Analiza și proiectarea sistemelor software

UC 2.0

Traducere și rezumat la
USE-CASE 2.0
The Guide to Succeeding with Use Cases
Ivar Jacobson
Ian Spence
Kurt Bittner
December 2011

Use Cases

Conferința OOPSLA’87 - Ivar Jacobson : primul articol despre use cases.
Use cases = abordare în zona de *cerințe* a procesului de dezvoltare de software.

Use Case: Totalitatea modurilor (*utile*) de utilizare a unui sistem pentru a realiza un *anumit scop* pentru un *anumit utilizator*.

Setul de cazuri de utilizare ale unui sistem = toate modurile *utile* de a folosi sistemul, ilustrând *valoarea produsă de acesta*.

Clarifică ce FACE și ce NU FACE sistemul.

Use Case:

- O *secvență de acțiuni* realizate de sistem care conduce la un *rezultat observabil* de *valoare* pentru un anumit *utilizator.*
- *Comportament* specific al unui sistem care participă într-o *colaborare* cu un utilizator pentru a livra ceva *de valoare* pentru acel utilizator.
- Cea mai mică *unitate de activitate* care oferă un *rezultat semnificativ* utilizatorului.
- *Context* pentru un set de *cerințe corelate*.

PROBLEME ale metodelor agile bazate doar pe *user stories*

- Dificultatea de a avea o vedere de ansamblu în contextul multor *story cards*
- Dificultatea de a stabili priorități între multe fragmente de *stories* din *backlog*
- Acces limitat la stakeholders
- Dificultatea de a vedea ce face sistemul curent și dacă este complet pentru a fi util
- Dorința de a obține cât mai mult de la produse COTS
- Dificultatea de a stabili o arhitectură eficace
- Necesitatea de a avea trasabilitate de la cerințe la teste
- Necesitatea de a avea permanent accesibile informații generale despre ce face sistemul
- Necesitatea de a gestiona setul testelor aflat în continuă creștere
- Necesitatea de a realiza modificări sincronizate cu multe aplicații suport
- Modelele și obiectivele afacerii nealiniate cu modelele sistemului
- Conversațiile dintre echipa de dezvoltare și conducere încep să se destrame

SUBIECTE TRATATE

Concepte UC 2.0

Principiile aplicării cu succes a UC 2.0

Descriere esențializată a practicii UC 2.0

Use-Case 2.0

Use-Case 2.0 - descris ca ***practică*** de dezvoltare condusă de cazuri de utilizare, demonstrată și bine definită.

- light, agile – inspirată de *user stories* și de metodologiile agile Scrum și Kanban.
- suport pentru cerințe, dar și pentru architectură, proiectare, test, și UX (user experience)
- mijloc pentru modelarea afacerilor și pentru reutilizare software

Concept nou : **Use-case slice** - adaptare la modurile de lucru specifice Scrum (uitilzare backlog) și Kanban (one-piece flow).

Use-Case 2.0 *include tehnicile* furnizate de *user stories* dar oferă suport semnificativ pentru utilizare la *sisteme mari*, *echipe mari* și *dezvoltare mai complexă și exigentă*:

- de categorie ușoară, ca și *user stories*,
- scalează în mod fluent și structurat pentru a incorpora oricâte detalii necesare.
- dirijează și conectează multe alte aspecte ale dezvoltării de software.

Use-Case 2.0

Use-Case 2.0: *Practică** agilă scalabilă care folosește cazuri de utilizare pentru a captura un set de cerințe și a dirija *dezvoltarea incrementală* a unui sistem pentru a le îndeplini.

Utilitate largă :

- Echipe mici, agile, care dezvoltă aplicații orientate pe utilizator
- Proiecte mari de dezvoltare sisteme complexe interconectate (ex. Sisteme enterprise, linii de produse, sisteme în cloud)

Dirijează activitățile de dezvoltare:

- analiză, proiectare, planificare, estimare, urmărire și testare.

*Descriere la :*https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication*

Ivar Jacobson, Ian Spence, and Brian Kerr
,Use Case 2.0. The hub of software development, ACM Queue, April 5, 2016
Volume 14, issue 1

https://www.youtube.com/watch?v=p5gDbf0je8k

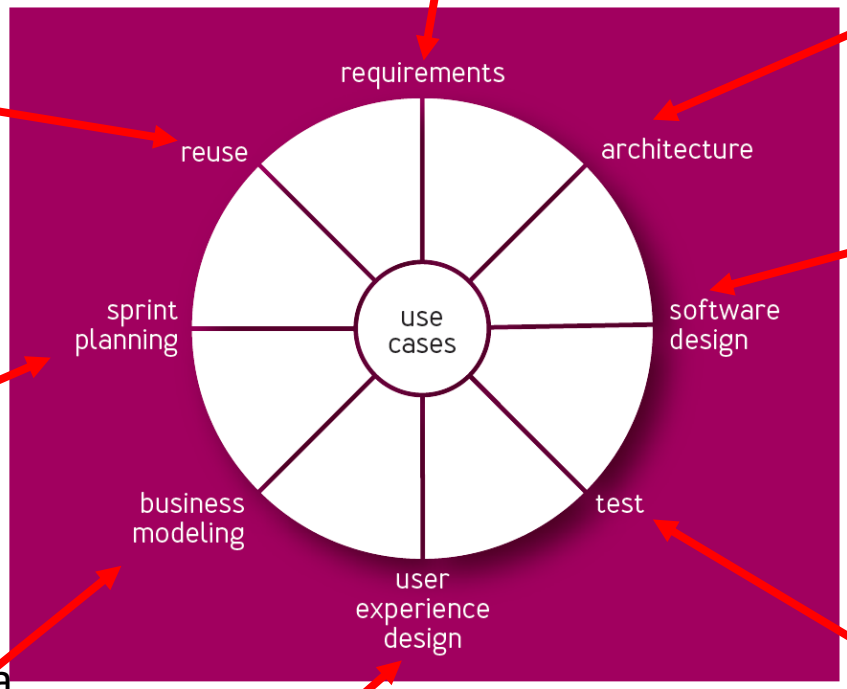
Use-Case 2.0

Capturare *cerințe* din perspectiva utilizării sistemului.

Asistă sistematic în proiectarea *architecturii* aplicației.

Ajută la găsirea elementelor comune care să permită *reutilizarea* software-lui.

FIGURE 1: USE CASES ARE THE HUB OF SOFTWARE DEVELOPMENT



Dirijează identificarea componentelor și a altor elemente software în *proiectarea software-lui.*

UC slice - element plasat în backlog la planificare *sprints* sau pe canvas când se utilizează Kanban.

Modelare business cu UCs conduce direct la identificarea cazurilor de utilizare ale sistemului ce trebuie dezvoltat ca suport pentru afacere.

Proiectarea unei *experiențe utilizator (UX*) practică.

UC slice - element ce trebuie *testat* ⇒ suport pentru test-driven design.

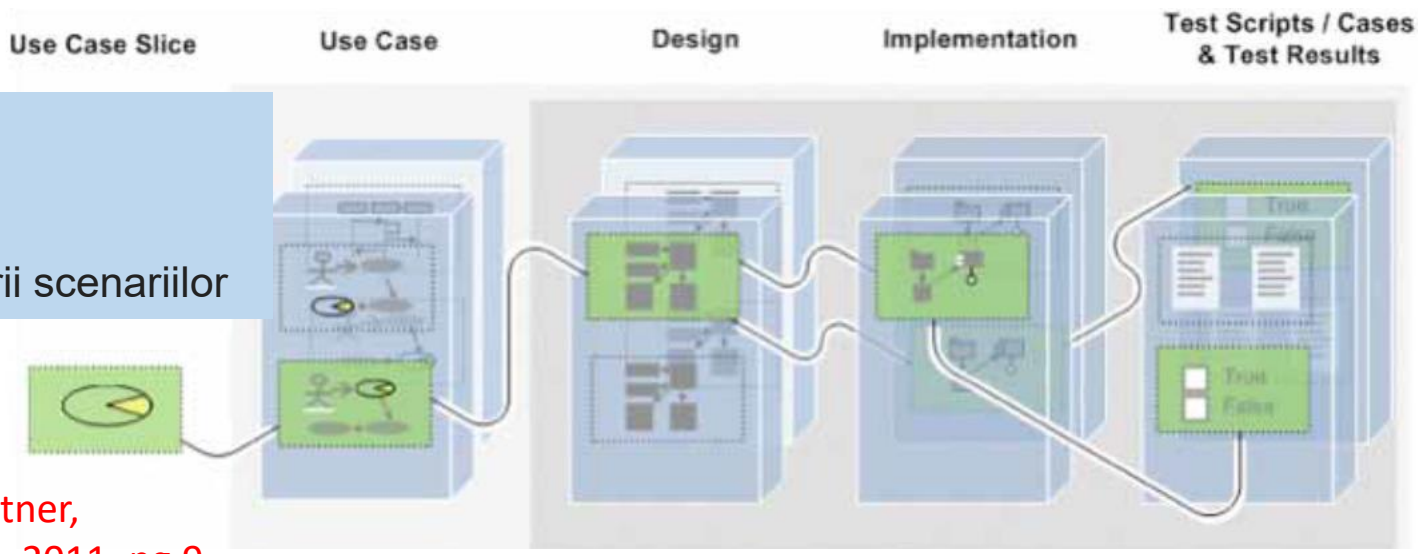
Use-Case 2.0

Use-Case 2.0: Practică agilă scalabilă care folosește cazuri de utilizare pentru a captura un set de cerințe și a dirija dezvoltarea incrementală a unui sistem pentru a le îndeplini.

Use Case Slice : *unitate de lucru* formată din unul sau mai multe *scenarii (stories),* care are *o valoare clară pentru utilizator*.

- Întreaga activitate necesară realizării complete a scenariilor selectate.
- Evoluează pe parcursul proiectării, implementării și testării, pentru a include scenariile respective.
- Ajută inclusiv la dirijarea activităților de dezvoltare pentru îndeplinirea cerințelor.

