D6 UNIT 3-A
PLANNING

Work breakdown structure, prioritization and task planning

Work breakdown techniques

- Allow to know the scope of a project
- Are applied before planning the project

- Types (different degrees of detail and information):
  - Work breakdown structure
  - Workflow diagrams
  - Workflow systems

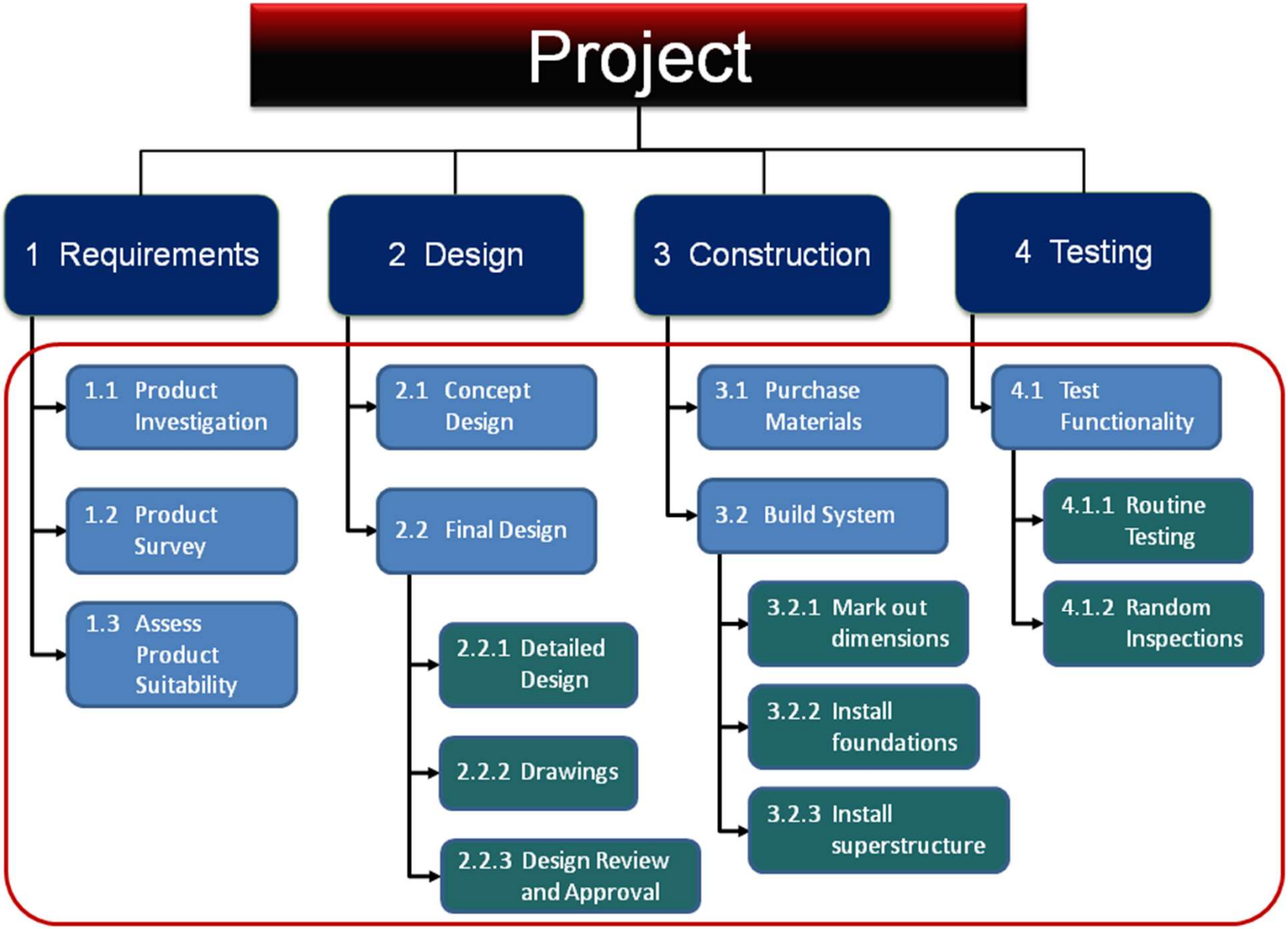
Work breakdown techniques

Work Breakdown Structure (WBS)

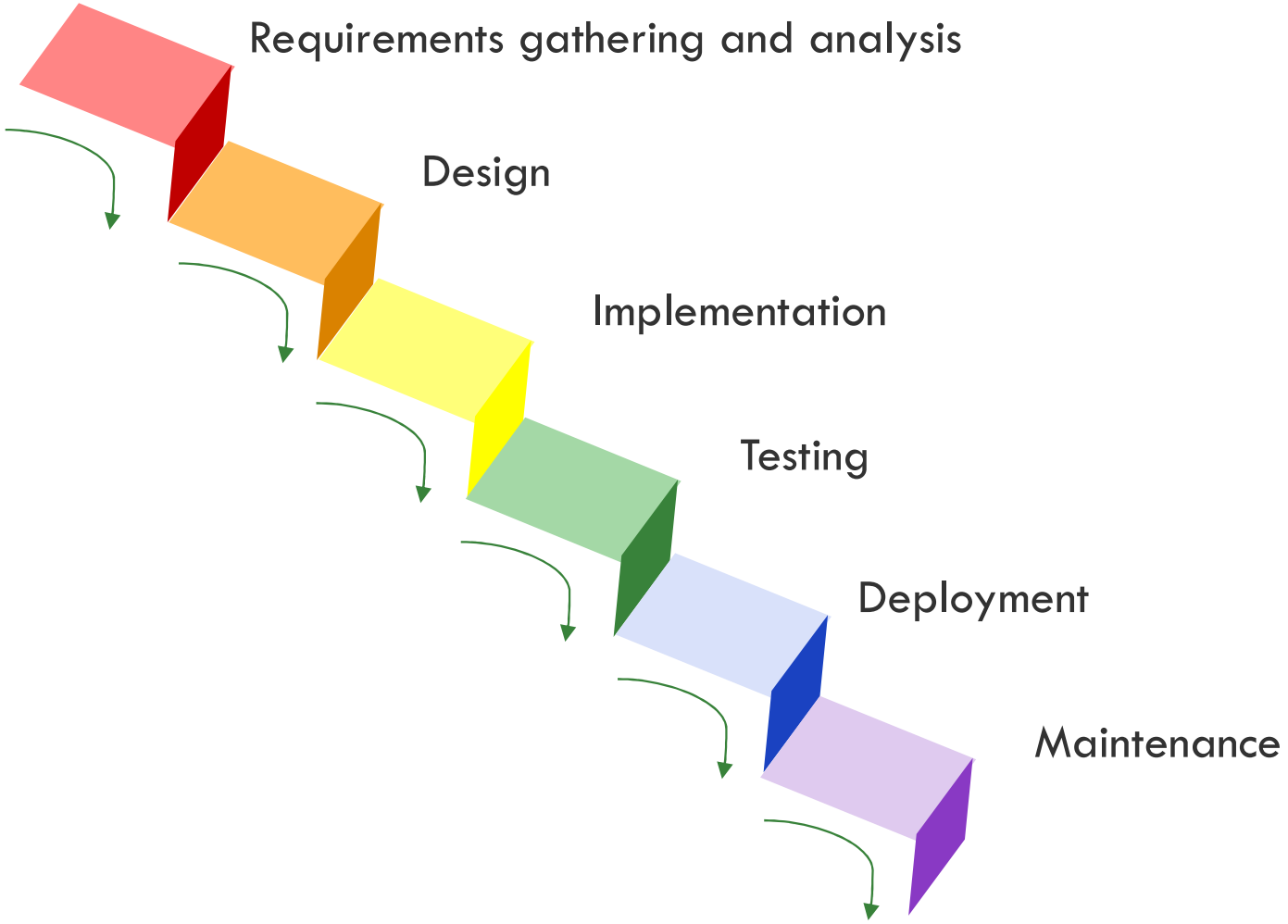
Estructura de descomposición de trabajos (EDT)

- The main goal is to organize and define the scope of the Project.
- Each level represents an increase in the detail of the description.
- Software projects usually imply at least three levels:
  1. Phases in the life cycle
  2. Standard tasks in each phase
  3. Specific tasks for that Project

Work breakdown techniques



Traditional model
Waterfall



Traditional model
Waterfall

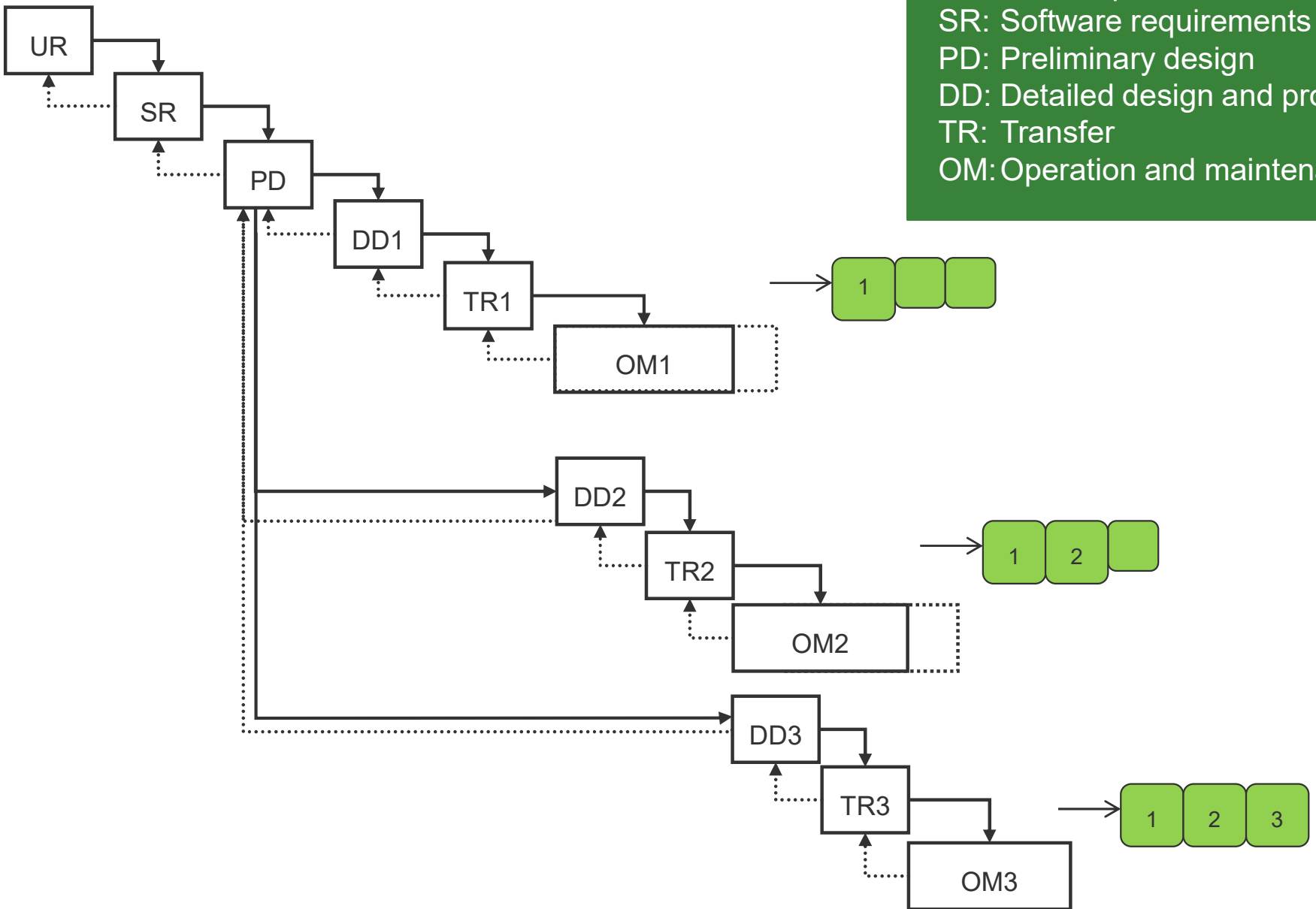
- Clear structure
- Objectives set at the beginning
- Information is well transmitted

- Changes are difficult
- Excludes client or final user
- Tests at the end

https://www.lucidchart.com/blog/es/pros-y-contras-de-la-metodologia-de-cascada#:~:text=Transmite%20bien%20la%20informaci%C3%B3n&text=Ya%20sea%20que%20est%C3%A9s%20traspasando,d%C3%ADa%20r%C3%A1pidamente%2C%20si%20es%20necesario.

