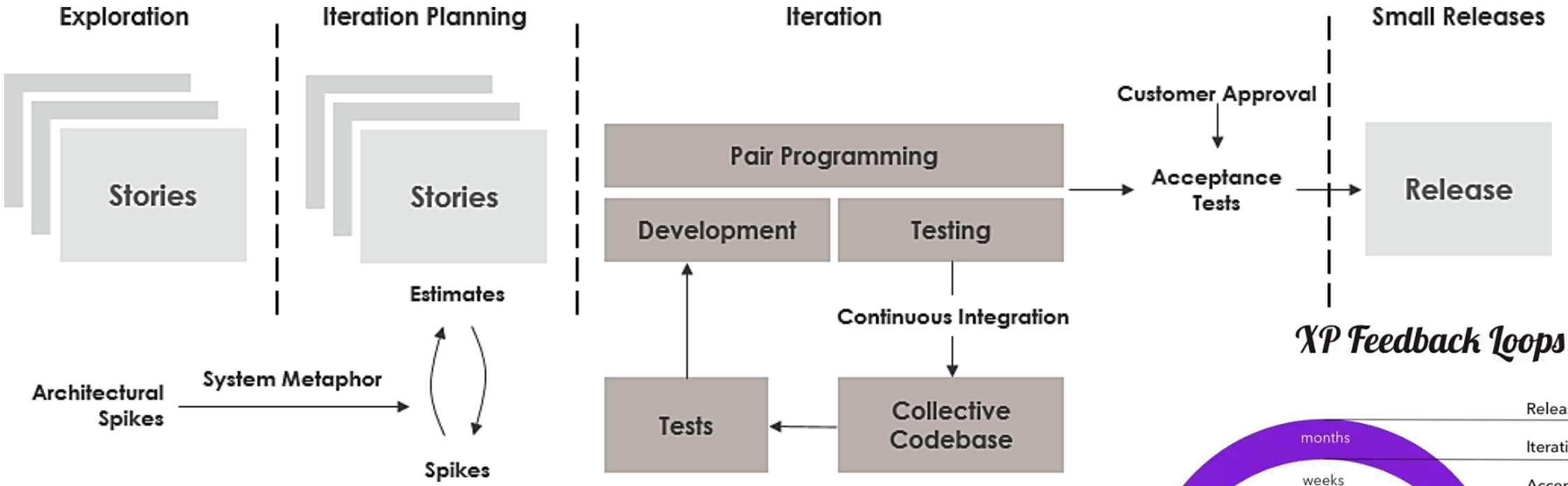
Extreme Programming

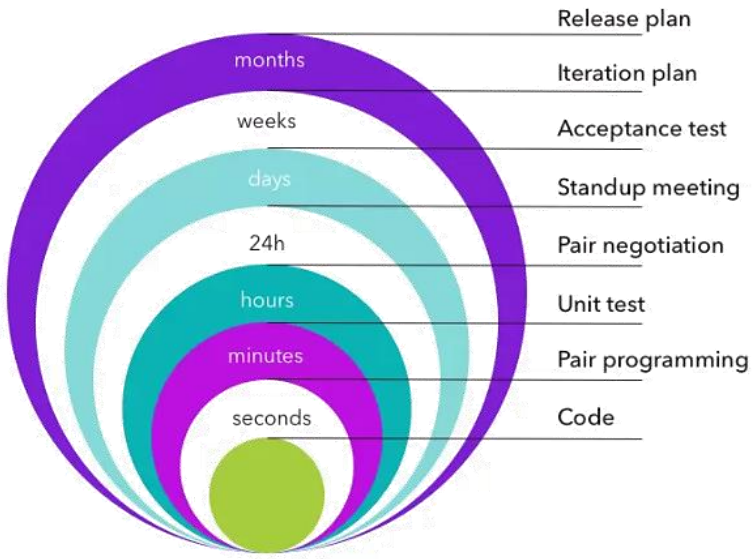
ТЕМА III, МОДУЛ II «СПЕЦИФИКА ПРИ УПРАВЛЕНИЕТО НА СОФТУЕРНИ ПРОЕКТИ»

https://www.agilealliance.org/glossary/xp/

Extreme Programming

Повече фокус върху инженерните практики, по-малко гъвкавост / свобода в сравнение със Scrum





- ***Iteration length:*** Typically one or two weeks long.
- ***Requirements:*** Much more amenable to change within their iterations. As long as the team hasn't started work on a particular feature, a new feature of equivalent size can be swapped into the XP team's iteration in exchange for the un-started feature.
- ***User Story:*** Work in a strict priority order. Features to be developed are prioritized by the customer (Scrum's Product Owner) and the team is required to work on them in that order.

ЗАДАЧА ЗА САМОСТОЯТЕЛНА РАБОТА

Запознайте се с концепцията за DevOps от теоретична и практико-приложна гледна точка.

https://opensource.com/tags/devops

DevOps – какво представлява?