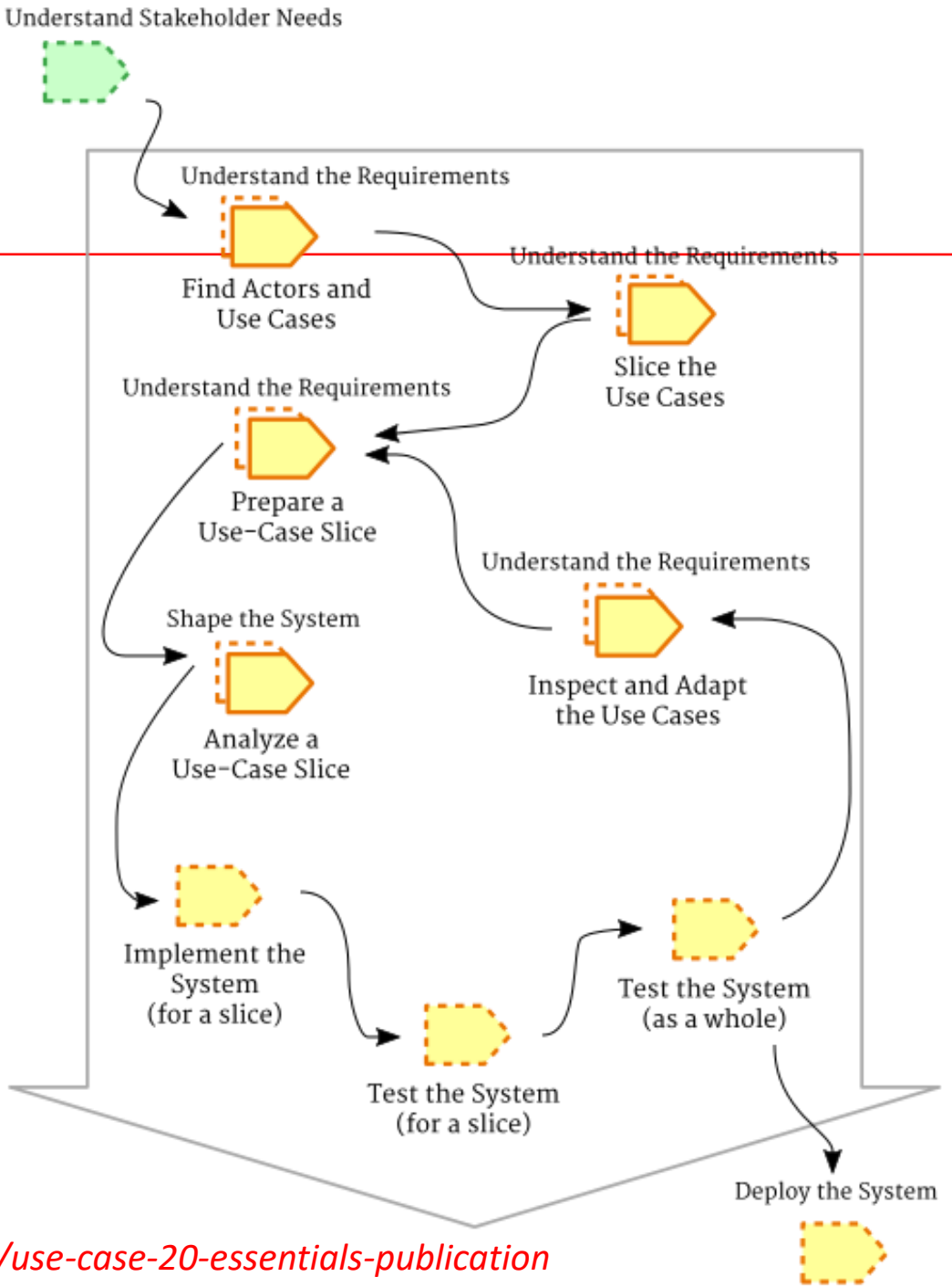
UC slice definește :
- Scenariile
- Setul de cazuri de testare
- Activitățile necesare realizării scenariilor



Ivar Iacobson, Ian Spence, Kurt Bittner,
USE-CASE 2.0 The Definitive Guide, 2011, pg.9

UC 2.0
Fluxul de activități

Use-Case 2.0: Practică agilă scalabilă care folosește cazuri de utilizare pentru a captura un set de cerințe și a dirija dezvoltarea incrementală a unui sistem pentru a le îndeplini.



Activity Space

Activity of the UC 2.0 practice

https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

SUBIECTE TRATATE

Concepte UC 2.0

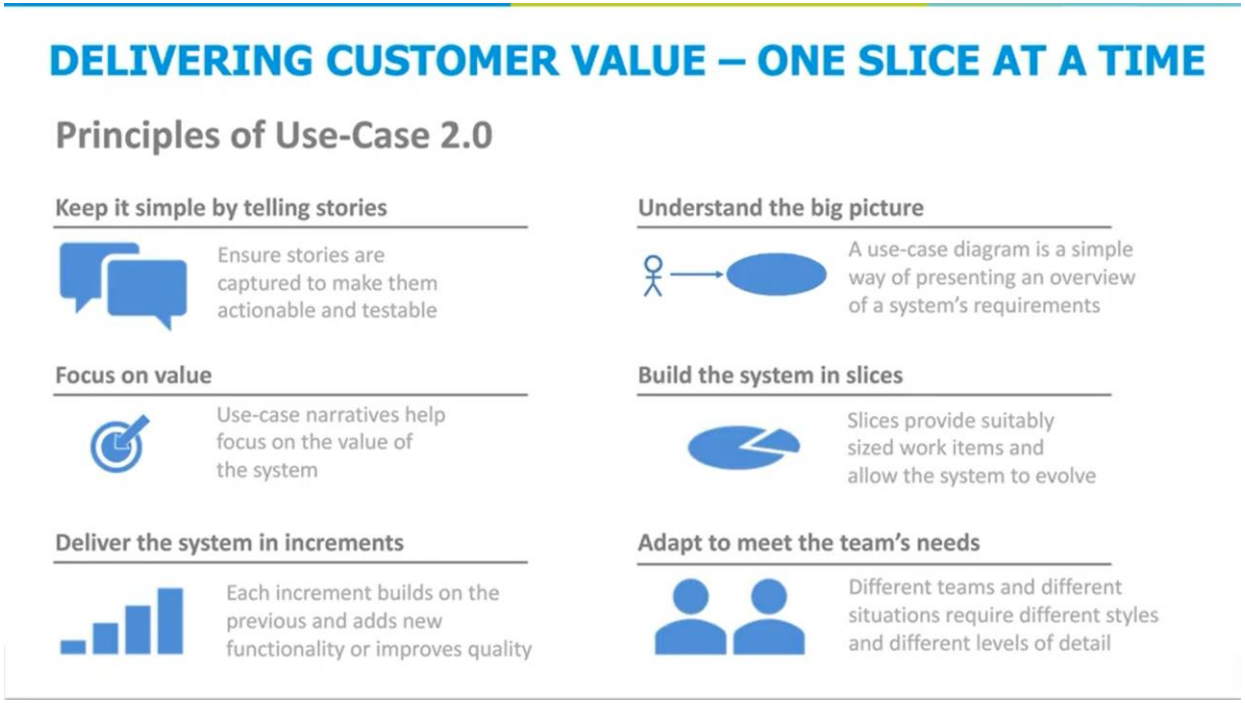
Principiile aplicării cu succes a UC 2.0

Descriere esențializată a practicii UC 2.0

Principiile aplicării cu succes a UC 2.0

1. Simplitate, bazată pe scenarii
2. Vedere de ansamblu
3. Focalizare pe valoare
4. Construire sistem în felii (slices)
5. Livrare system în incremente
6. Adaptare la nevoile echipei

https://www.youtube.com/watch?v=p5gDbf0je8k



Principiile aplicării cu succes a UC 2.0
1. Simplitate, bazată pe scenarii

Cazurile de utilizare oferă un mod de a identifica și a captura, într-un mod simplu și complet, toate *scenariile* (***stories***) diferite care sunt *în relație*. Scenariile sunt capturate într-un mod care le face acționabile și testabile.

$\Rightarrow$ Cerințele sunt ușor capturate, partajate și înțelese.

Cazurile de testare sunt părțile cele mai importante ale descrierii cazului de utilizare (chiar mai importante decât descrierea textuală):

- transformă scenariile în realitate.
- pot demonstra fără ambiguități că sistemul face ceea ce trebuie să facă.
- definesc ceea ce înseamnă o implementare de succes a cazului de utilizare.

Principiile aplicării cu succes a UC 2.0

2. Imagine de ansamblu

Diagrama cazurilor de utilizare, este un mod simplu de a reprezenta o **vedere de ansamblu a cerințelor** sistemului.

- Pe diagrama UC sunt reprezentate :
  - *cine declanșează* interacțiunile,
  - toate *modurile de utilizare* a sistemului,
  - toate *elementele de context* implicate

Actori = părți interesate (stakeholders) care folosesc sistemul și contribuie la îndeplinirea obiectivelor.

Cazurile de utilizare – toate modurie **utile** de a folosi sistemul pentru a realiza/obține aceste obiective.