Life-cycle models
Incremental

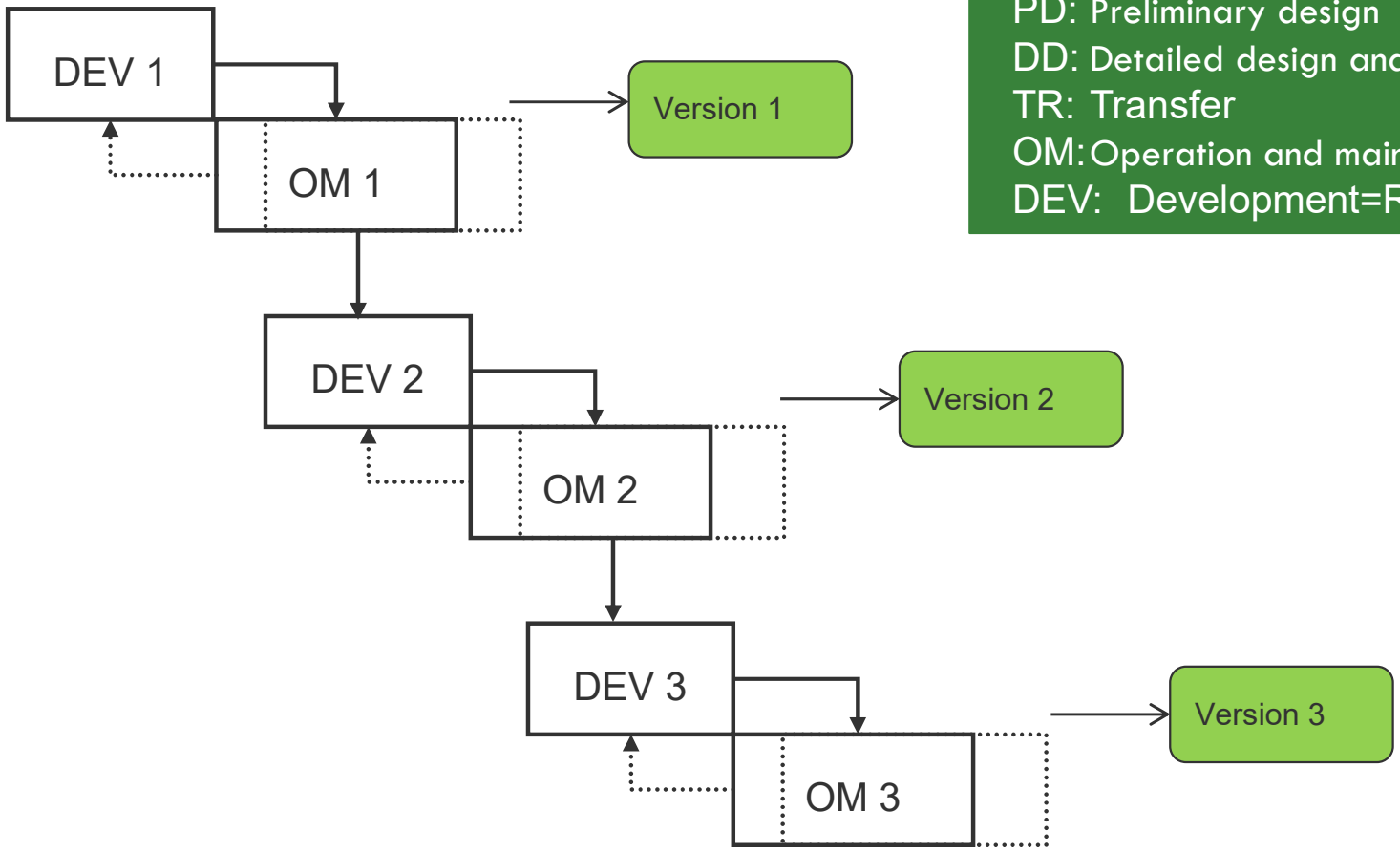
UR: User requirements
SR: Software requirements
PD: Preliminary design
DD: Detailed design and production
TR: Transfer
OM: Operation and maintenance



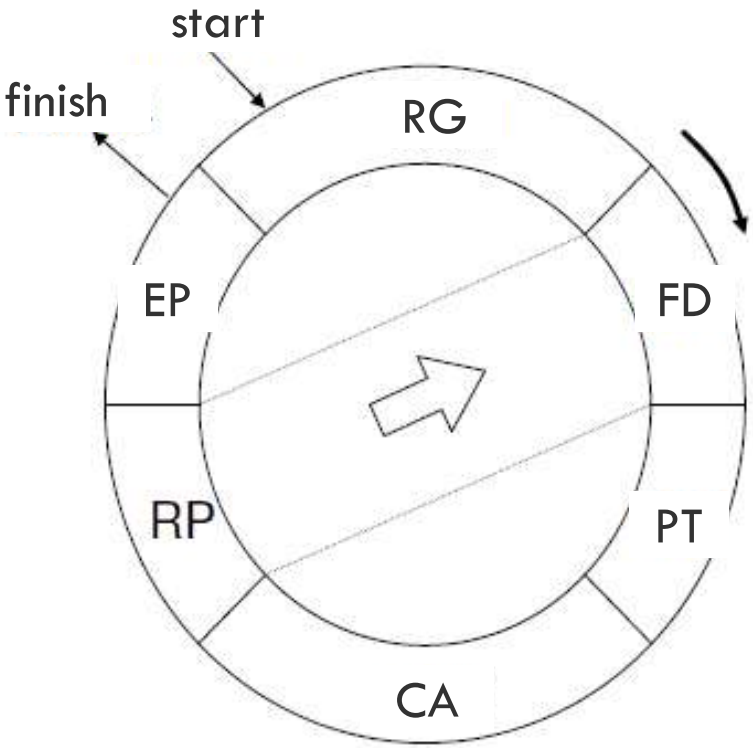
Life-cycle models

Evolutive

UR: User requirements
SR: Software requirements
PD: Preliminary design
DD: Detailed design and production
TR: Transfer
OM: Operation and maintenance
DEV: Development=RU+RS+DP+DD+TR



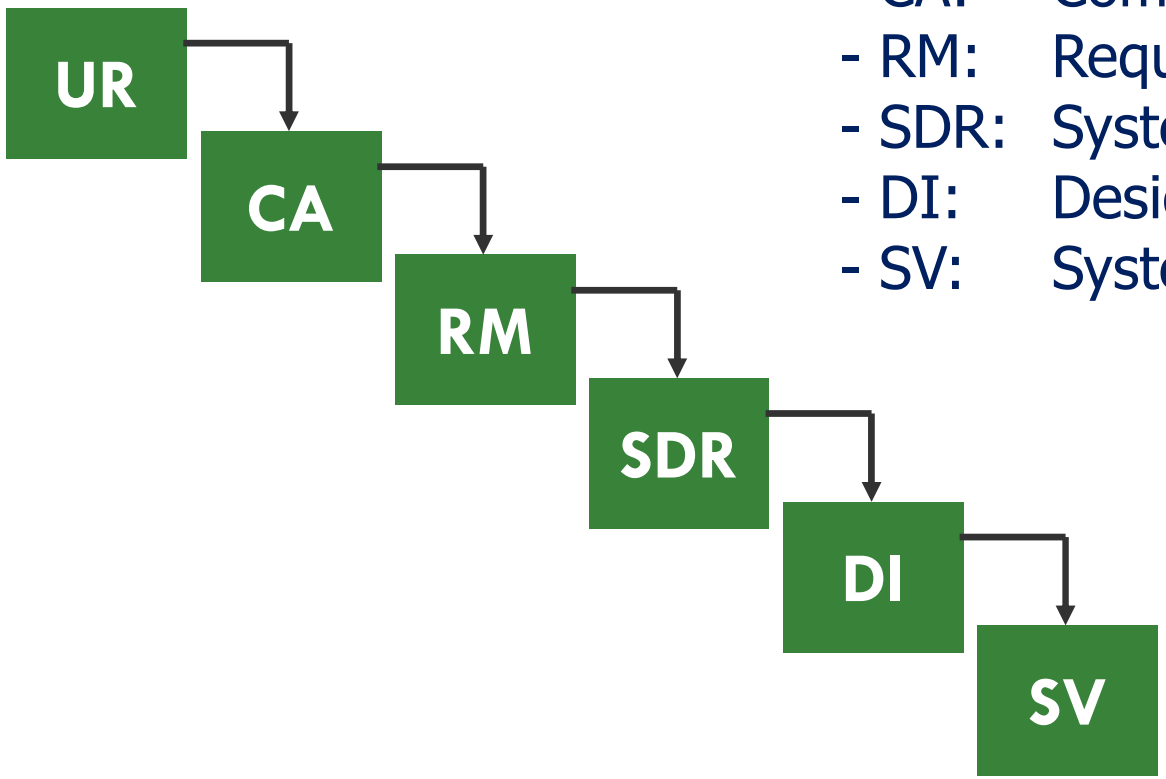
Life-cycle models
Prototyping



Stages

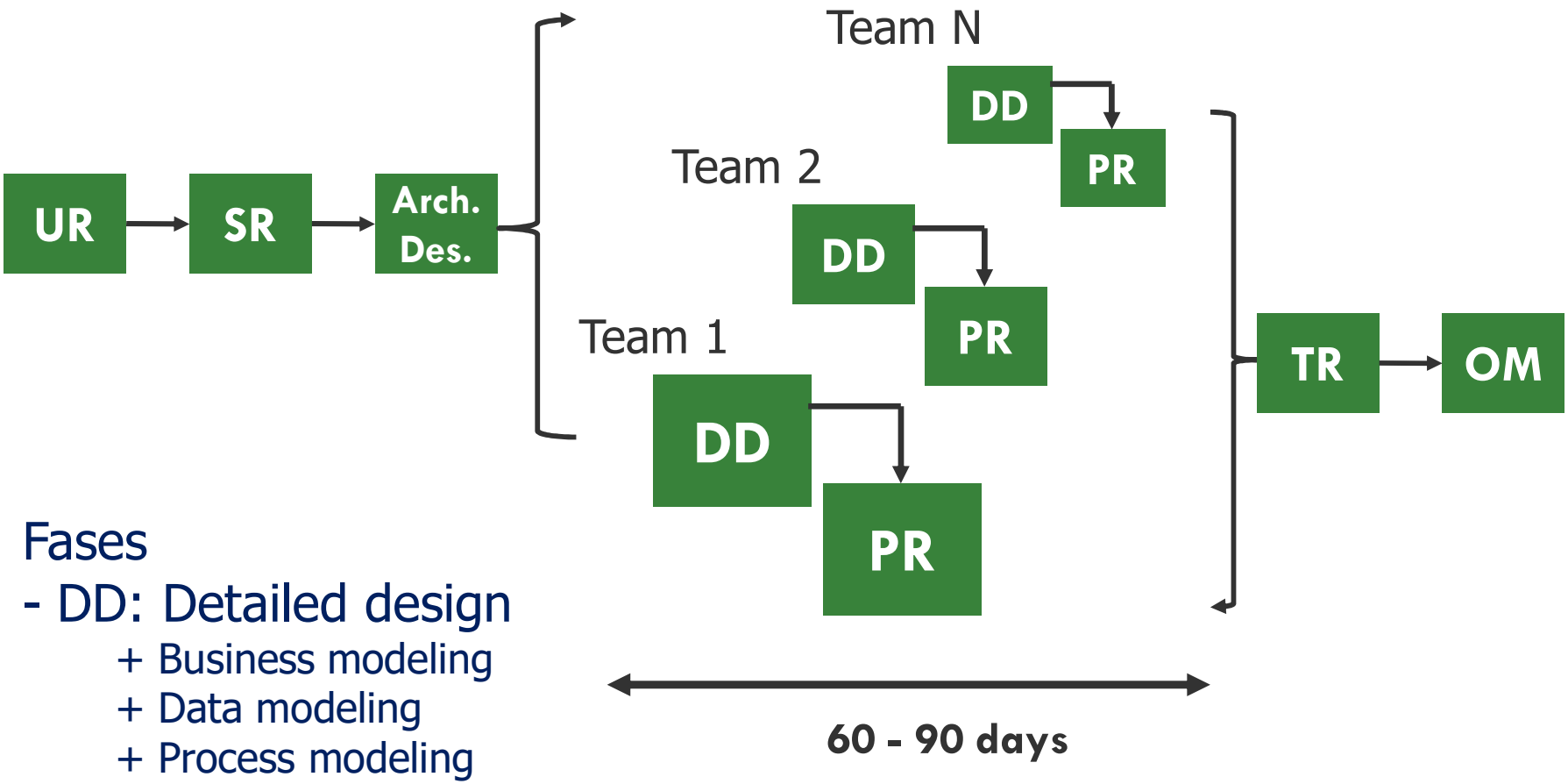
RG: requirements gathering
FD: fast design
PT: prototype
CA: assessment by the client
RP: refinement of prototype
EP: engineering product

Life-cycle models
Based on components



Fases
- UR: User requirements
- CA: Component analysis
- RM: Requirement modification
- SDR: System design with reuse
- DI: Design and integration
- SV: System validation

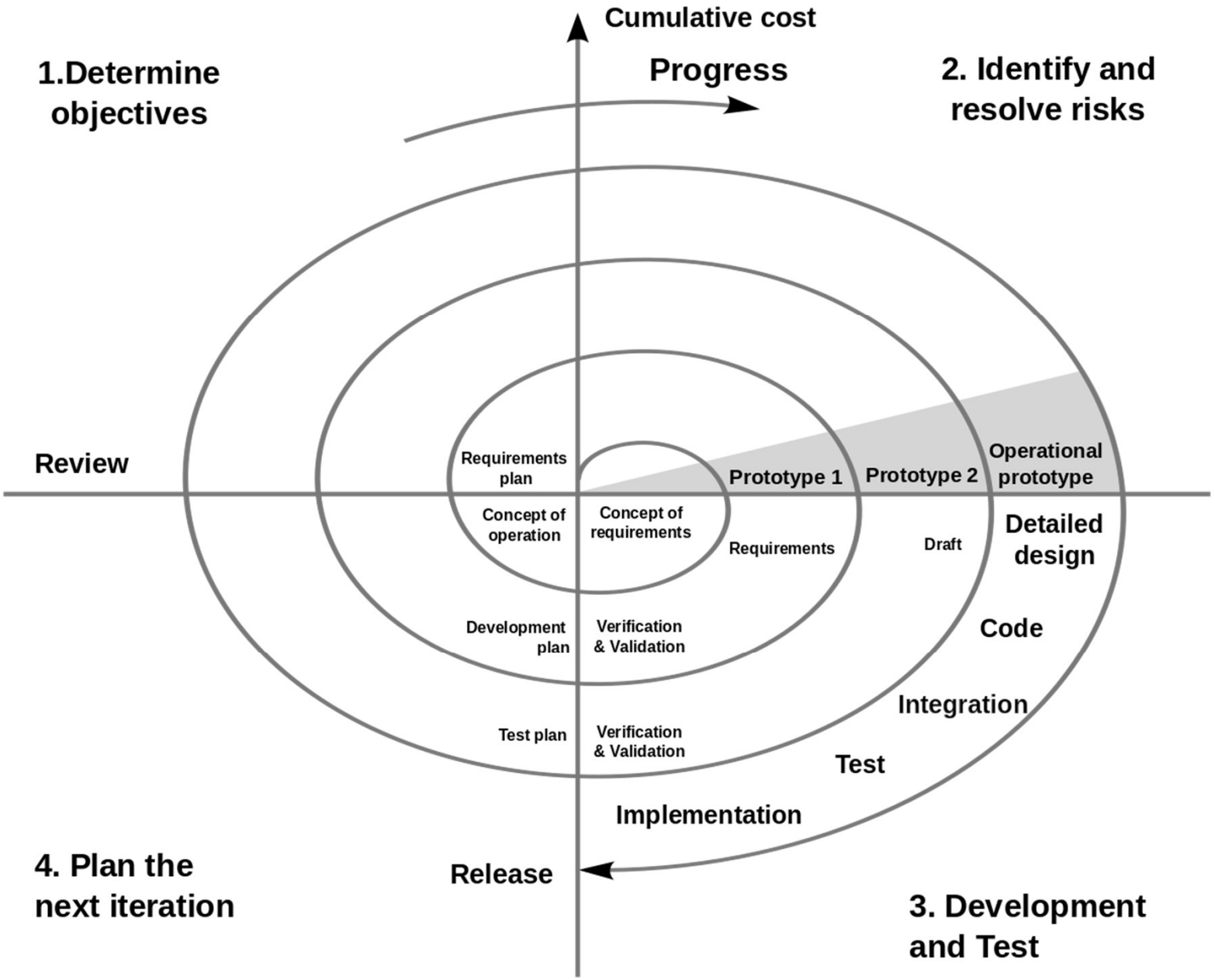
Life-cycle models
Rapid application development