DevOps е комбинация от ……

- Философии,
- Практики, и
- Инструменти

….that **supports** an organization’s **ability to deliver** applications and services at **SPEED and SCALE**

….enabling software to be improved and delivered at a **faster pace** than traditional software development processes of the past.

Speed and agility enables businesses to better serve their customers and compete more effectively in the marketplace.

DevOps модел

Under a DevOps model….

- Development (Dev) and Operations (Ops) teams are no longer “siloed” => Team Members from both Dev and Ops come together on the same team, working collaboratively across the application to deliver functionality
- The entire lifecycle, from development and test to deployment and operations, are covered by a single team who possess or develop the range of skills necessary to support all required lifecycle phases and project requirements
- Quality Assurance & Other Functions (like Security or Infrastructure) may also become more tightly integrated with development and operations throughout the application lifecycle to deliver the full functionality front to back

These teams use

- Practices to automate processes that historically have been manual, slow or disparate
- Technology stacks and tooling to help build, operate and evolve applications quickly and reliably
- Tools to help team members independently accomplish tasks through automation that normally would have required help from other teams (for example, deploying code or provisioning infrastructure)

DevOps в практиката на проекта

- **Continuous Integration**: a software development practice where developers regularly (hourly and daily) merge their code changes into a central repository, then automated builds and tests are run at check-in, providing instant feedback to the developers on issues or success.
- **Continuous Delivery:** a software development practice where code changes are automatically built, tested, deployed to an environment and prepared for a release to production.
- **Micro services Architecture**: a design approach to build a single application as a set of small services, which are built around business capabilities and scoped to a single purpose.
- **Infrastructure as Code**: a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration, allowing interaction with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources; code is used to automate operating systems and host configurations, operational tasks, and more.
- **Monitoring & Logging**: metrics and logs are used to monitor application and infrastructure performance, noting impacts to the experience of the product's end users then rapidly feeding information back to development teams for future improvements.
- **Communication & Collaboration**: a key cultural aspect of DevOps; the use of tooling to automate the software delivery process promotes collaboration by physically bringing together the workflows and responsibilities of development and operations into a single team, eliminating silos and wait times, and speeding delivery to market; use of enhanced knowledge sharing and communication techniques such as chat or IM enhance collaboration between team members.

Защо DevOps е от значение?

- The world is evolving and becoming much smaller through the advancements of the internet.
- “Mom and Pop” Organizations now have access to consumers miles away, instead of just in their own neighborhood, greatly increasing their opportunities to reach a wider audience with their goods and services.
- The global economy is forcing companies in all industries to have a greater online presence in order to do business with consumers around the world.
- Software and applications are becoming an integral part of a businesses online presence, with a greater focus on user experience to capture and retain new customers.
- Companies interact with their customers worldwide through software delivered as services or applications on all sorts of devices and through 24/7 operations and availability.
- Software is used to increase operational efficiencies by transforming every part of the value chain.
- Companies in today’s world must transform how they build, package and deploy software enhancements and the speed at which they deliver applications to the market to remain competitive.