Principiile aplicării cu succes a UC 2.0
2. Imagine de ansamblu

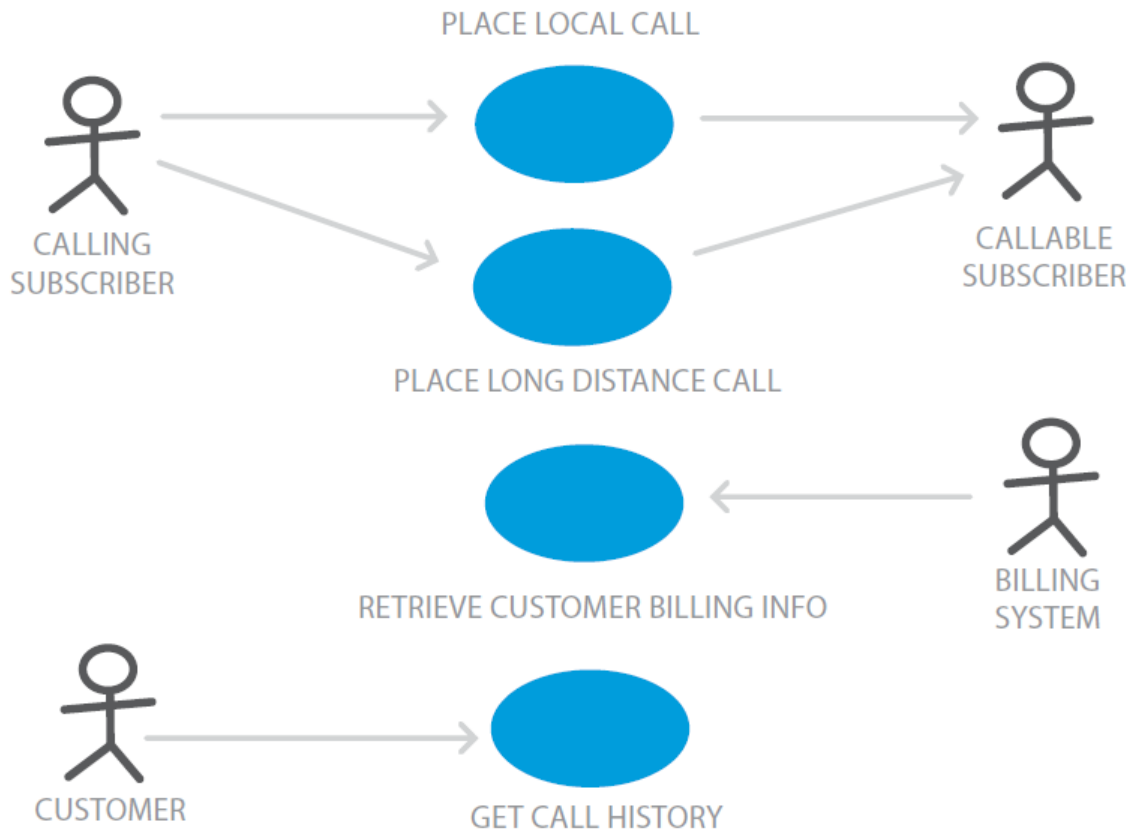
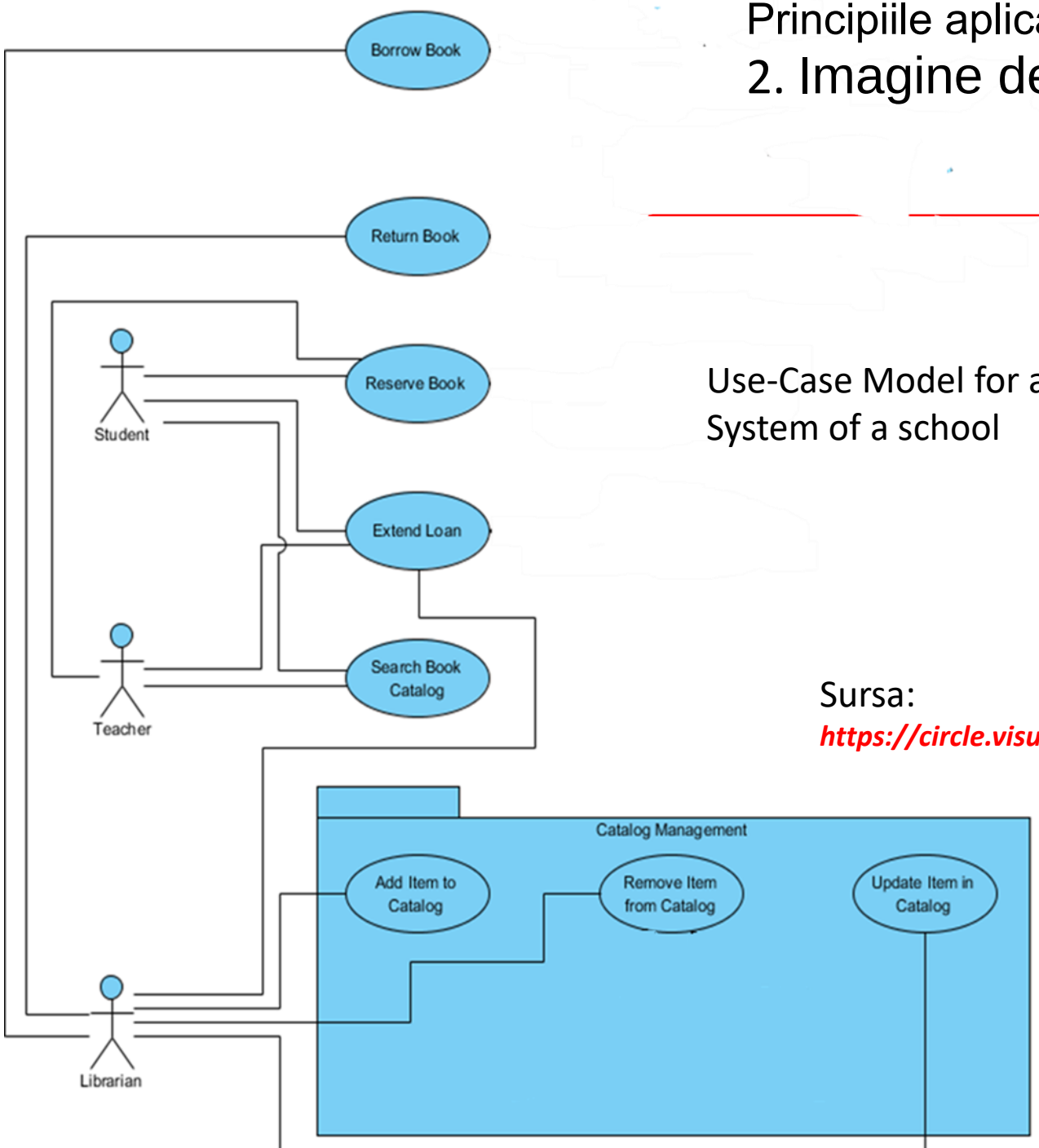


FIGURE 1: THE USE-CASE DIAGRAM FOR A SIMPLE TELEPHONE SYSTEM

Principiile aplicării cu succes a UC 2.0
2. Imagine de ansamblu



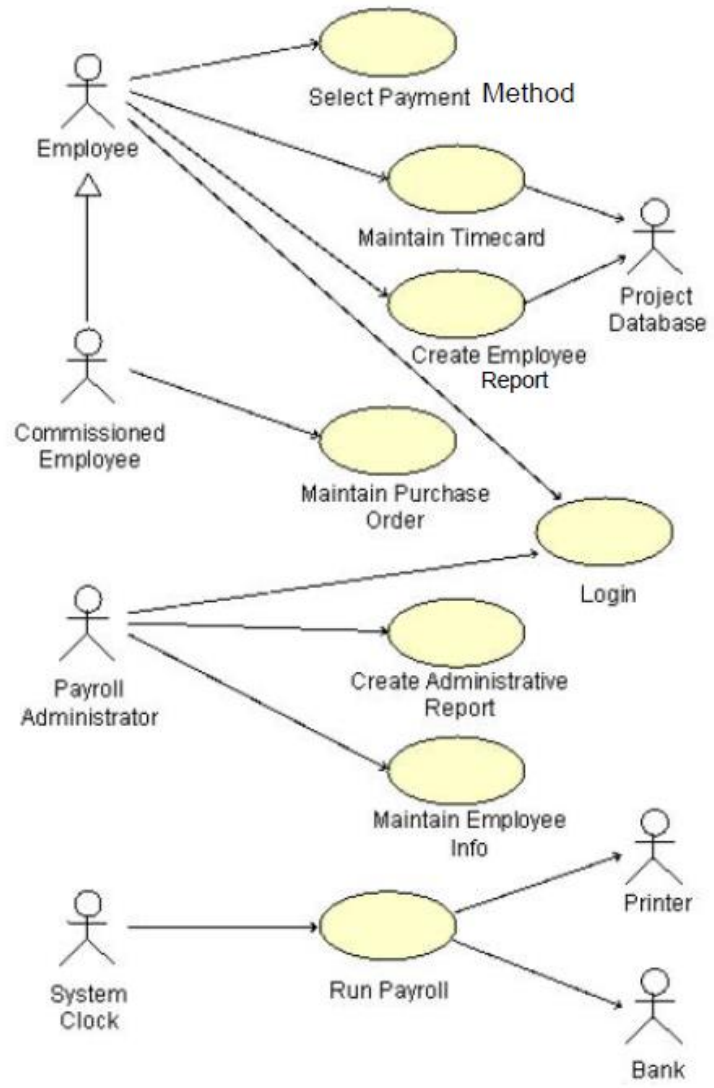
Use-Case Model for a Library Management System of a school

Sursa:
https://circle.visual-paradigm.com/steps-wizards/use-case-2.0

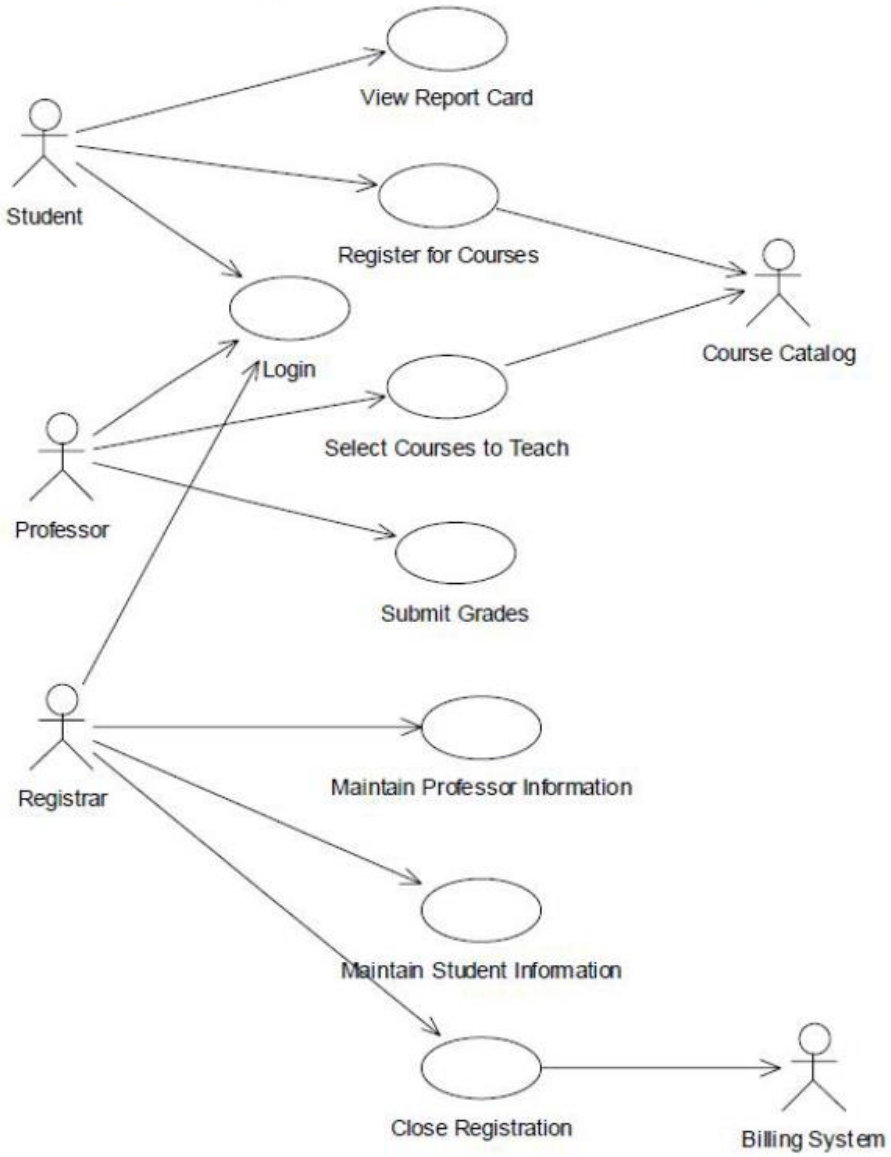
Principiile aplicării cu succes a UC 2.0

2. Imagine de ansamblu

Payroll System Use-Case Model Main Diagram



Course Registration System Use-Case Model Main Diagram



Principiile aplicării cu succes a UC 2.0

3. Focalizare pe valoare

Focalizare pe valoarea pe care o oferă utilizatorilor și altor părți interesate, din perspectiva acestora. Adică, cum va fi utilizat sistemul pentru a atinge un anumit scop pentru un anumit utilizator.