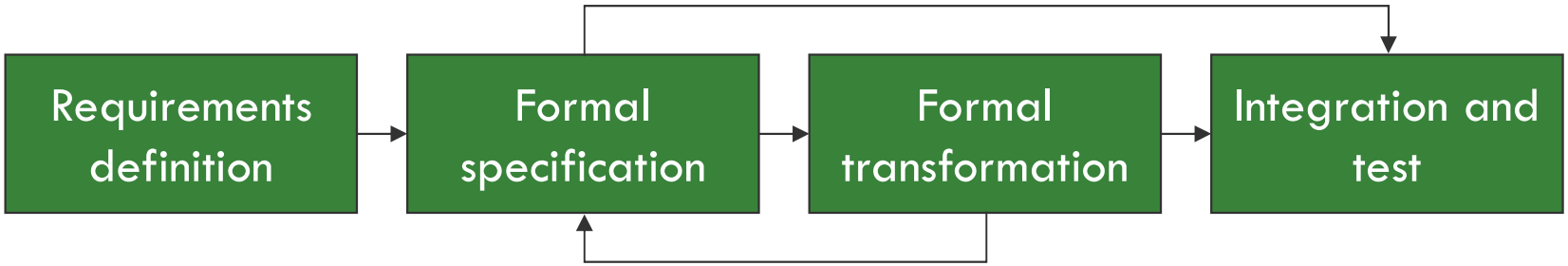
Fases

- DD: Detailed design
  - + Business modeling
  - + Data modeling
  - + Process modeling
- PR: production
  - + Reuse of components
  - + Coding
  - + Tests

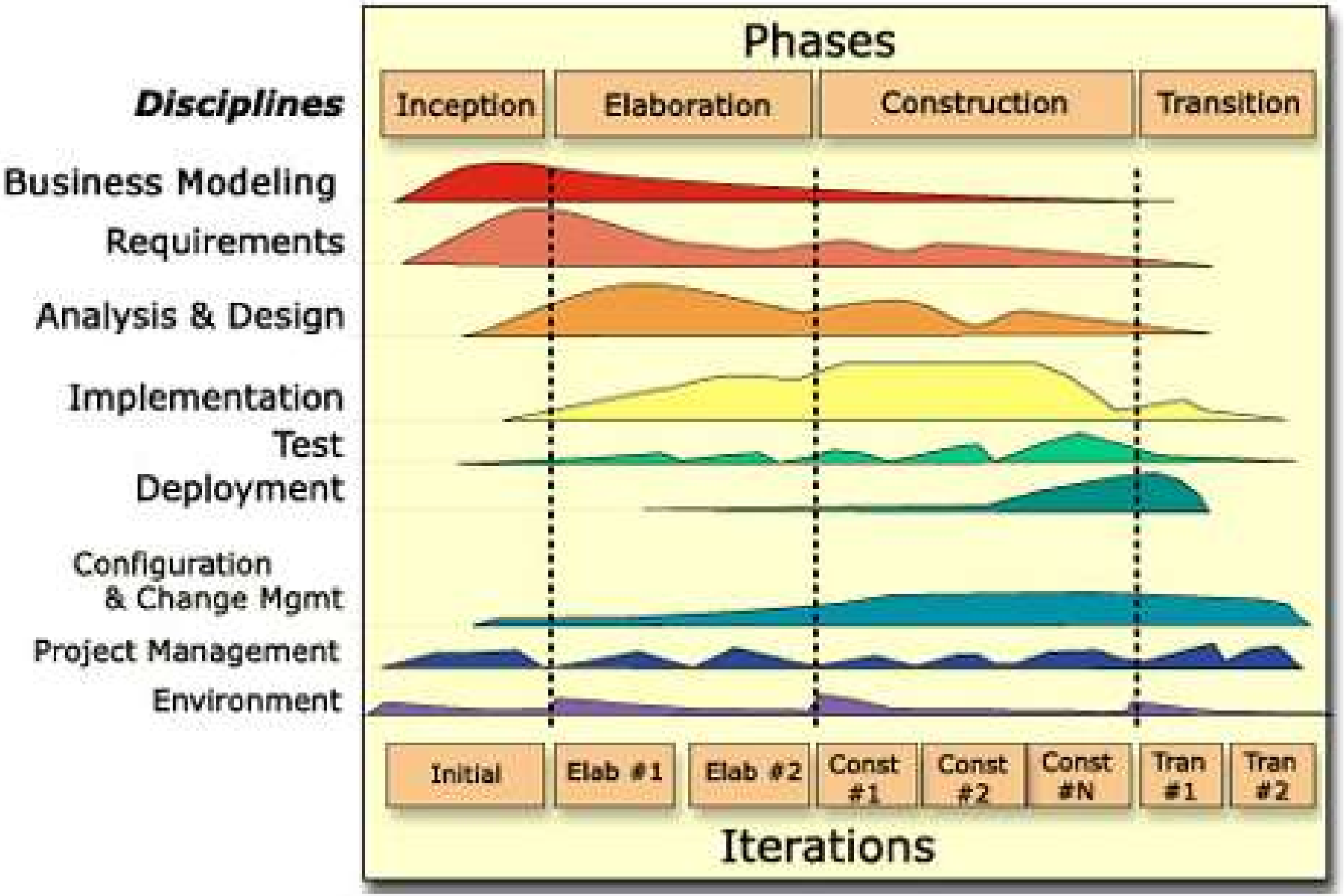
Life-cycle models
Spiral



Life-cycle models
Formal



Life-cycle models
RUP (rational unified process)



Life-cycle models

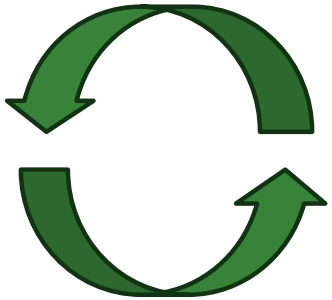
Agile manifest

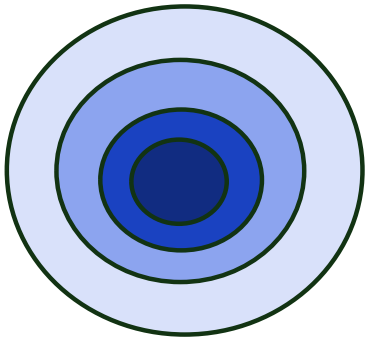
1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months).
4. Close, daily cooperation between business people and developers.
5. Projects are built around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location).
7. Working software is the primary measure of progress.
8. Sustainable development, able to maintain a constant pace.
9. Continuous attention to technical excellence and good design.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. Best architectures, requirements, and designs emerge from self-organizing teams.
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly.

Iterative incremental model
Iterative + Incremental

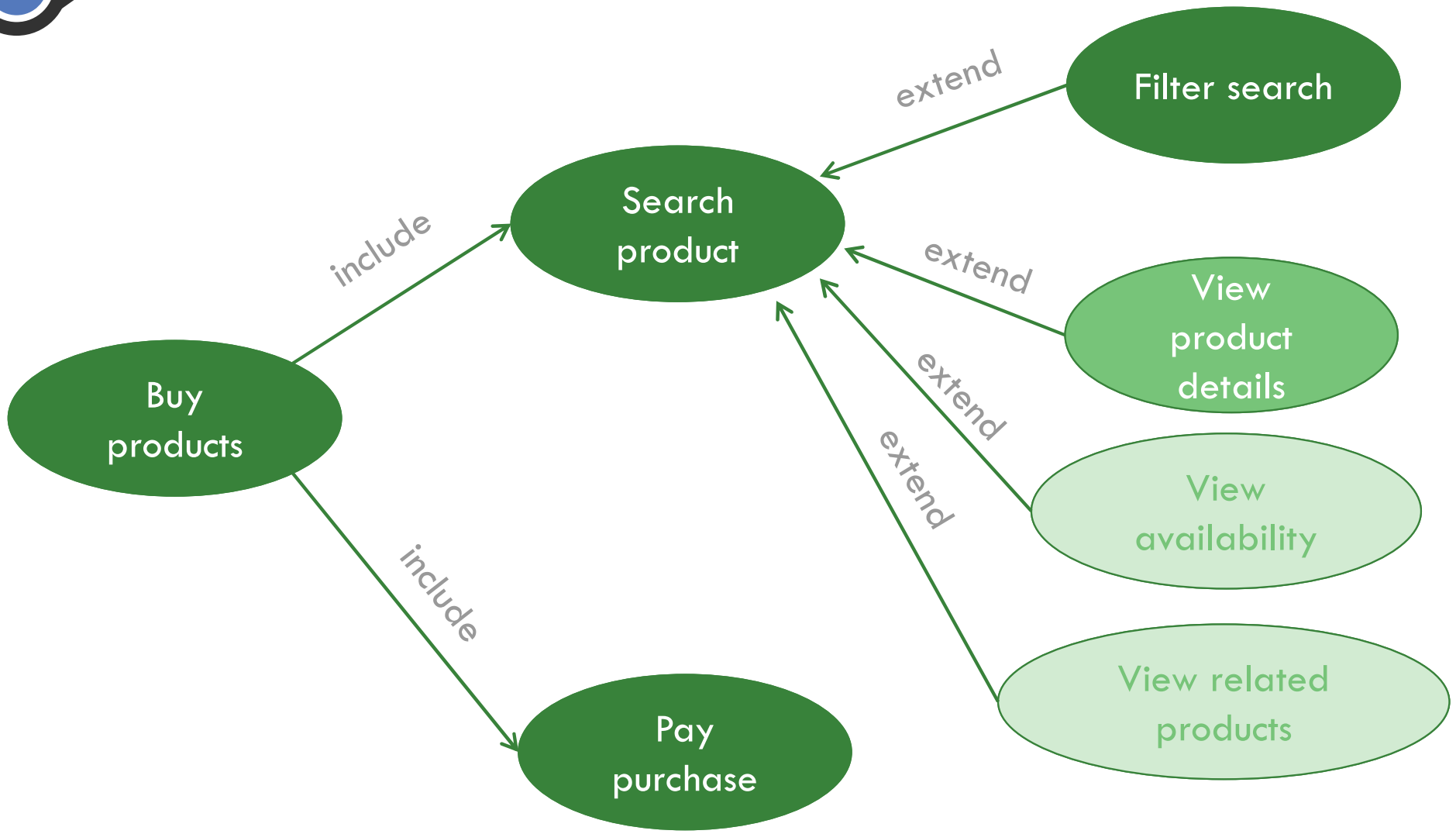
- **Iterative**: In each iteration, a similar work process is repited (iterations are similar to miniprojects).

- **Incremental**: New options are added in each iteration.





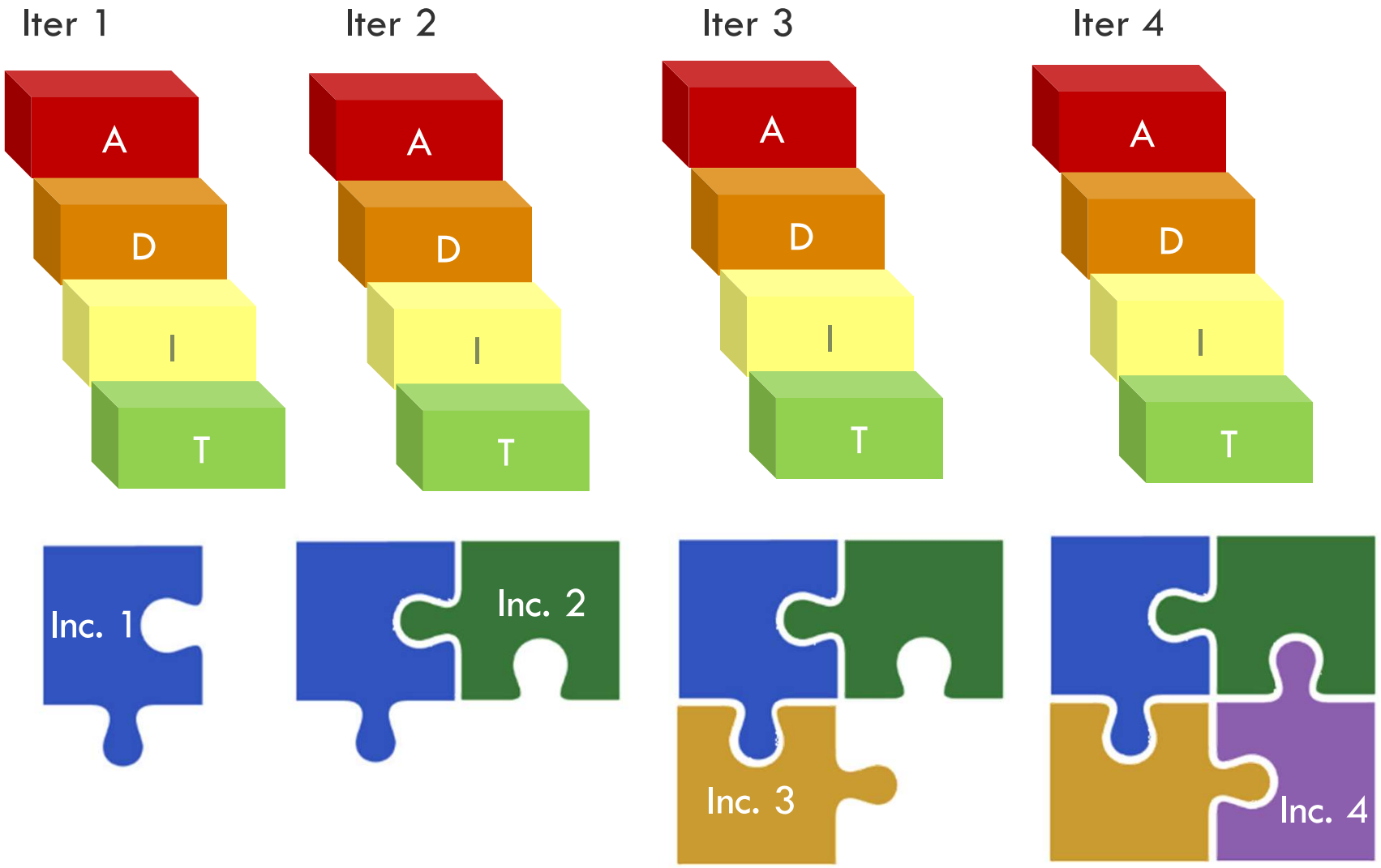
Iterative incremental model
Prioritizing requirements