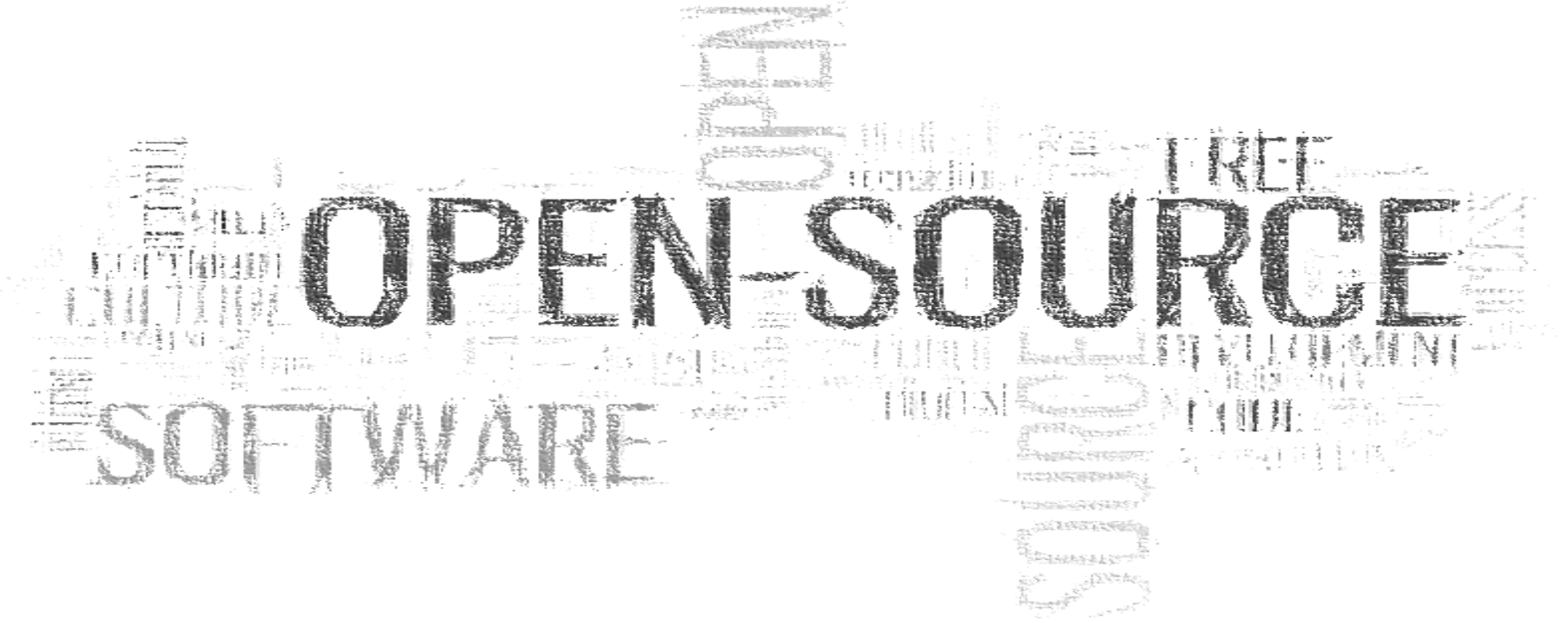
ЗАКЛЮЧЕНИЕ

ЗРЕЛОСТЕН МОДЕЛ ЗА AGILE DEVELOPMENT & DEV-OPS

Нива на зрялост	Организация	Екипно взаимодействие	Build Management Continuous Integration	Continuous Delivery / Deployment	LifeCycle Management & Compliance	Testing	Data & Integration Management
Optimized	Lean and agile are part of the organizational culture. Continuous learning and optimization of the work processes.	Effective knowledge sharing and individual empowerment.	Teams regularly meet to discuss integration problems and resolve them with automation, fast feedback and better visibility.	Environments are managed effectively. Provisioning is fully automated. Standard topologies are available for common components.	Full portfolio and lifecycle management are in place integrating user requirements, development, testing, staging and production.	Testing is fully automated, production rollbacks are rare, defects are found and fixed immediately.	Release to release feedback loop of database and integration performance and deployment processes
Measured	Communities of practice support agile habits and high maturity across the organization. Measurement systems are in place to keep track of business value delivered. Inefficiencies are identified and acted upon.	Integration between development, testing and operations teams are implemented even with teams in multiple countries. Success is celebrated. People are fully respected and recognized for their contribution.	Build metrics are gathered, made visible and acted upon. Lessons are learned and continuous improvement is in place.	Deployments are orchestrated and managed. Release and rollback processes are tested.	Environment and application health is monitored and proactively managed. Cycle times are monitored.	Quality metrics and trends are tracked. Non functional requirements are defined and measured.	Database upgrades and rollbacks are tested with every deployment. Database performance monitored and optimized. Integration is performed through message queues enabling scale-up/scale-down.
Defined	Disciplined agile delivery processes and practices are implemented. Continuous improvement is in place. Appropriate governance is in place.	Co-located development, testing and operations teams are starting to work together towards a common goal. Communications and feedback looks are initiated across the whole supply chain.	Every time a source code change is committed automated build and test cycle is performed. Dependencies are managed and scripts and tools re-used	Software is deployed using a fully automated, self-service process. Same deployment process is used for every environment.	Lean portfolio management. Change management and approval processes are in place and enforced.	Automated unit and acceptance tests. Testing is part of the development process. Feedback loops are in place and continuous improvement is measured and managed.	Database and integration are included in the deployment process.
Managed	The organization is starting to embrace agile habits within development, operations is still seen as separate. Consistency across teams is variable.	Some knowledge sharing activities get under way. Teams are starting to communicate within development. Standard work practices are defined and mostly followed.	Automation is implemented in the build and test phases, but is still siloed. No central infrastructure in place.	Deployment are partially automated to some environments. Some environments can be provisioned automatically. Some middleware and database component are provided centrally.	Lifecycle Management is painful and infrequent but releases are reliable. There is limited traceability from requirement to release. Quality is improving.	Test scripts and test data are generated as part of the development process. These are used for automating some tests.	Database changes are done through the use of automated scripts with versions associated to applications. An Enterprise Application Integration bus is implemented to facilitate integration.
Initial	Development teams work in isolation and use the tools, middleware and components they deem most appropriate to deliver the work.	Poor teamwork, ad-hoc communication & coordination. No knowledge pool. Little sharing. Success achieved primarily through heroic individual efforts.	Manual and reactive processes, little management of artefacts, documentation, source code. uncontrolled	Software is deployed manually, using environment specific binaries. Environment is provisioned manually.	Infrequent and unreliable releases, manual application lifecycle. Software Quality turns out to be variable.	Manual testing, no test scripts and test data. Typically done after development.	Data migrations are performed manually. Integration is add-hoc and often point to point.

Следващи теми:

Теми № 11, 12: ***Инструменти и практики за бизнес развитие по отворен модел. Конкурентни бизнес предимства на отворения код.***
Модул 3: ***„Бизнес развитие по отворен модел“***




БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

DASKALOV.HR@GMAIL.COM I WWW.DASKALOV.INFO