Valoarea se generează atunci când sistemul este folosit.

⇓

Focalizare pe *felul în care este utilizat sistemul* și nu pe o listă de funcții și caracteristici pe care le va oferi.

Use cases are useful casses not useless cases !!!

Principiile aplicării cu succes a UC 2.0

3. Focalizare pe valoare

Focalizare pe felul în care este utilizat sistemul și nu pe o listă de funcții și caracteristici pe care le va oferi.

- **Procedura :**
  - Identificarea celui mai simplu* mod de a atinge obiectivul
  - Capturarea modurilor alternative de a atinge scopul
  - Capturarea modurilor de gestionare a problemelor ce pot să apară pe parcursul acțiunilor implicate în atingerea scopului
- **Consecințe :**
  - Clarificarea relațiilor dintre modurile de utilizare a sistemului
  - Identificarea celor mai valoroase moduri și promovarea acestora
  - Adăugarea ulterioară a modurilor mai puțin valoroase, fără a strica sau modifica ceea ce s-a făcut deja.

Concentrare pe *Basic Flow*. Completare ulterioară a fluxurilor alternative, pe măsură ce devin necesare.

*Obs. Uneori implementarea altor moduri decât cele mai simple nu aduce valoare semnificativă, dar alteori acestea pot face diferența semnificativă pentru câștigarea pieței.

Principiile aplicării cu succes a UC 2.0
3. Focalizare pe valoare

BASIC FLOW	ALTERNATIVE FLOWS
1. Insert Card	A1 Invalid Card
2. Validate Card	A2 Non-Standard Amount
3. Select Cash Withdrawal	A3 Receipt Required
4. Select Account	A4 Insufficient Funds in ATM
5. Confirm Availability of Funds	A5 Insufficient Funds in Acct
6. Return Card	A6 Would Cause Overdraft
7. Dispense Cash	A7 Card Stuck
	A8 Cash Left Behind
	etc..

FIGURE 2: THE STRUCTURE OF A USE-CASE NARRATIVE

Detaliile UC au o *structură aditivă*:

- Fluxul principal este absolut necesar și trebuie implementat primul
- Fluxurile alternative sunt opționale și pot fi adăugate atunci când devin necesare

⇓

- Nu este necesară livrarea întregului caz de utilizare, ci doar a părților care oferă cea mai mare valoare.

⇓

- Nu este necesar un model complet al cazurilor de utilizare înainte de începerea dezvoltării sistemului. Odată înțeles cel mai important caz de utilizare și fluxul său principal, există deja ceva de valoare ce se poate adăuga sistemului.

Focalizarea pe valoare în crearea scenariilor (stories) și validarea acestora pentru completitudine (cu filtrarea celor cu valoare redusă).

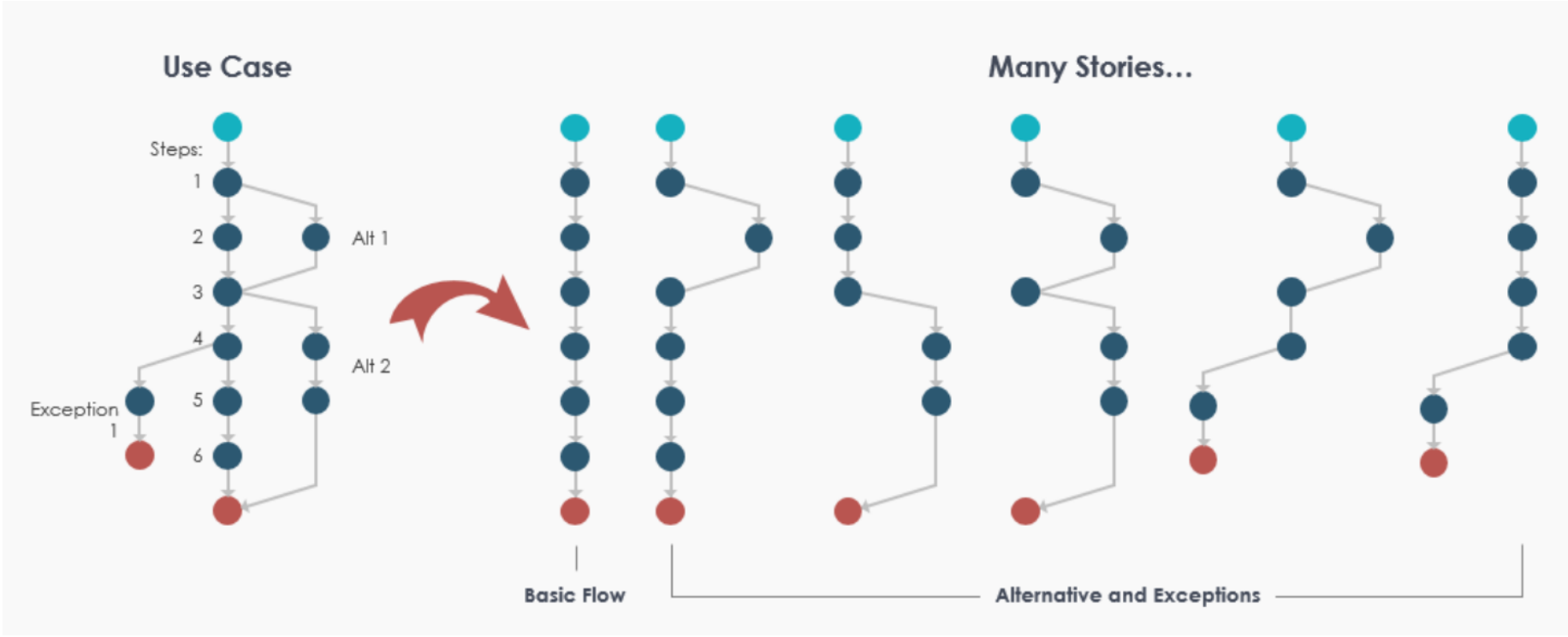
⇓

Lansări** (release) **de dimensiuni mici ale a sistemului, dar care oferă valoare reală utilizatorilor și părților interesate care finanțează dezvoltarea sistemului.

Principiile aplicării cu succes a UC 2.0
4. Construire sistem în felii (slices)
UC și UC stories (scenarii/fluxuri)

Use case story poate fi:

- Fluxul principal, ce descrie modul tipic (de regulă cel mai simplu) de realizare a obiectivului
- Flux alternativ de realizare a obiectivului
- Flux excepțional



Sursa:
https://circle.visual-paradigm.com/steps-wizards/use-case-2.0

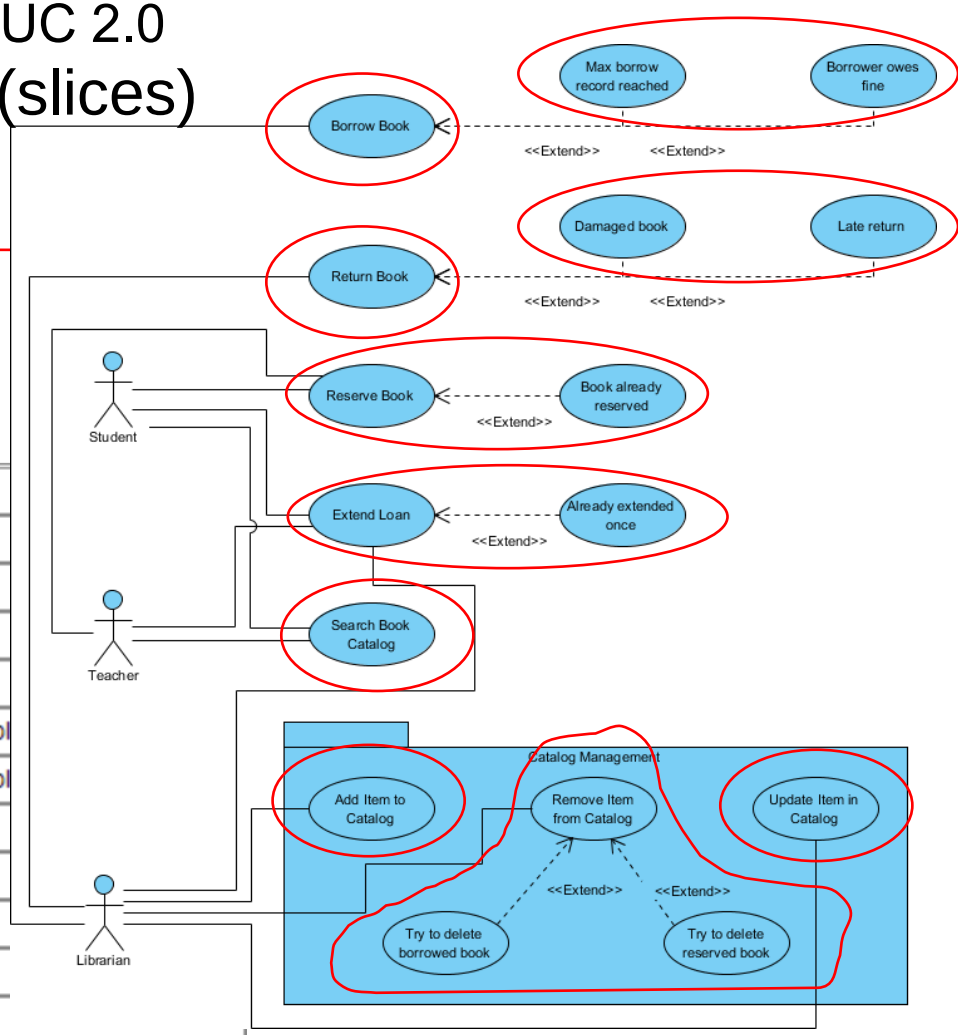
Principiile aplicării cu succes a UC 2.0

4. Construire sistem în felii (slices)

EXEMPLU

Gruparea fluxurilor în slice-uri este flexibilă, depinzând de valoarea utilă de livrat cu fiecare increment.