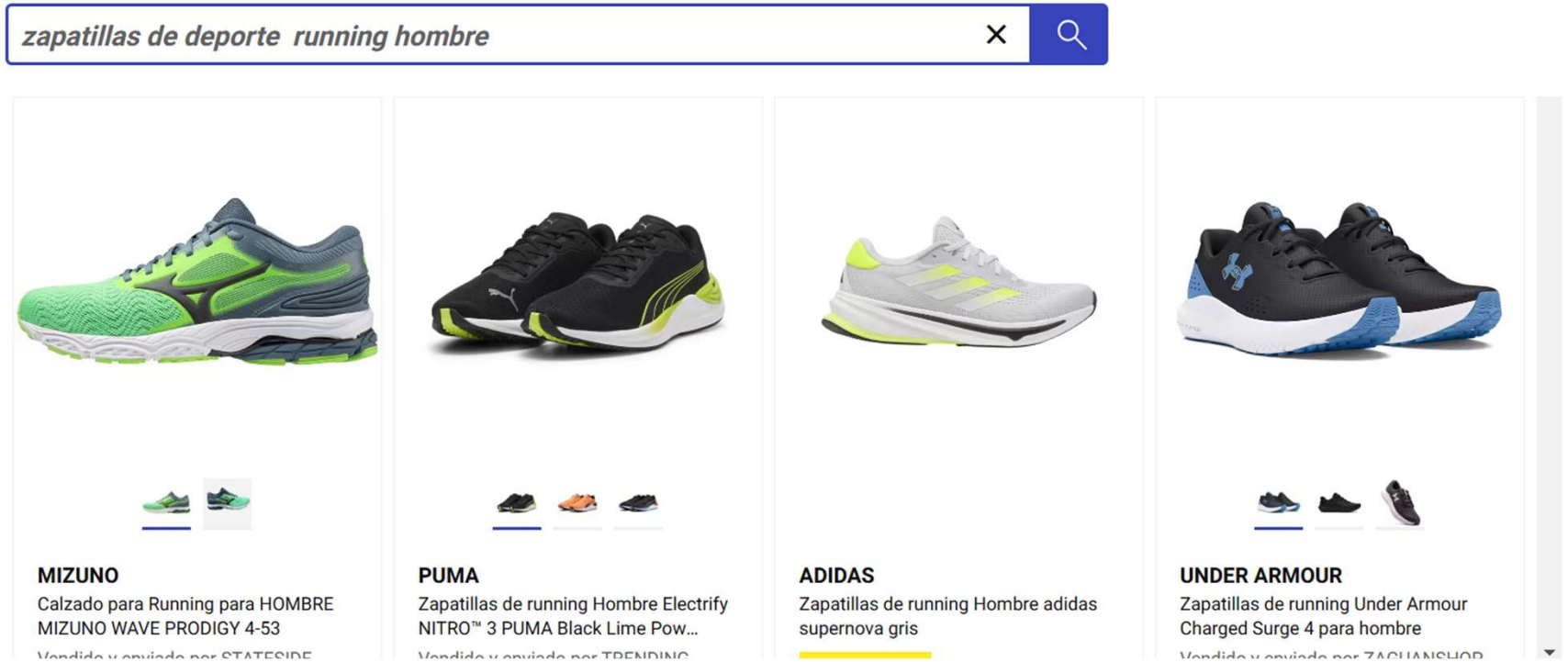
Iterative incremental model
Prioritizing requirements

- *A user can search for products in the catalog indicating key words.*
- *A user can add products to the shopping cart, indicating size and amount.*
- *A user can filter the results of a search by category, sport or gender.*
- *A user can see the details of a product, including pictures, uses, materials and technical specifications.*
- *A user can see the availability of a product in the shops near a given address.*
- *A user can pay for the selected products with a credit card.*
- *The system shows those products related to a selected one, including similar and complementary products.*

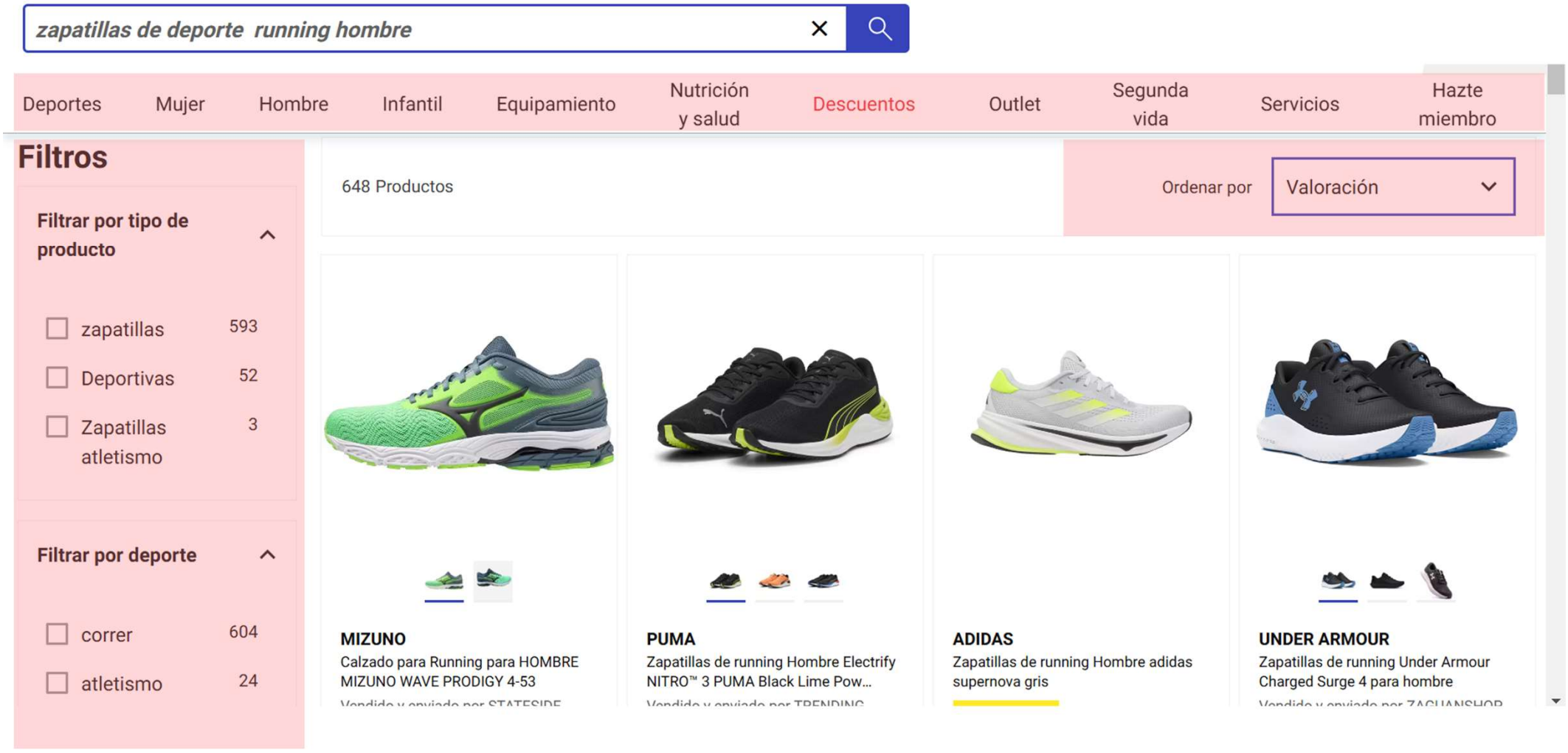
Iterative incremental model
Iterations and increments



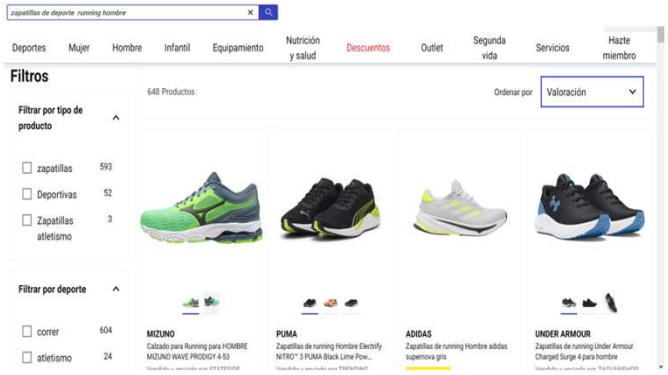
Example: shop online
Search



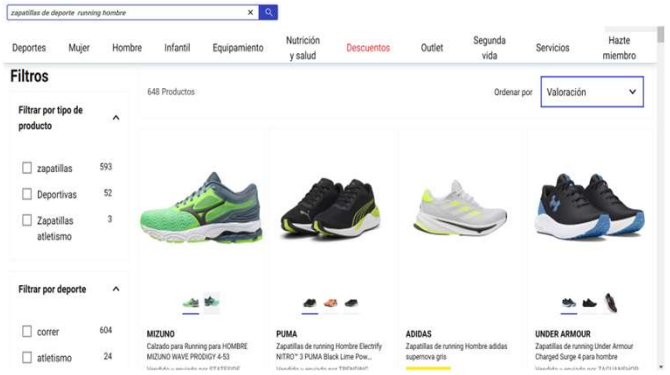
Example: shop online
Filter and sort



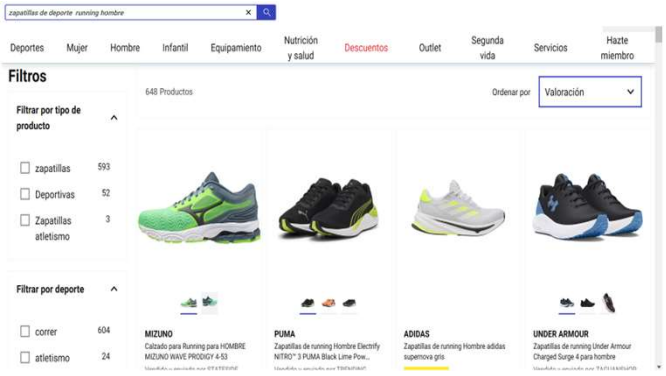
Example: shop online
Purchase, payment and delivery