Use Case	Use Case Story	Use Case Slice
Borrow Book	Borrow Book (Basic)	Borrow book success
Borrow Book	Max borrow record reached	Borrow book failed
Borrow Book	Borrower owes fine	Borrow book failed
Return Book	Return Book (Basic)	Return book success
Return Book	Damaged book	Return book - user probl
Return Book	Late return	Return book - user probl
Reserve Book	Reserve Book (Basic)	Reserve book
Reserve Book	Book already reserved	Reserve book
Extend Loan	Extend Loan (Basic)	Extend loan
Extend Loan	Already extended once	Extend loan
Search Book Catalog	Search Book Catalog (Basic)	Search book catalog
Update Item in Catalog	Update Item in Catalog (Basic)	Manage catalog - Update
Remove Item from Catalog	Remove Item from Catalog (Basic)	Manage catalog - Remove
Remove Item from Catalog	Try to delete reserved book	Manage catalog - Remove
Remove Item from Catalog	Try to delete borrowed book	Manage catalog - Remove
Add Item to Catalog	Add Item to Catalog (Basic)	Manage catalog - Add



Sursa:

https://circle.visual-paradigm.com/steps-wizards/use-case-2.0

Principiile aplicării cu succes a UC 2.0
4. Construire sistem în felii (slices)

UC slice = cerință, elementele de proiectare (design) implicate în realizare slice, elementele de cod care implementează cerința din slice, cazuri de testare, scripturi de testare și rezultatele testării.

UC slice definește :

- Scenariile
- Setul de cazuri de testare
- Activitățile necesare realizării scenariilor

UC slice asigură trasabilitate de la cerințe la cod și la test.

⇒ înțelegerea impactului asupra proiectării și implementării sistemului:
  - modificare elemente existente
  - posibil noi elemente necesare
  - dimensiunea impactului și decizia de a implementa sau nu slice-ul

Principiile aplicării cu succes a UC 2.0
5. Livrare sistem în incremente

Modul recomandat de a produce sisteme software :

- Livrare cu lansări multiple, fiecare dezvoltabilă în incremente.
- Fiecare increment produce o versiune a sistemului demonstrabilă sau utilizabilă.

Aplicare UC 2.0

- Fiecare increment va cuprinde slice-uri din diferite cazuri de utilizare.

Prin felierea cazurilor de utilizare obținem un *control mai fin*, ceea ce permite *maximizarea valorii oferită la fiecare lansare*.

UC slices : alocate la incremente asigură că fiecare increment este mai mare decât, și se construiește peste cel anterior.
Slice-uri : implementabile și testabile independent în fiecare increment.

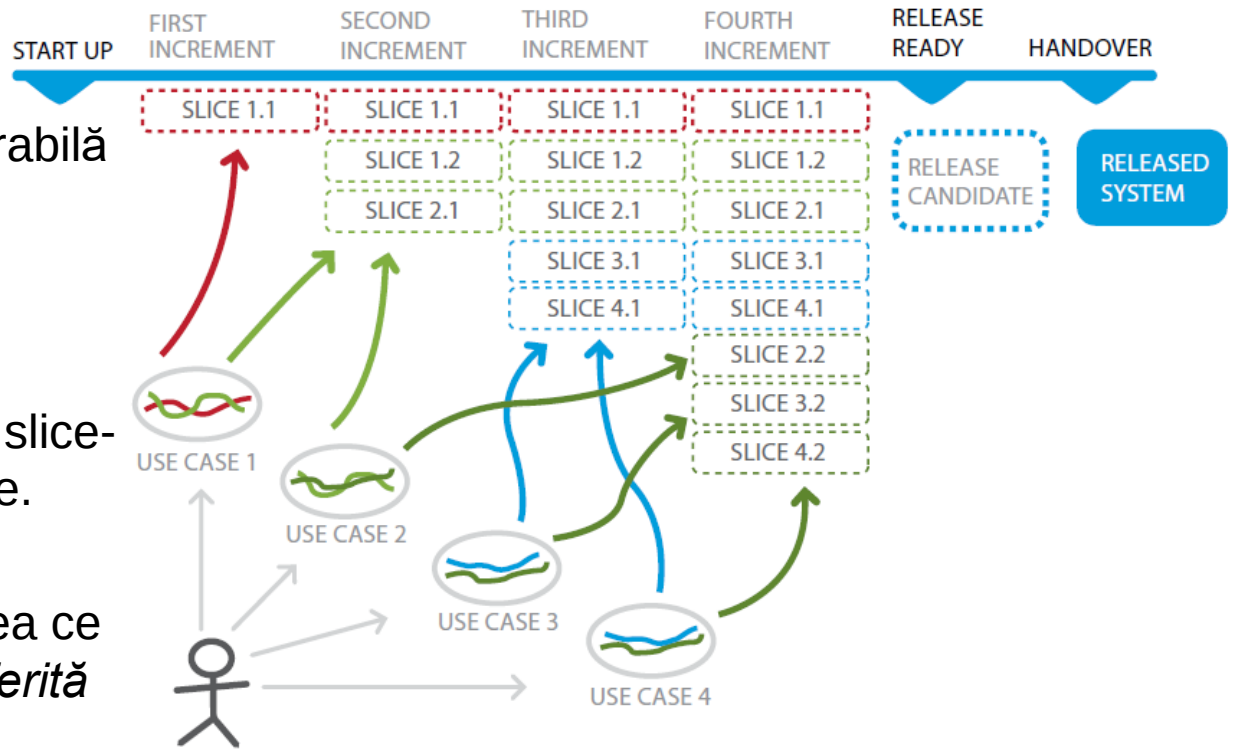


FIGURE 4: USE CASES, USE-CASE SLICES, INCREMENTS, AND RELEASES

Principiile aplicării cu succes a UC 2.0

6. Adaptare la nevoile echipei

Use-Case 2.0

- Echipe mici, colaborative : descrieri simple (lightweight) ale UCs capturează esența fundamentală a scenariilor și pot fi notate pe carduri index simple.

- Echipe mari distribuite : descrieri UC mai detaliate și organizate în documente. Echipa poate pleca de la esență și adaugă detalii pe măsură ce apar probleme.

SUBIECTE TRATATE

Concepte UC 2.0

Principiile aplicării cu succes a UC 2.0

Descriere esențializată a practicii UC 2.0

UC 2.0

Use-Case 2.0 se referă la

- cerințe
- sistemul dezvoltat pentru îndeplinirea cerințelor
- testele folosite pentru a demonstra că sistemul îndeplinește cerințele

Nucleul Use-Case 2.0

- use-case – capturare cerințe
- story
- use-case slice – gestionarea domeniului de aplicare a unui use-case; manipulabile independent; corelate cu user stories

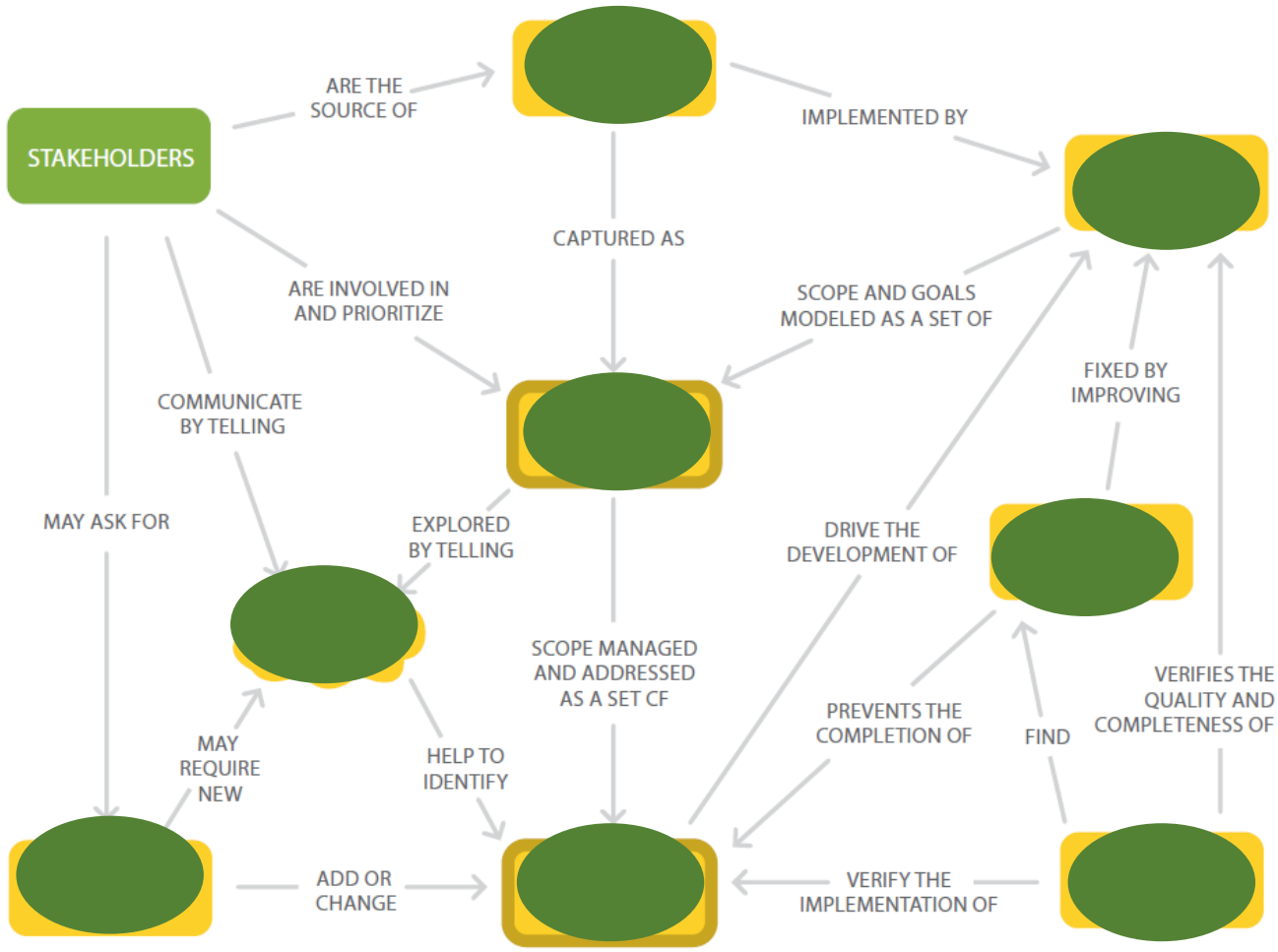


FIGURE 5: USE-CASE 2.0 CONCEPT MAP.

UC 2.0 : Things to work with (alphas), work products, and things to do (activities)

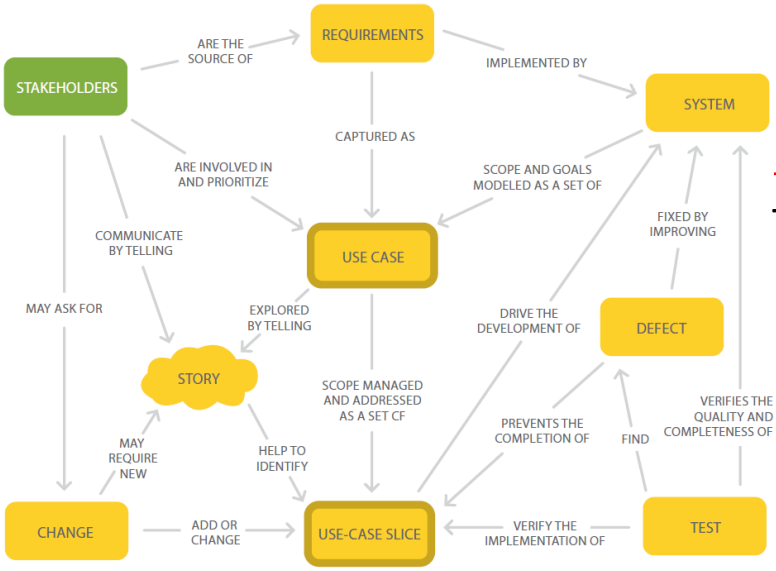


FIGURE 5: USE-CASE 2.0 CONCEPT MAP.

Things to work with (alphas) : UC și UC slice

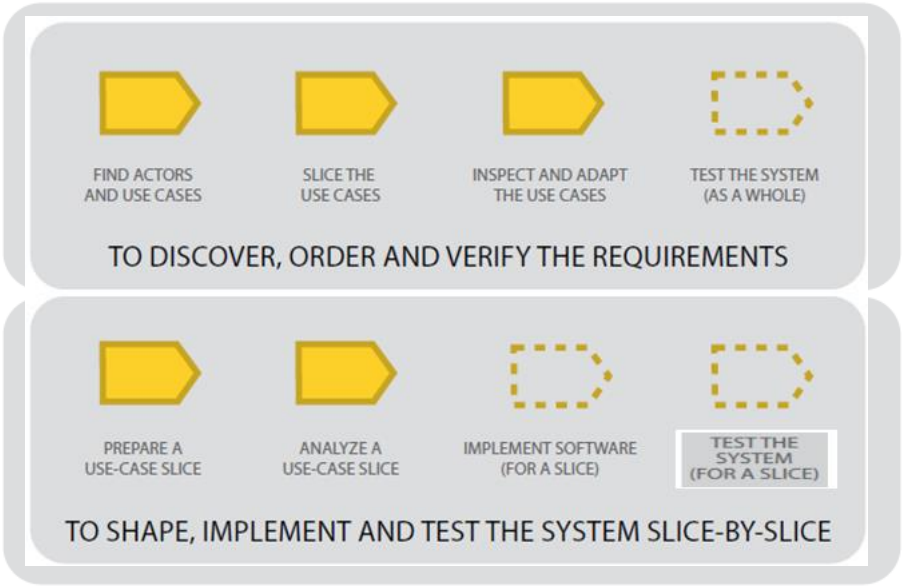


FIGURE 15: THE ACTIVITIES IN USE-CASE 2.0

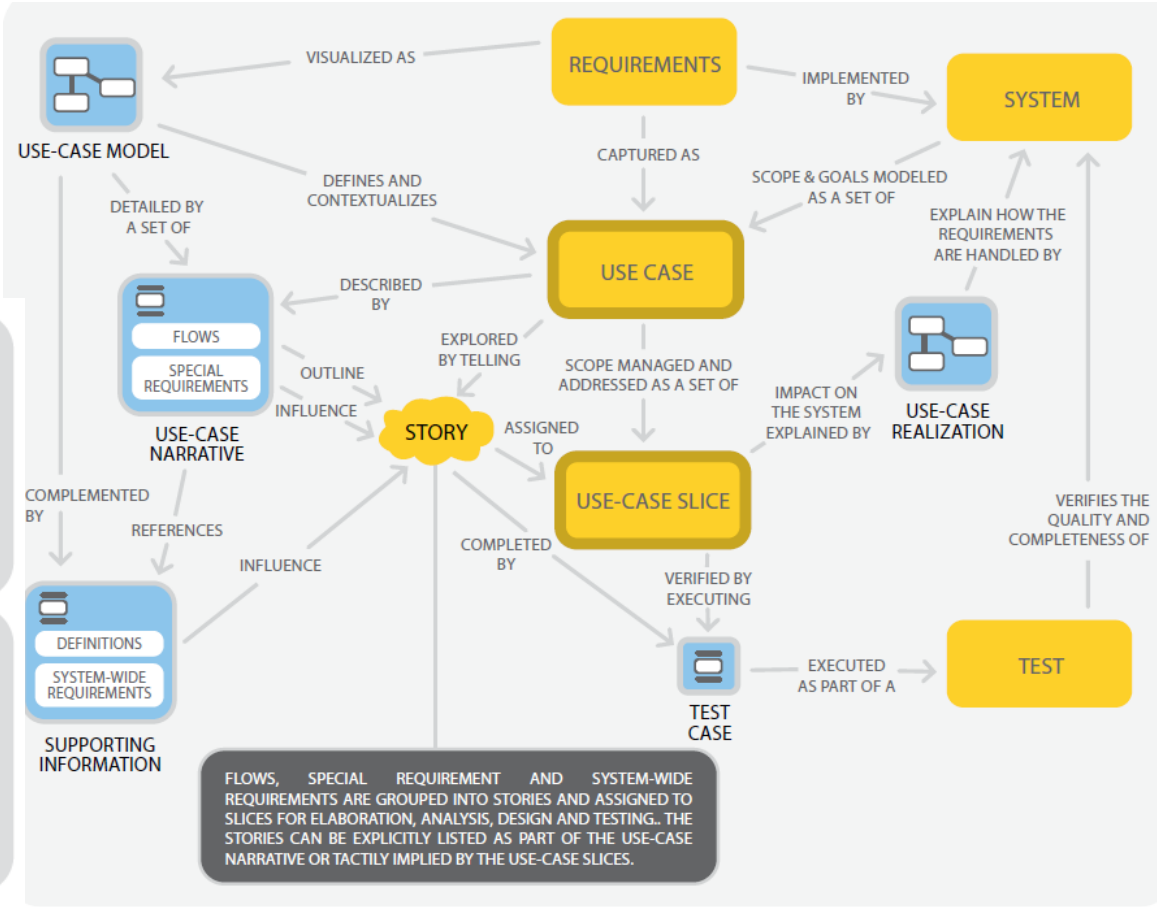


FIGURE 9: THE USE-CASE 2.0 WORK PRODUCTS

UC 2.0 : Things to work with

Use case, story

1) *Goal Established*: when the goal of the use case has been established.

2) *Story Structure Understood*: when the structure of the use-case narrative has been understood enough for the team to start work identifying and implementing the first use-case slices.

3) *Simplest Story Fulfilled*: when the system fulfills the simplest story that allows a user to achieve the goal.

4) *Sufficient Stories Fulfilled*: when the system fulfills enough of the stories to provide a usable solution.

5) *All Stories Fulfilled*: when the system fulfills all the stories told by the use case.

https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

FIGURE 6: THE LIFECYCLE OF A USE CASE

UC 2.0 : Things to work with

Use case slice

1) *Scoped*: when it has been scoped and the extent of the stories covered has been clarified.
2) *Prepared*: when the slice has been prepared by enhancing the narrative and test cases to clearly define what it means to successfully implement the slice.
3) *Analyzed*: when the slice has been analyzed so its impact on the components of the system is understood and the pieces affected are ready for coding and developer testing.
4) *Implemented*: when the software system has been enhanced to implement the slice and the slice is ready for testing.
5) *Verified*: and finally when the slice has been verified as done and is ready for inclusion in a release.