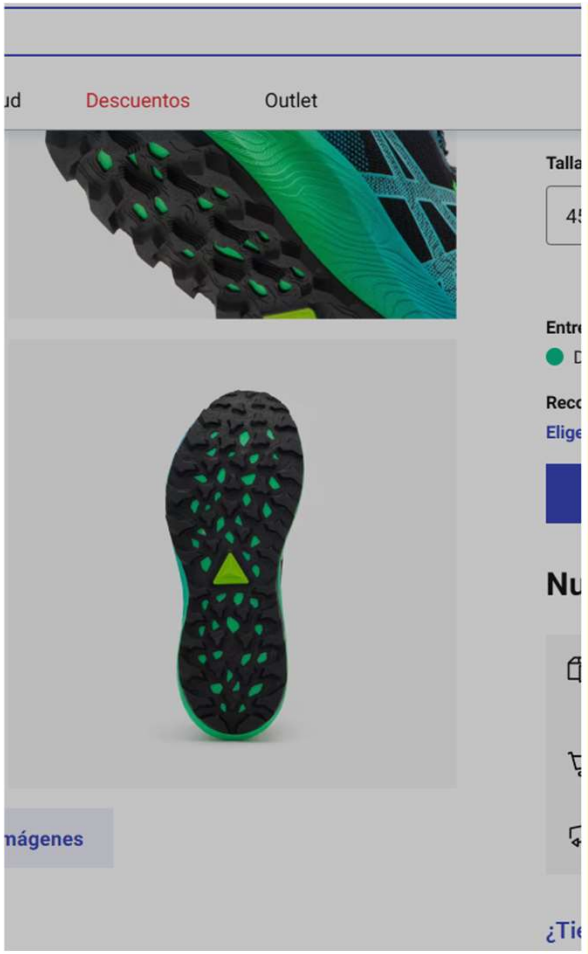


Example: shop online
Product details

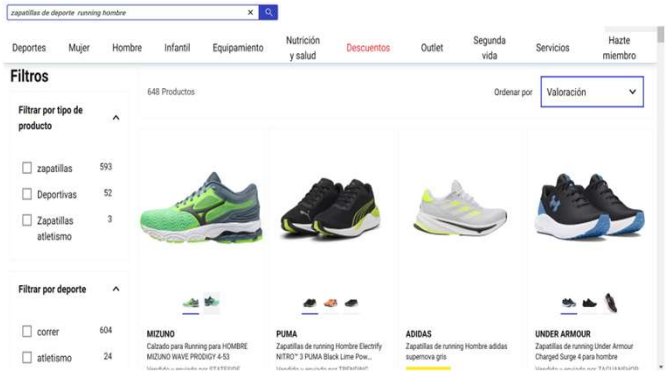


Example: shop online
Availability

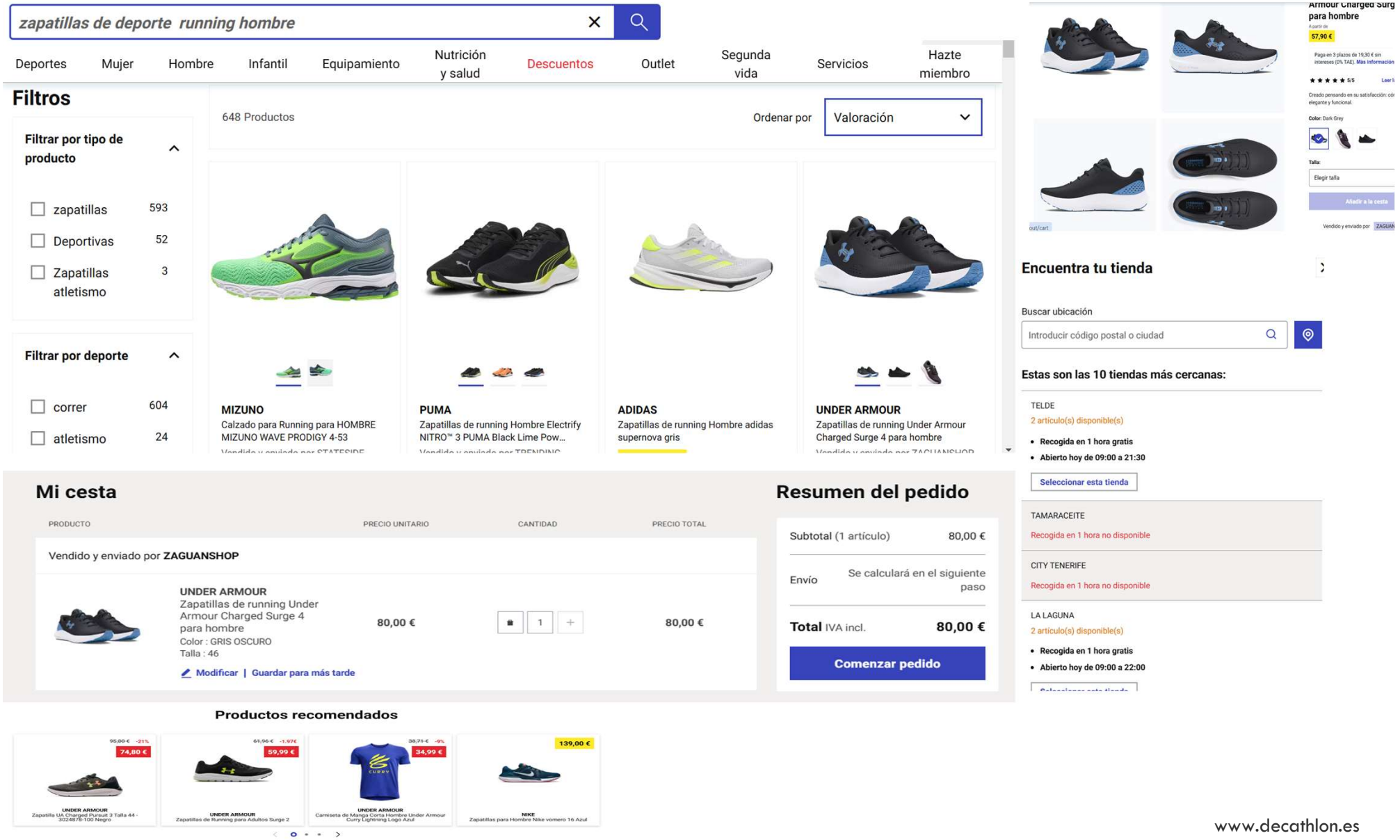




Example: shop online
Recommendations



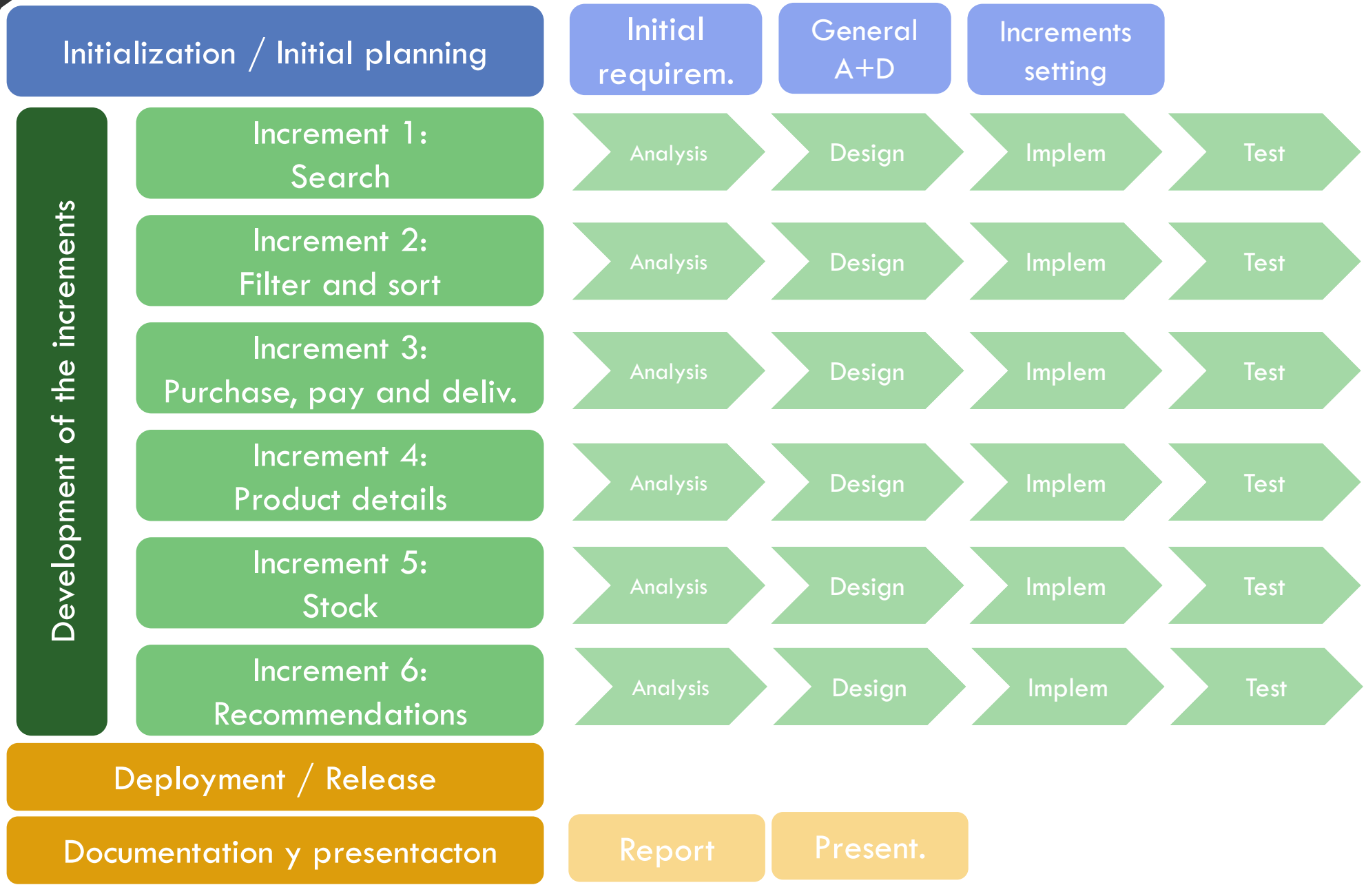
Example: shop online
Whole project



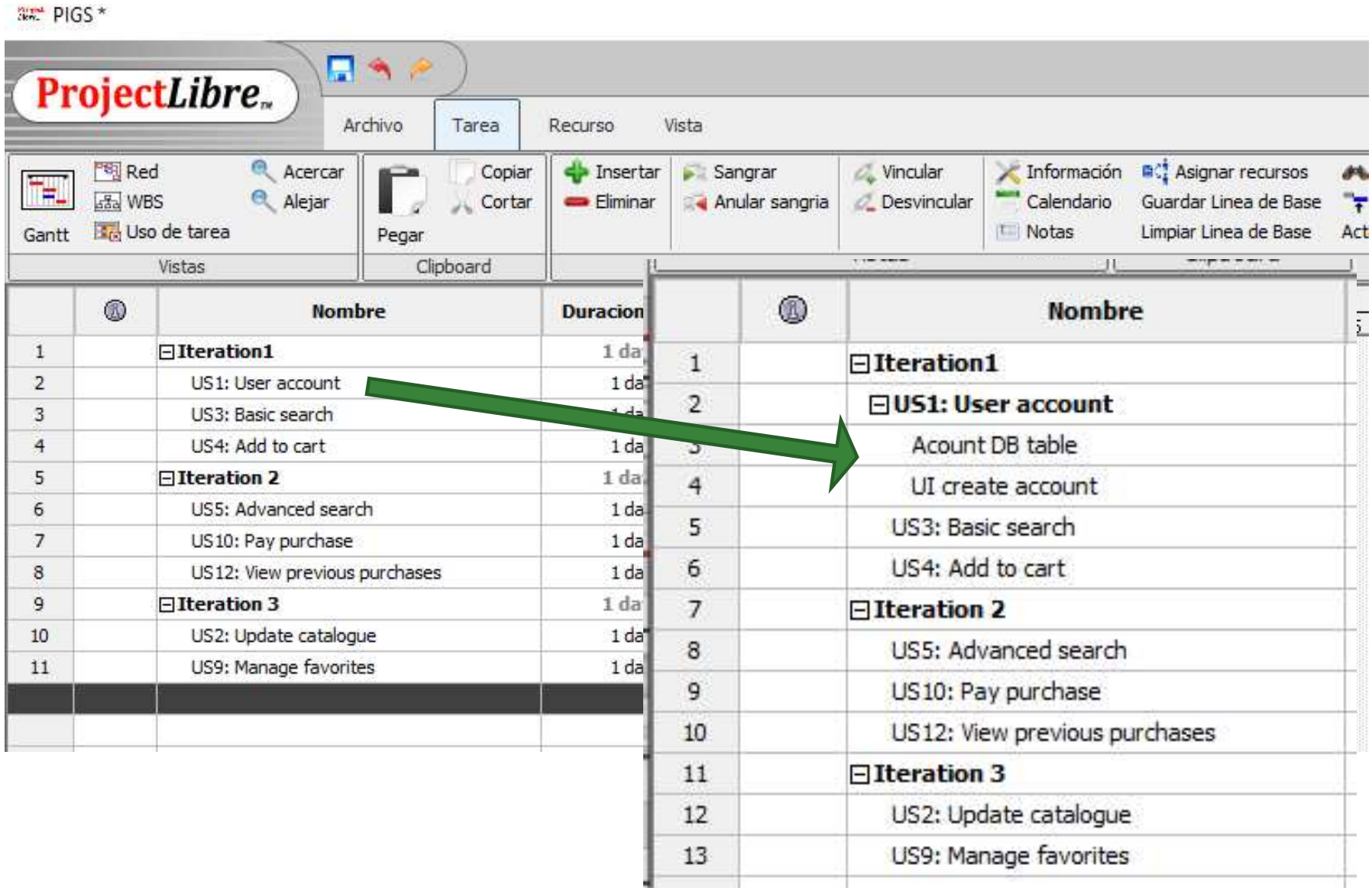
www.decathlon.es

Example: online shopping

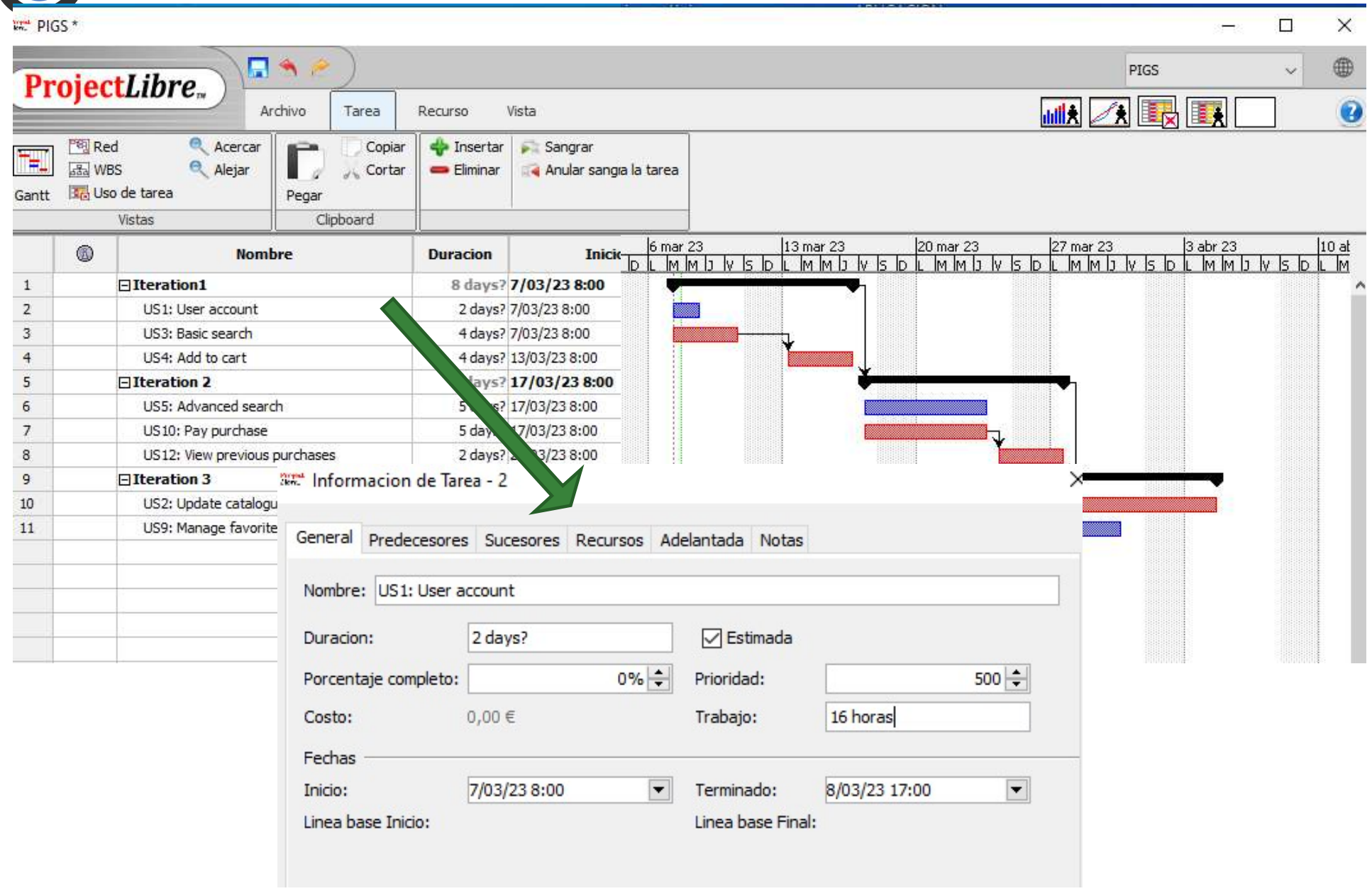
Phases



Example


PIGS *
ProjectLibre
Archivo
Tarea
Recurso
Vista
Red
WBS
Uso de tarea
Gantt
Acercar
Alejar
Vistas
Pegar
Copiar
Cortar
Clipboard
Insertar
Eliminar
Sangrar
Anular sangria
Vincular
Desvincular
Información
Calendario
Notas
Asignar recursos
Guardar Linea de Base
Limpiar Linea de Base
Nombre
Duracion
1 Iteration1
2 US1: User account
3 US3: Basic search
4 US4: Add to cart
5 Iteration 2
6 US5: Advanced search
7 US10: Pay purchase
8 US12: View previous purchases
9 Iteration 3
10 US2: Update catalogue
11 US9: Manage favorites
Nombre
1 Iteration1
2 US1: User account
Acount DB table
4 UI create account
5 US3: Basic search
6 US4: Add to cart
7 Iteration 2
8 US5: Advanced search
9 US10: Pay purchase
10 US12: View previous purchases
11 Iteration 3
12 US2: Update catalogue
13 US9: Manage favorites