FIGURE 7:
THE LIFECYCLE OF A USE-CASE SLICE

https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

UC 2.0 : Work products

https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

Use-Case Model

Use-Case Narrative

Use-Case Realization

Supporting Information

Use Case Test Case

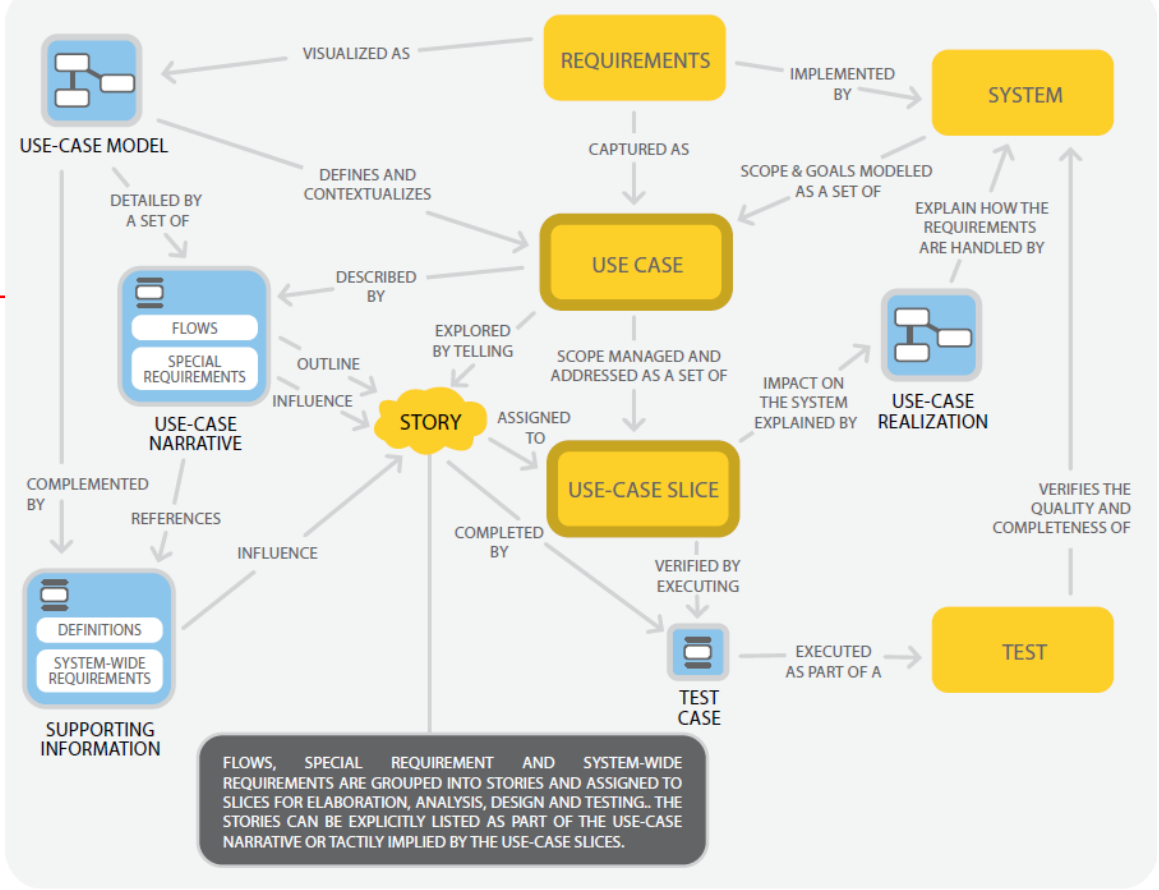
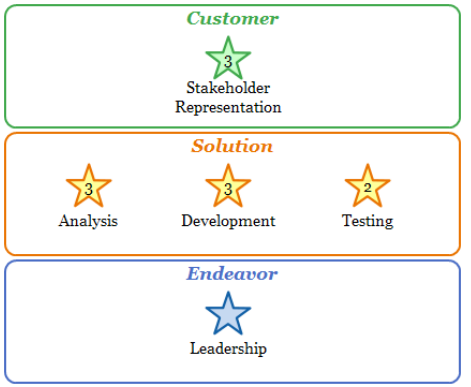


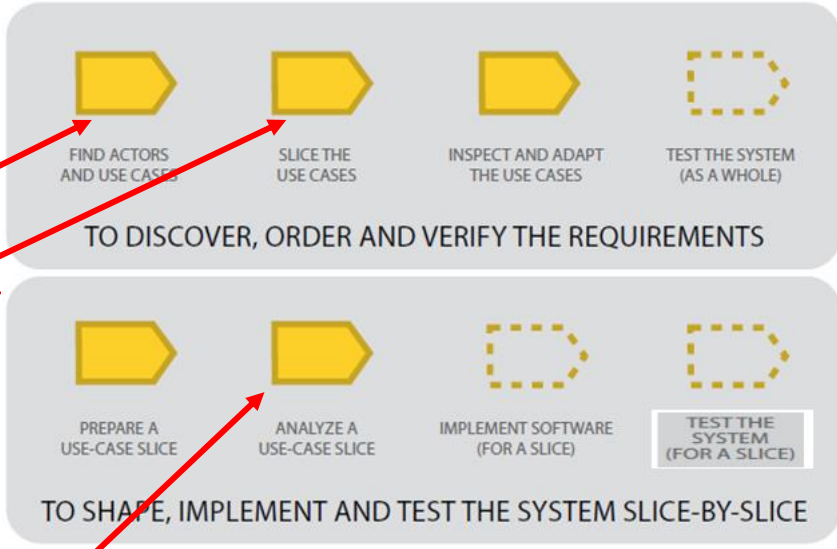
FIGURE 9: THE USE-CASE 2.0 WORK PRODUCTS

	USE-CASE MODEL	USE-CASE NARRATIVE	USE--CASE REALIZATION	TEST CASE	SUPPORTING INFORMATION
SKETCH:		BRIEFLY DESCRIBED		TEST IDEAS FORMULATED	OUTLINED
BARE ESSENTIALS:	VALUE ESTABLISHED	BULLETED OUTLINE	IMPLEMENTATION ELEMENTS IDENTIFIED	SCENARIO CHOSEN	SIMPLY DEFINED
ENHANCED:	SYSTEM BOUNDARY ESTABLISHED	ESSENTIAL OUTLINE	RESPONSIBILITIES ALLOCATED	VARIABLES IDENTIFIED	MODELLED & ILLUSTRATED
EXPANDED:	STRUCTURED	FULLY DESCRIBED	INTERACTION DEFINED	VARIABLES SET	COMPREHENSIBLY DEFINED
FURTHER EXPANDED:				SCRIPTED /AUTOMATED	

FIGURE 14: WORK PRODUCT LEVELS OF DETAIL



UC 2.0 : Things to do



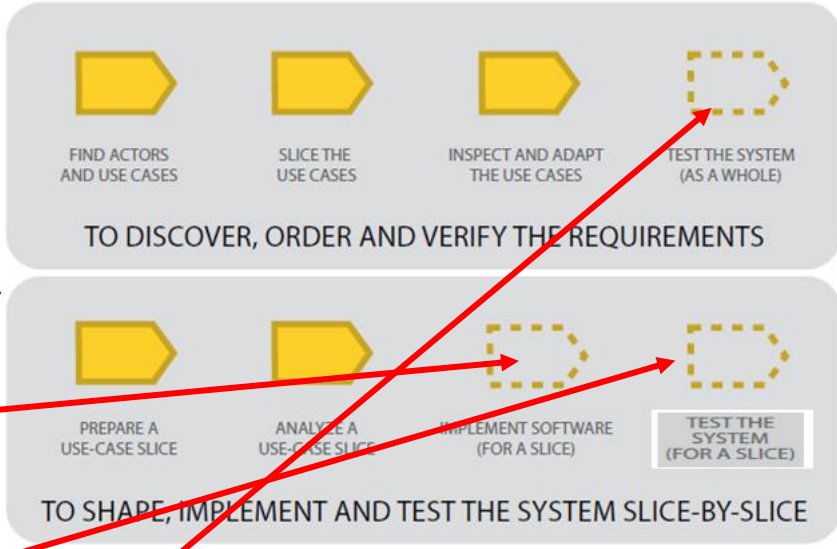
Procedură :

1. Identificare actori și UCs
   - ședință cu stakeholders
   - identificare moduri relevante de utilizare
2. Feliere UCs
   - ordonare UCs
   - focalizare pe cele mai importante fără a detalia restul
   - creare primul slice din fluxul principal (sau chiar parte a acestuia) al celui mai important UC.
   - creare doar slice-urile necesare pentru a avansa proiectul; restul vor fi atacate când vor fi necesare

rezultă MVP (minimum viable product) – criterii : combinație de prioritate, valoare, risc, necesitate.

3. Analizare UC slice
   - Definirea realizării UC de către elemente ale sistemului

UC 2.0 : Things to do



Procedură (cont.):

4. *Implementare slice*

- Proiectare, codificare, unit testare, integrare componente software ce implementează UC slice.

Obs. Se poate lucra în paralel la implementarea diferitelor slice-uri, după care este necesară reconcilierea implementărilor.

5. *Testare slice*

- Testare la nivel de slice, deoarce slice este independent funcțional.
- Utilizare cazuri de testare definite pentru slice.

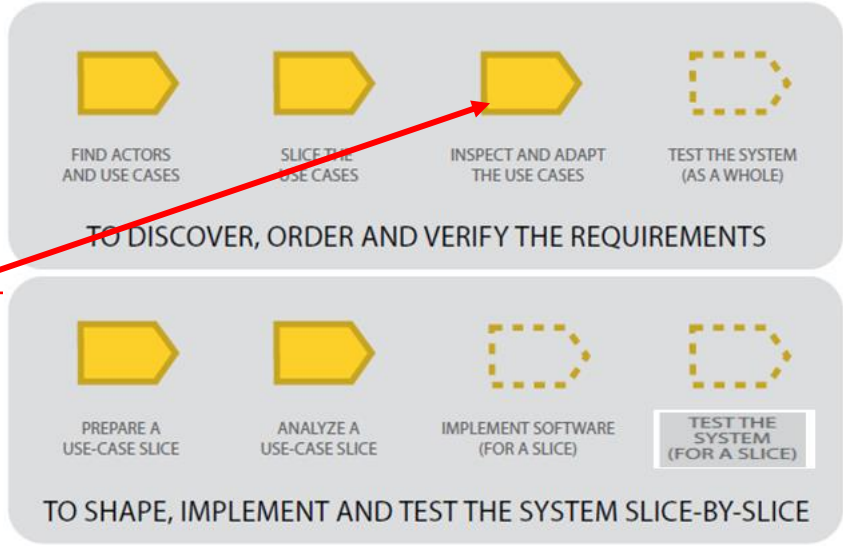
Obs. Variantă de TDD, deoarece cazurile de testare au fost definite în avans

6. *Testare sistem (în ansamblu)*

- Testare increment pentru compatibilitate elemente componente și inexistență regresii.

OBS. Cazurile de testare UC 2.0 sunt robuste și reziliente deoarece structura narativului rezultă în cazuri de testare bazate pe scenarii și executabile independent.

UC 2.0 : Things to do



Procedură (cont.):

7. *Inspectare și adaptare UCs*

- Manipulare modificări (priorități, lecții învățate, descoperire noi UC, refactorizare UC existente)
- Urmărire progres
- Corelare cu timp și buget alocat
- Menținerea actualizată a modelului
- Reglare :
  - dimensiune slice pentru a crește throughput (debitul)
  - nivel de detaliu al descrierii, al definițiilor suport, al cazurilor de testare

Prin ordonarea slice-urilor și urmărirea stării acestora (scoped, prepared, analyzed, implemented, verified) se poate crea un **backlog al cerințelor rămase de implementat**.