Example


PIGS *
ProjectLibre
PIGS
Archivo
Tarea
Recurso
Vista
Red
WBS
Uso de tarea
Gantt
Acercar
Alejar
Vistas
Pegar
Copiar
Cortar
Clipboard
Insertar
Eliminar
Sangrar
Anular sangra la tarea
Nombre
Duracion
Inicio
1 Iteration1 8 days? 7/03/23 8:00
2 US1: User account 2 days? 7/03/23 8:00
3 US3: Basic search 4 days? 7/03/23 8:00
4 US4: Add to cart 4 days? 13/03/23 8:00
5 Iteration 2 17/03/23 8:00
6 US5: Advanced search 17/03/23 8:00
7 US10: Pay purchase 17/03/23 8:00
8 US12: View previous purchases
9 Iteration 3
10 US2: Update catalogu
11 US9: Manage favorite
6 mar 23
13 mar 23
20 mar 23
27 mar 23
3 abr 23
Informacion de Tarea - 2
General
Predecesores
Sucesores
Recursos
Adelantada
Notas
Nombre: US1: User account
Duracion: 2 days?
Estimada
Porcentaje completo: 0%
Prioridad: 500
Costo: 0,00 €
Trabajo: 16 horas
Fechas
Inicio: 7/03/23 8:00
Terminado: 8/03/23 17:00
Linea base Inicio:
Linea base Final:

Example

PIGS *

ProjectLibre

Archivo | Tarea | Recurso | Vista

Recursos | RBS | Uso de recurso | Acercar | Alejar — Vistas

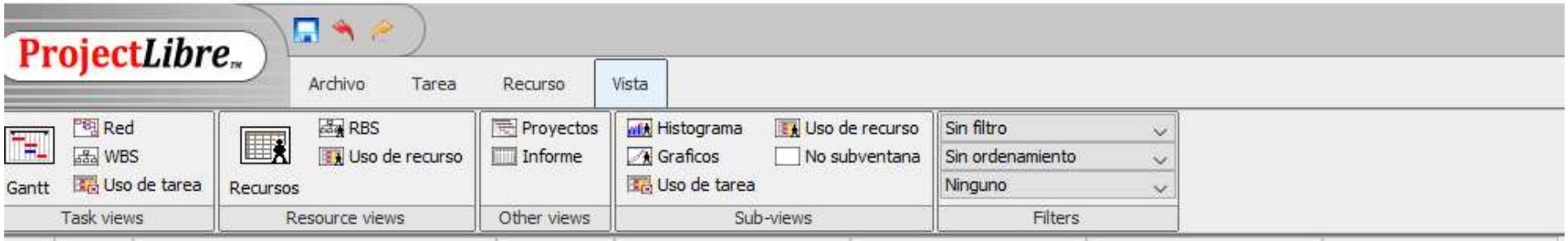
Pegar | Copiar | Cortar — Clipboard

Insertar | Eliminar | Sangrar | Anular sangria | Información | Calendario | Notas | Buscar — Recurso

		Nombre	RBS	Tipo	Direccion de correo elec...	Etiqueta material	Iniciales
1		Pepe		Trabajo			P
2		Ana		Trabajo			A
3		Luis		Trabajo			L
4		Paula		Trabajo			P

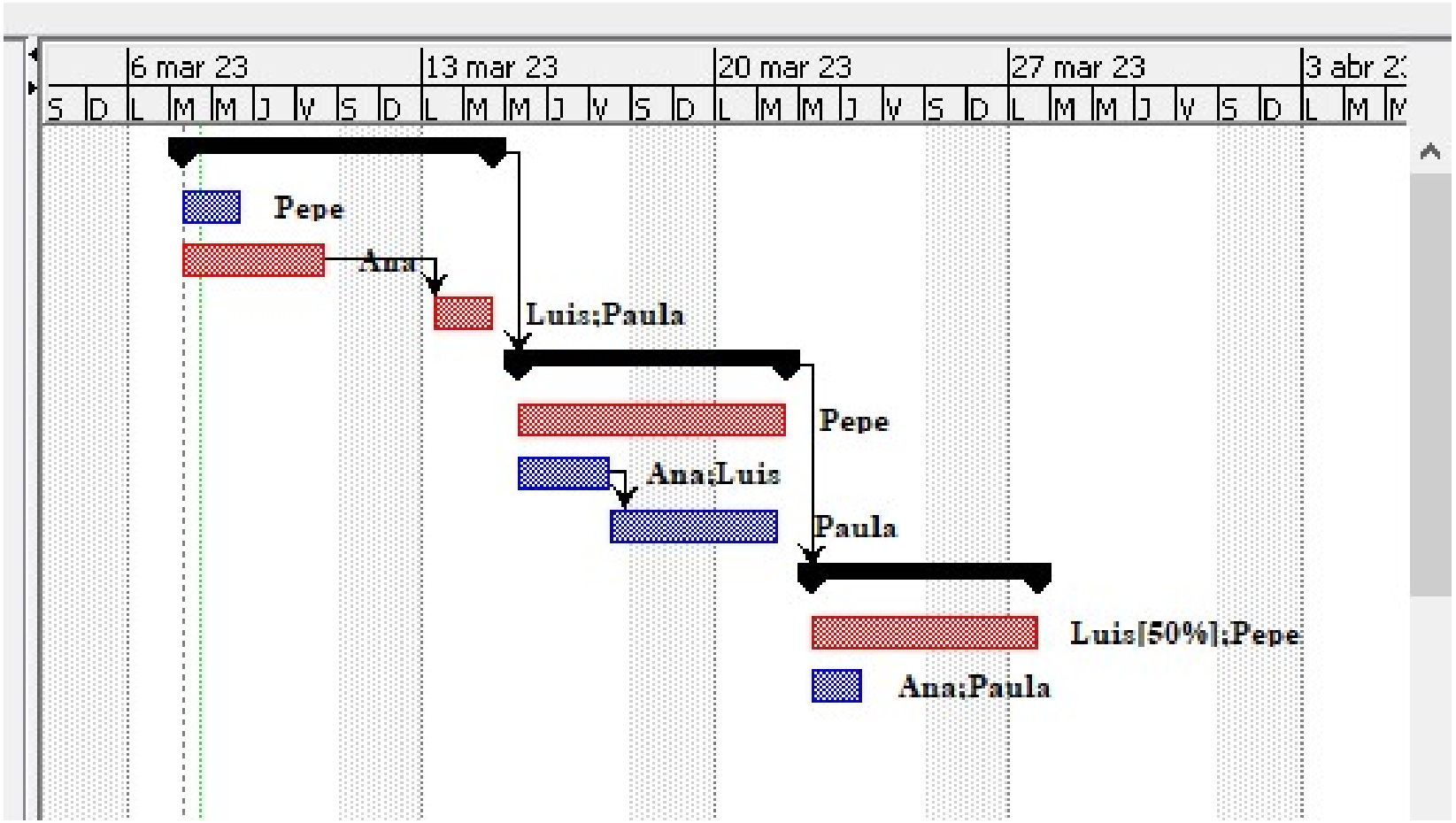
Example

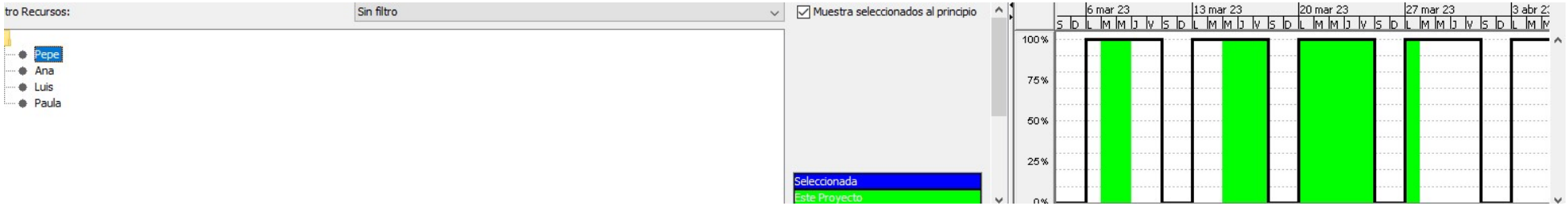
PIGS *



	Nombre	Duracion	Inicio	Terminado	Predecesores	Nombres del Recurso
1	**Iteration1**	6 days?	**7/03/23 8:00**	**14/03/23 17:00**		
2	US1: User account	2 days?	7/03/23 8:00	8/03/23 17:00		Pepe
3	US3: Basic search	4 days?	7/03/23 8:00	10/03/23 17:00		Ana
4	US4: Add to cart	2 days?	13/03/23 8:00	14/03/23 17:00	3	Luis;Paula
5	**Iteration 2**	5 days?	**15/03/23 8:00**	**21/03/23 17:00**	**1**	
6	US5: Advanced search	5 days?	15/03/23 8:00	21/03/23 17:00		Pepe
7	US10: Pay purchase	2,5 days?	15/03/23 8:00	17/03/23 13:00		Ana;Luis
8	US12: View previous purchases	2 days?	17/03/23 13:00	21/03/23 13:00	7	Paula
9	**Iteration 3**	4 days?	**22/03/23 8:00**	**27/03/23 17:00**	**5**	
10	US2: Update catalogue	4 days?	22/03/23 8:00	27/03/23 17:00		Luis[50%];Pepe
11	US9: Manage favorites	1,5 days?	22/03/23 8:00	23/03/23 13:00		Ana;Paula

Example


6 mar 23
13 mar 23
20 mar 23
27 mar 23
Pepe
Ana
Luis;Paula
Pepe
Ana;Luis
Paula
Luis[50%];Pepe
Ana;Paula


tro Recursos:
Sin filtro
Muestra seleccionados al principio
Pepe
Ana
Luis
Paula
Seleccionada
Este Proyecto
6 mar 23
13 mar 23
20 mar 23
27 mar 23
100%
75%
50%
25%

The Release Plan

Steps

- Select an iteration length
- Estimate the velocity
- Prioritize the stories
- Allocate stories to the one or more iterations

The Release Plan
Prioritizing user stories

- Sort the stories so that they maximize the value delivered to the organization.
- Stories are prioritized by the customer but with input from the developers.

The Release Plan
Prioritizing user stories

Dimensions along which we can sort stories:

- the risk that the story cannot be completed as desired (for example, with desired performance characteristics or with a novel algorithm)
- the impact the story will have on other stories if deferred (we don't want to wait until the last iteration to learn that the application is to be three-tiered and multi-threaded)

The Release Plan
Prioritizing user stories

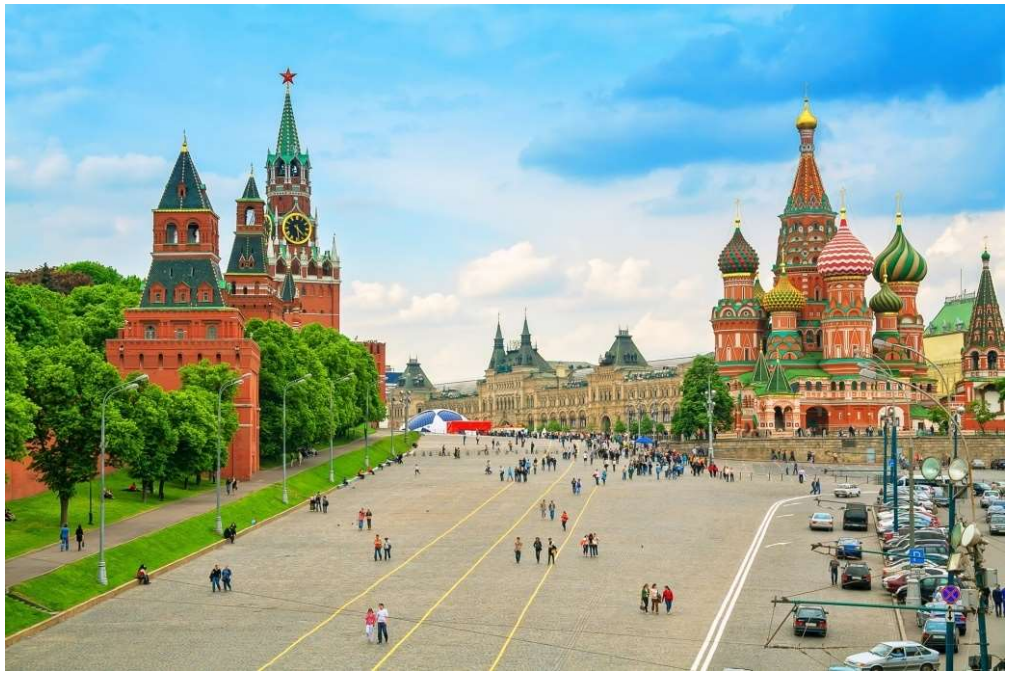
Additional factors:

- the desirability of the story to a broad base of users or customers
- the desirability of the story to a small number of important users or customers
- the cohesiveness of the story in relation to other stories (for example, a “zoom out” story may not be high priority on its own but may be treated as such because it is complementary to “zoom in,” which is high priority)

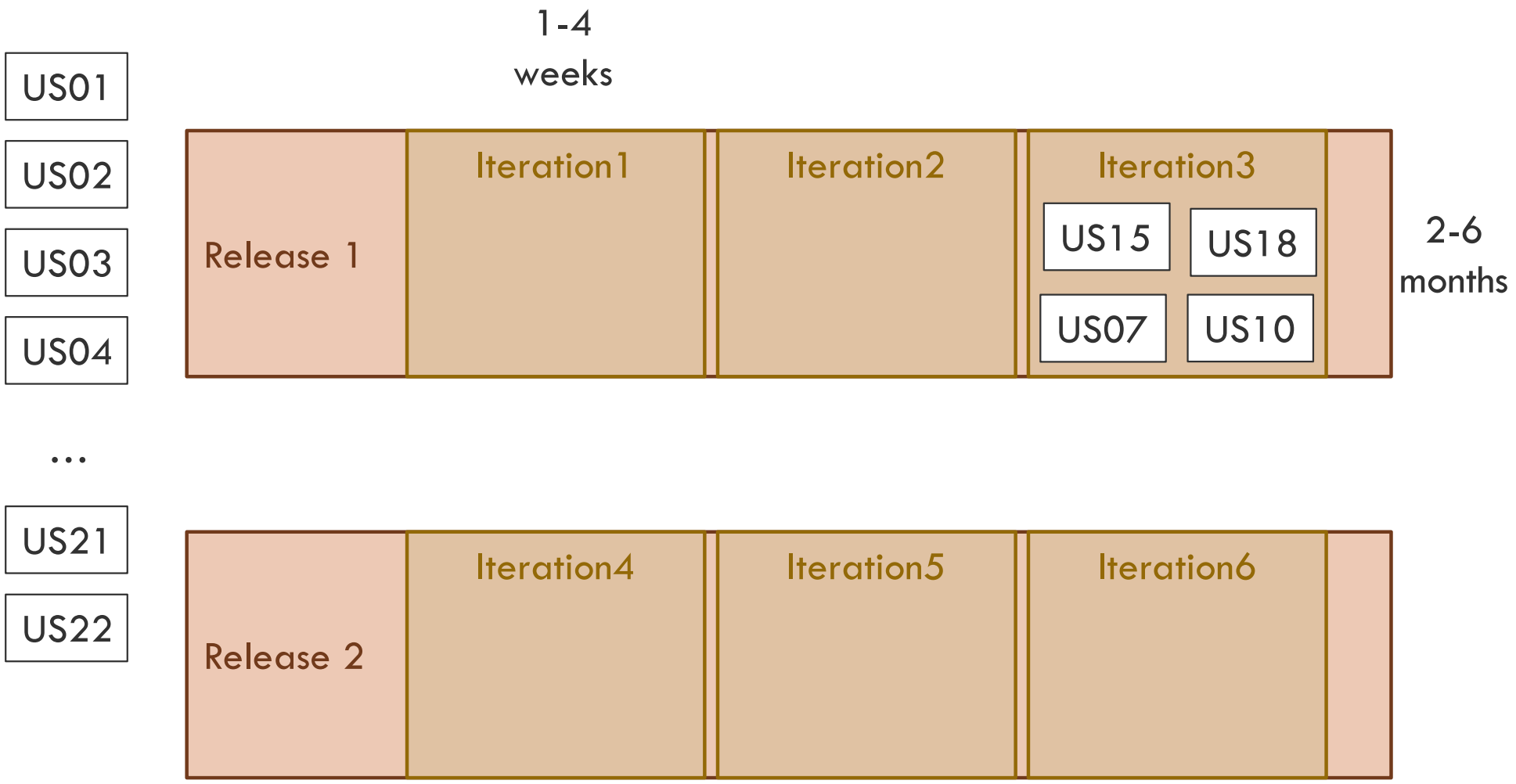
The Release Plan
Prioritizing user stories

MoSCoW rules.

- Must have
- Should have
- Could have
- Won’t have this time



The Release Plan
Releases and iterations



The Release Plan
Releases and iterations

- Releases and iterations are planned by placing stories into iterations.
- Velocity is the amount of work the developers can complete in an iteration.
- The sum of the estimates of the stories placed in an iteration cannot exceed the velocity the developers forecast for that iteration.
- If a story won't fit in an iteration, you can split the story into two or more smaller stories.
- Acceptance tests validate that a story has been developed with the functionality the customer team had in mind when they wrote the story.

The Release Plan

Story	Story Points
Story A	3
Story B	5
Story C	5
Story D	3
Story E	1
Story F	8
Story G	5
Story H	5
Story I	5
Story J	2

The team estimates a velocity of thirteen story points per iteration (assume decreasing priority)

Iteration	Stories	Story Points
Iteration 1	A, B, C	13
Iteration 2	D, E, F	12
Iteration 3	G, H, J	12
Iteration 4	I	5

What if we can split I into Y(3) and Z(2)?

Iteration	Stories	Story Points
Iteration 1	A, B, C	13
Iteration 2	D, E, F	12
Iteration 3	G, H, Y	13
Iteration 4	J, Z	4

Planning a release

- What are three ways of estimating a team's initial velocity?

  - Historical values
  - Take a guess
  - Run an initial iteration and use the velocity of that iteration

Planning an iteration

- Iteration planning takes release planning one step further but only for the iteration being started.
- To plan the iteration, the team discusses each story and disaggregates it into its constituent tasks.
- There is no mandatory size range for tasks (for example, three to five hours). Instead, stories are disaggregated into tasks to facilitate estimation or to encourage more than one developer to work on various parts of the story.
- Developers accept responsibility for the tasks.
- Developers assess whether they have over-committed themselves by estimating each task they have accepted.

Planning an iteration
Decomposing a story into tasks

"A user can view detailed information about a hotel."

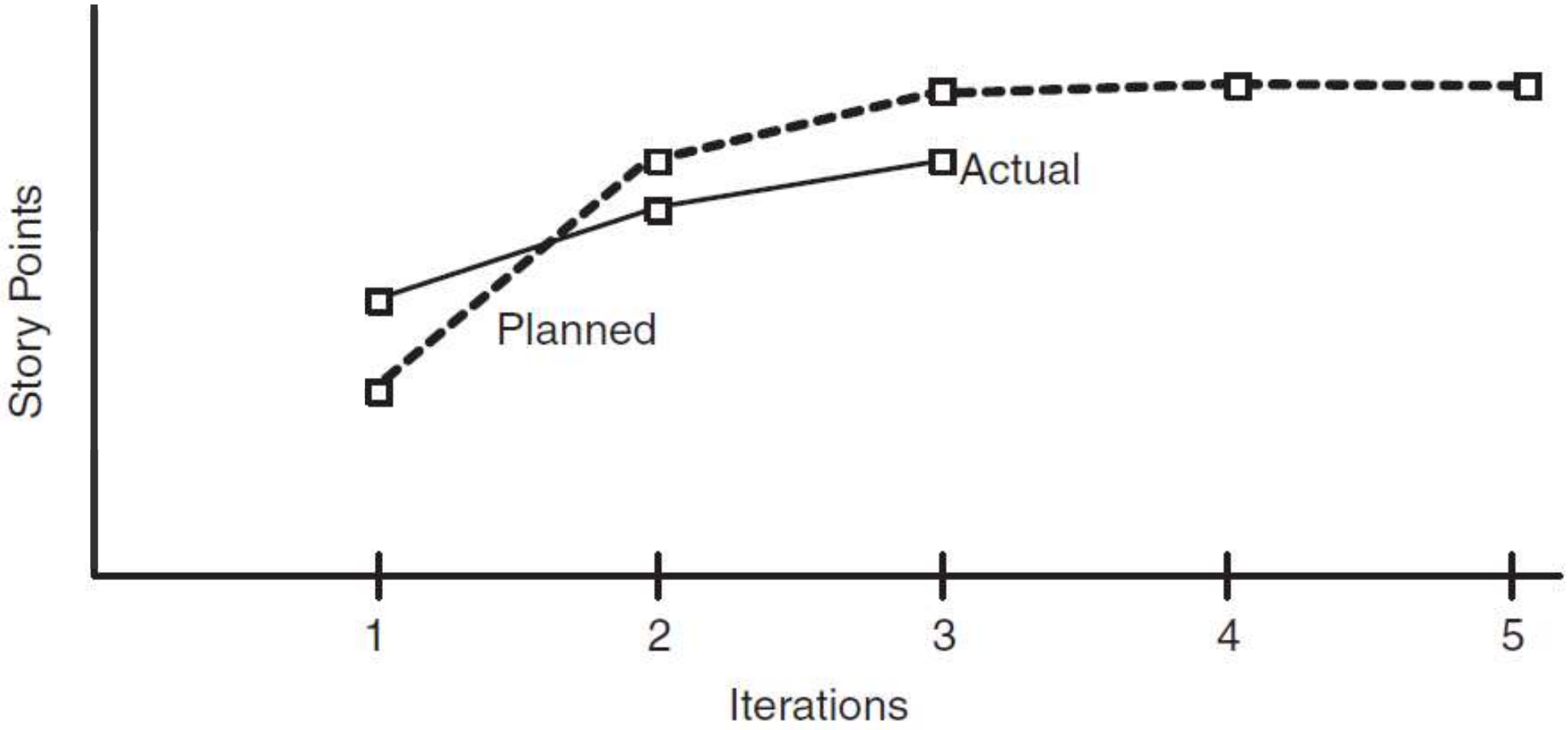
- Design the look of these web pages.
- Code the HTML to display hotel and room photos.
- Code the HTML to display a map showing where the hotel is.
- Code the HTML to display a list of hotel amenities and services.
- Figure out how we're generating maps.
- Write SQL to retrieve information from the database.
- Document new functionality in help system and user's guide

Planning an iteration
Decomposing a story into tasks

"A user can search for a hotel on various fields"

Task	Who	Estimate
Code basic search screen	Susan	6
Code advanced search screen	Susan	8
Code results screen	Jay	6
Write and tune SQL to query the database for basic searches	Susan	4
Write and tune SQL to query the database for advanced searches	Susan	8
Document new functionality in help system and user's guide	Shannon	2

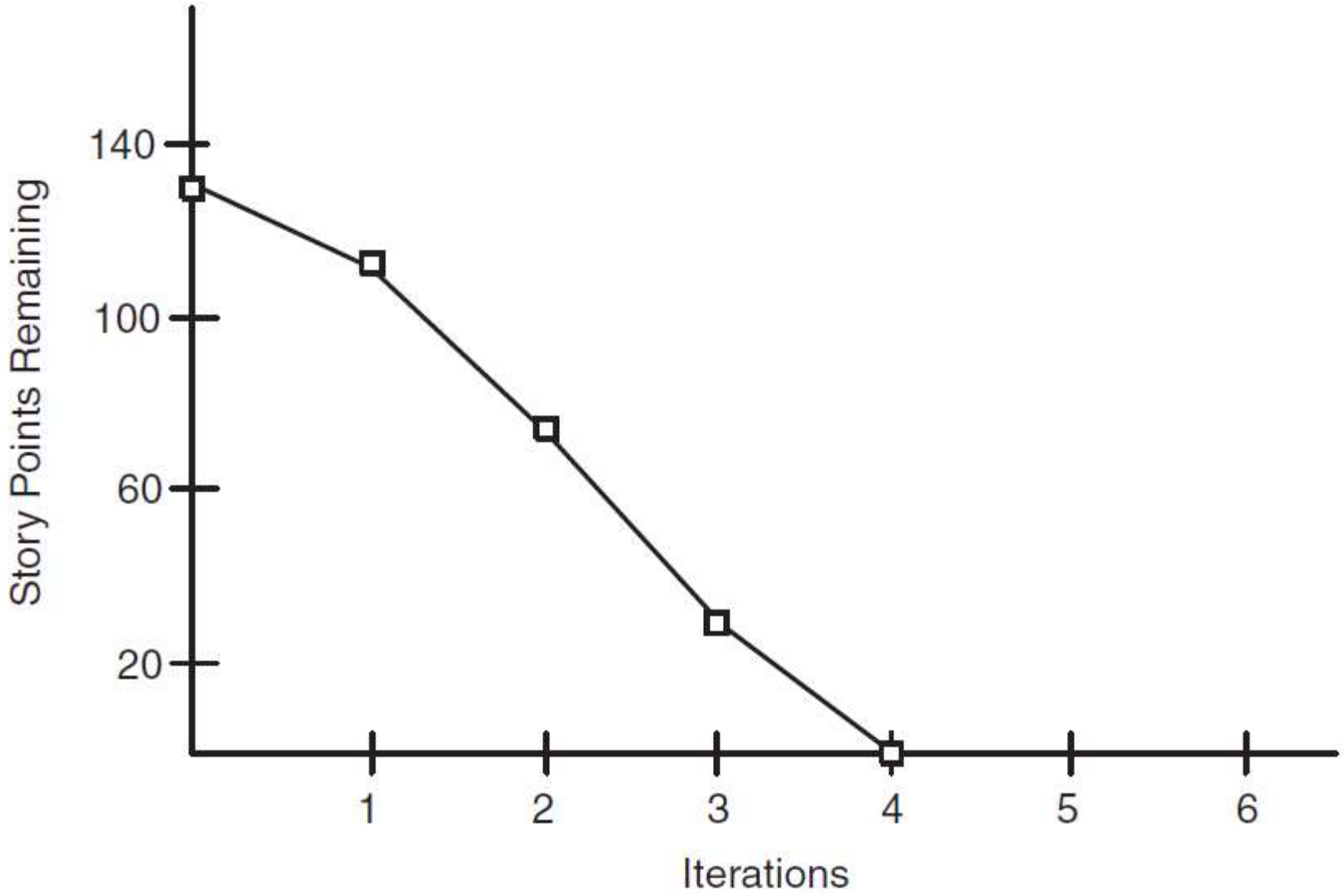
Monitoring
Expected and actual velocities



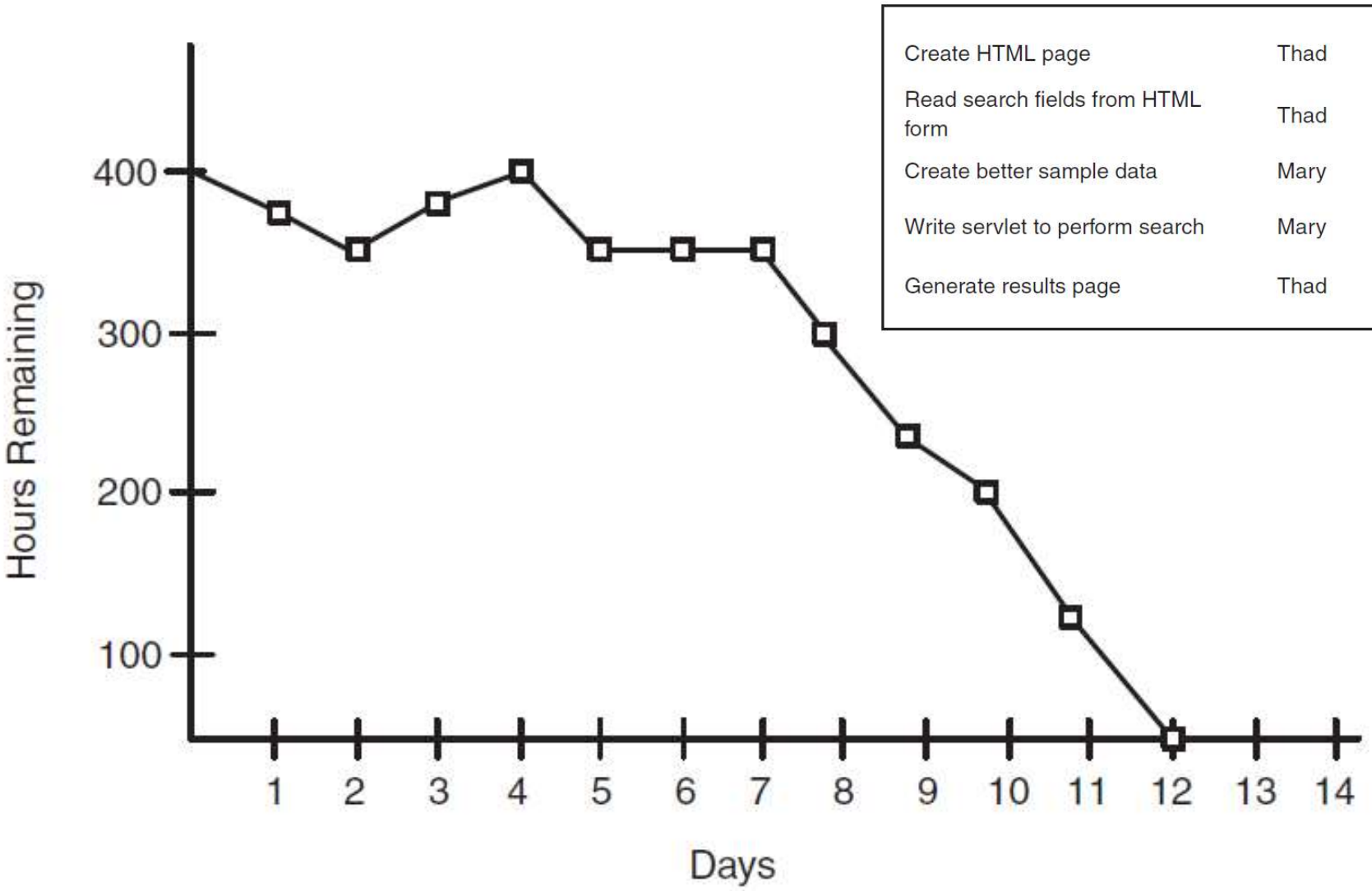
Monitoring
Evolution of story points

	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Story points at start of iteration	130	113	78	31
Completed during iteration	45	47	48	31
Changed estimates	10	4	–3	
Story points from new stories	18	8	4	
Story points at end of iteration	**113**	**78**	**31**	**0**

Monitoring
Burndown charts



Monitoring
Daily burndown charts



Create HTML page	Thad	~~2~~	0
Read search fields from HTML form	Thad	2	
Create better sample data	Mary	~~2~~	4
Write servlet to perform search	Mary	4	
Generate results page	Thad	2	

Monitoring

- A story estimated at **one story point** actually took **two days** to complete. How much does it contribute to velocity when calculated at the end of the iteration?

Monitoring

What is the velocity of this team?

Table 11.3 *Stories completed during an iteration.*

Story	Story Points	Status
Story 1	4	Finished
Story 2	3	Finished
Story 3	5	Finished
Story 4	3	Half finished
Story 5	2	Finished
Story 6	4	Not started
Story 7	2	Finished
Velocity	23 ?	16

Monitoring
Complete the table

	Iteration 1	Iteration 2	Iteration 3
Story points at start of iteration	100	76	34
Completed during iteration	35	40	36
Changed estimates	5	–5	0
Story points from new stories	6	3	2
Story points at end of iteration	76	34	0

Example
Online bookshop

Example
User stories

A user can do a basic simple search that searches for a word or phrase in both the author and title fields.	1
A user can search for books by entering values in any combination of author, title and ISBN.	1
A user can view detailed information on a book. For example, number of pages, publication date and a brief description.	1
A user can put books into a "shopping cart" and buy them when they are done shopping.	1
A user can remove books from their cart before completing an order.	½
To buy a book the user enters their billing address, the shipping address and credit card information.	2
A user can rate books from 1 (bad) to 5 (good). the book does not have to be one the user bought from us.	2
A user can write a review of a book. They can preview the review before submitting it. The book does not have to be one the user bought from us.	5
An administrator needs to approve or reject reviews before they are available on the site.	2

Example
User stories

A user can establish an account that remembers shipping and billing information.	2
A user can edit the credit card information stored in their account.	½
A user can edit the shipping and billing addresses stored in their account.	1
A user can put books into a "wish list" that is visible to other site visitors.	2
A user, especially a Non-Sailing Gift Buyer, can search for a wish list based on its owner's name and state.	1
A user can check the status of their recent orders.	½
If an order has not shipped, a user can add or remove books, change the shipping method, the delivery address and the credit card.	1
A user can place an item from a wish list (even someone else's) into their shopping cart.	½
A repeat customer must be able to find one book and complete an order in less than 90 seconds.	0

Must have
Should have
Could have
Won´t have

Example
User stories

A user can view a history of all of their past orders.	1
A user can easily re-purchase items when viewing past orders.	½
The site always tells a shopper what the last 3 (?) items they viewed are and provides links back to them (this works even between sessions).	1
A user can see what books we recommend on a variety of topics.	3
A user can choose to have items gift wrapped.	½
A user can choose to enclose a gift card and can write their own message for the card.	½
A Report Viewer can see reports of daily purchases broken down by book category, traffic, best- and worst-selling books and so on.	8
A user must be properly authenticated before viewing reports.	1
Orders made on the website have to end up in the same order database as telephone orders.	0

Example
User stories

An administrator can add new books to the site.	1
An administrator can delete a book.	½
An administrator can edit the informaion about an existing books.	1
The system must support peak usage of up to 50 concurrent users.	0

Must have

Should have

Could have

Won´t have

Example
Release plan for velocity=8

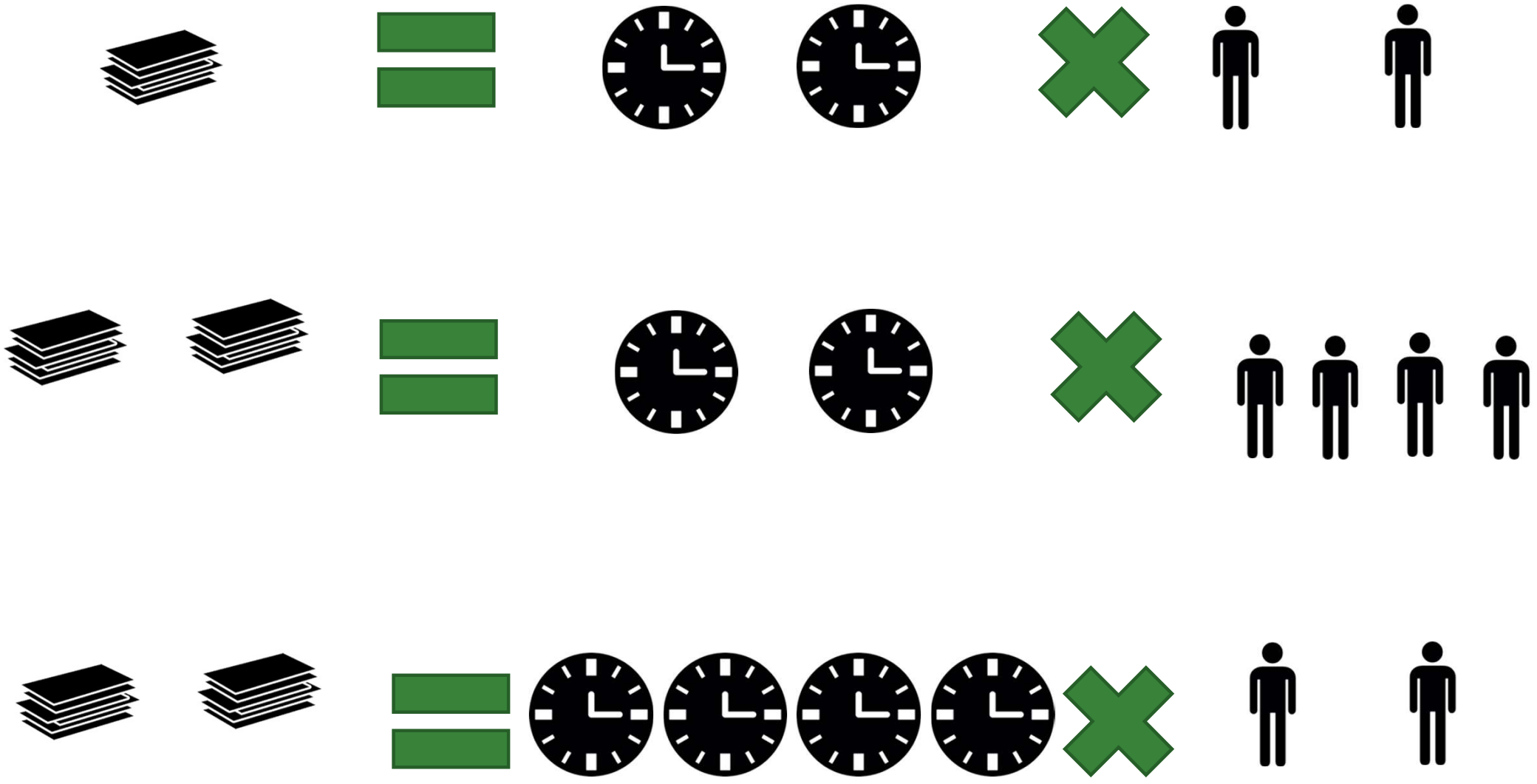
ITERATION 1	ITERATION 2
A user can do a basic simple search that searches for a word or phrase in both the author and title fields. (1)	An administrator can edit the informaion about an existing books. (1)
A user can put books into a "shopping cart" and buy them when they are done shopping. (1)	A user can search for books by entering values in any combination of author, title and ISBN. (1)
A user can remove books from their cart before completing an order. (½)	A user can edit the credit card information stored in their account. (½)
To buy a book the user enters their billing address, the shipping address and credit card information. (2)	A user can edit the shipping and billing addresses stored in their account. (1)
Orders made on the website have to end up in the same order database as telephone orders. (0)	A user can see what books we recommend on a variety of topics. (3)
A user can establish an account that remembers shipping and billing information. (2)	
An administrator can add new books to the site. (1)	
An administrator can delete a book. (½)	
The system must support peak usage of up to 50 concurrent users. (0)	

Task scheduling
Variables

$$\text{Duration} = \frac{\text{Work}}{\text{Units} \cdot \text{hd}}$$

Duration	Necessary time to finish the task
Work	Amount of work hours necessary to finish the task
Units	Full-time people assigned to the task
hd (constant)	Daily work hours of a person

Task scheduling
Variables



Task scheduling
Fixed units

$$\text{Duration} = \frac{\text{Work}}{\text{Units} \cdot \text{hd}}$$

Duration	10 days
Work	80 hours
Units	1 person (8h/d)

What if we increase the estimation by 20 hours?

Units	1 person
Work	100 hours
Duration	12,5 days

(we need more time to do it)

What if we only have 8 days?

Units	1 person
Duration	8 days
Work	64 hours

(we must do it faster)

Task scheduling
Fixed work

$$\text{Duration} = \frac{\text{Work}}{\text{Units} \cdot \text{hd}}$$

Duration	10 days
Work	80 hours
Units	1 person (8h/d)

What if we only have 8 days?

Work	80 hours
Duration	8 days
Units	1,25 people

(we include more people)

What if we have one additional developer?

Work	80 hours
Units	2 people
Duration	5 days

(we finish earlier)

Task scheduling
Fixed duration

$$\text{Duration} = \frac{\text{Work}}{\text{Units} \cdot \text{hd}}$$

Duration	10 days
Work	80 hours
Units	1 person (8h/d)

What if we increase the estimation by 20 hours?

Duration	10 days
Work	100 hours
Units	1,25 people

(we need more people)

What if we have one additional developer?

Duration	10 days
Work	80 hours
Units	2 people working 50%

(we can work more relaxed)

References

“User Stories Applied
For agile software development”
Mike Crohn
The Addison Wesley signature series
2009