Acesta trebuie monitorizat și ajustat continuu pentru a reflecta progresul echipei și dorințele stakeholderi-lor.

Pe măsură ce proiectul avansează, se poate ajusta dimensiunea slice-urilor pentru a elimina risipa și a crește eficacitatea.

Utilizare UC 2.0 : toate tipurile de sisteme

Tipic – pentru sisteme intens interactive

DAR : UC descriu colaborarea sistemului cu contextul său, deci sunt potrivite și pentru sisteme embedded.

Tipic pentru sisteme software

DAR : Se poate aplica recursiv pentru modelarea afacerii și a interacțiunii cu lumea (business use cases)

$\Downarrow$

Contextul pentru modelarea sistemelor necesare susținerii și creșterii afacerii.

Modelul business poate fi, de asemenea, modelat în slice-uri $\Rightarrow$ dezvoltare incrementală, în paralel, atât a afacerii cât și a sistemelor suport.

Utilizare UC 2.0 : toate abordările de dezvoltare

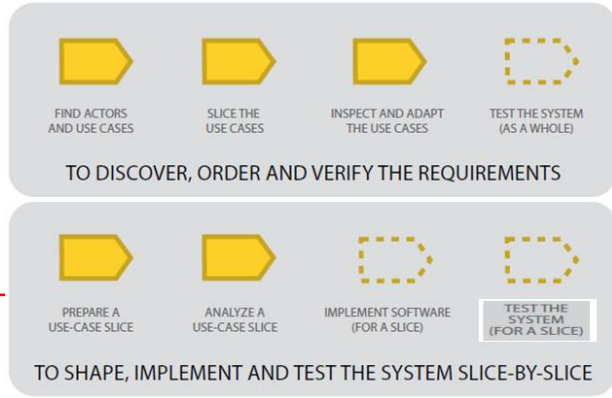
Dezvoltare iterativă (backlog- driven; ex. Scrum, EssUP and OpenUP)

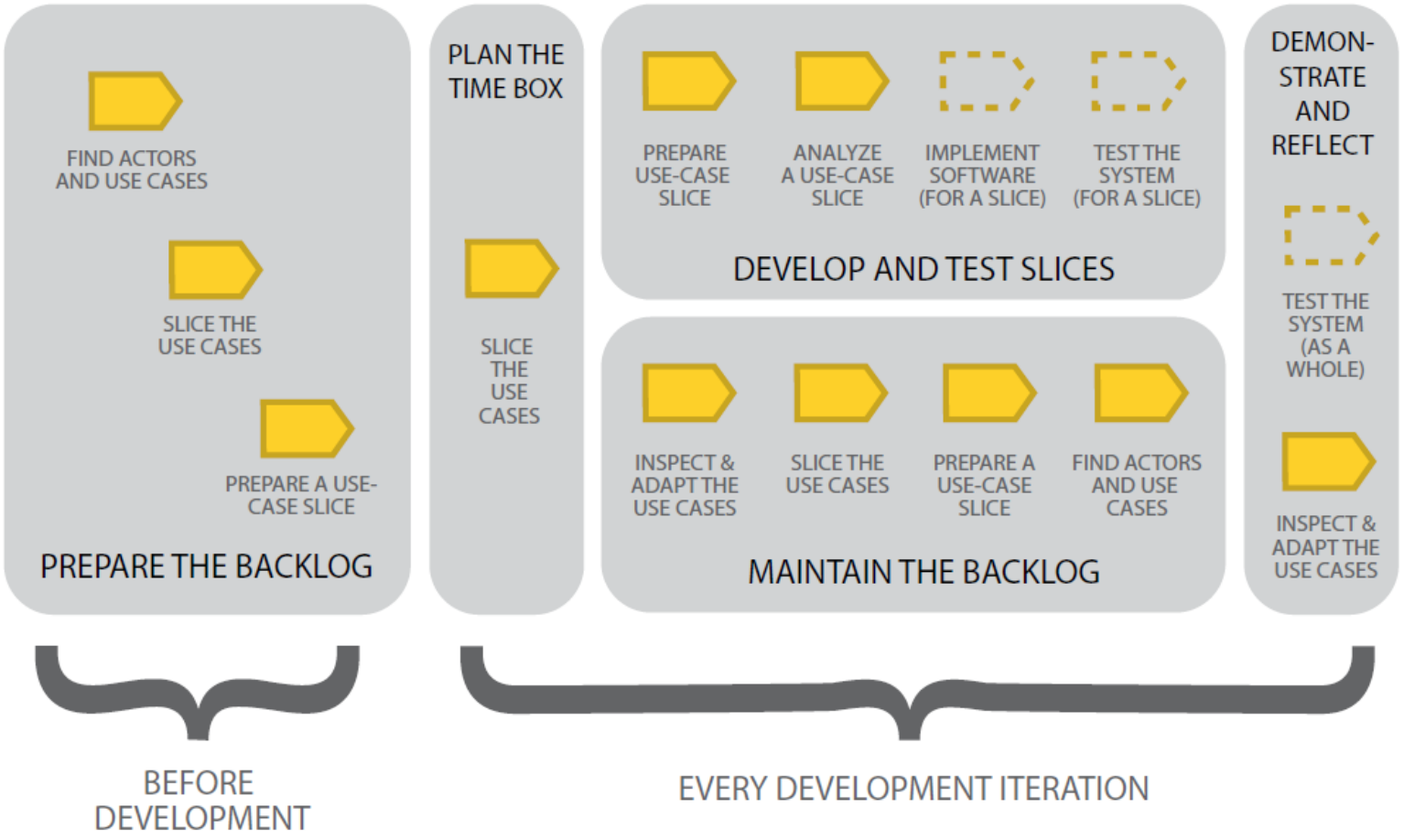
Dezvoltare în flux continuu (bandă de asemblare) (ex. Kanban)

Dezvoltare Waterfall

UC 2.0 : Things to do

Dezvoltare iterativă





UC slice –

- element primar din backlog,
- independent,
- cu valoare,
- testabil,
- estimabil,
- negociabil,
- de dimensiune convenabilă.

Backlog – adaptat și rafinat permanent (grooming).

FIGURE 17: USE-CASE 2.0 ACTIVITIES FOR ITERATIVE DEVELOPMENT APPROACHES

UC 2.0 : Dezvoltare în regim de bandă de asamblare (flux continuu)

UC slice – dimensiune suficient de mică pentru a permite lucrul în regim de bandă de asamblare.

Trebuie să fie disponibile suficient de multe UC slice-uri pentru a asigura fluxul pe banda de asamblare (limita indicată cu roșu).

Personalul se poate redistribui funcție de încărcarea din fiecare etapă.

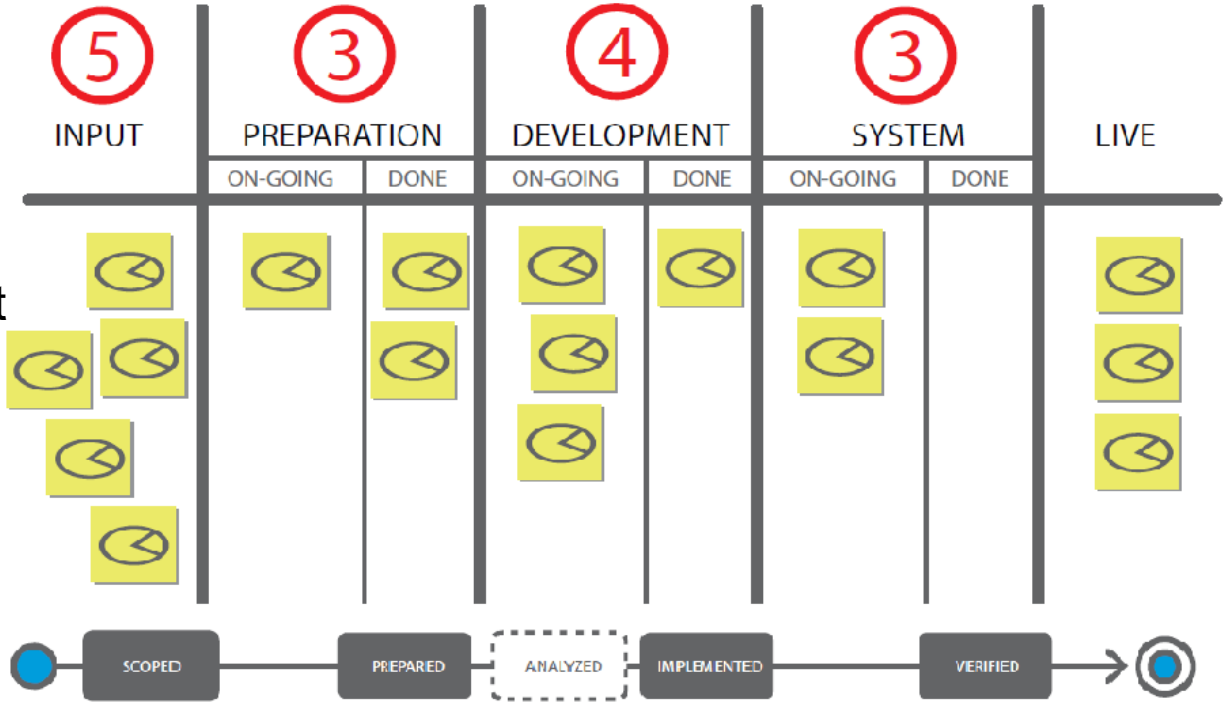
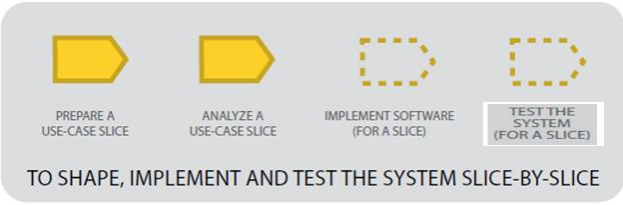


FIGURE 21: ALIGNING THE STATES OF THE USE-CASE SLICE TO THE KANBAN BOARD

UC 2.0 : Things to do

Dezvoltare în regim de bandă de asamblare (flux continuu)



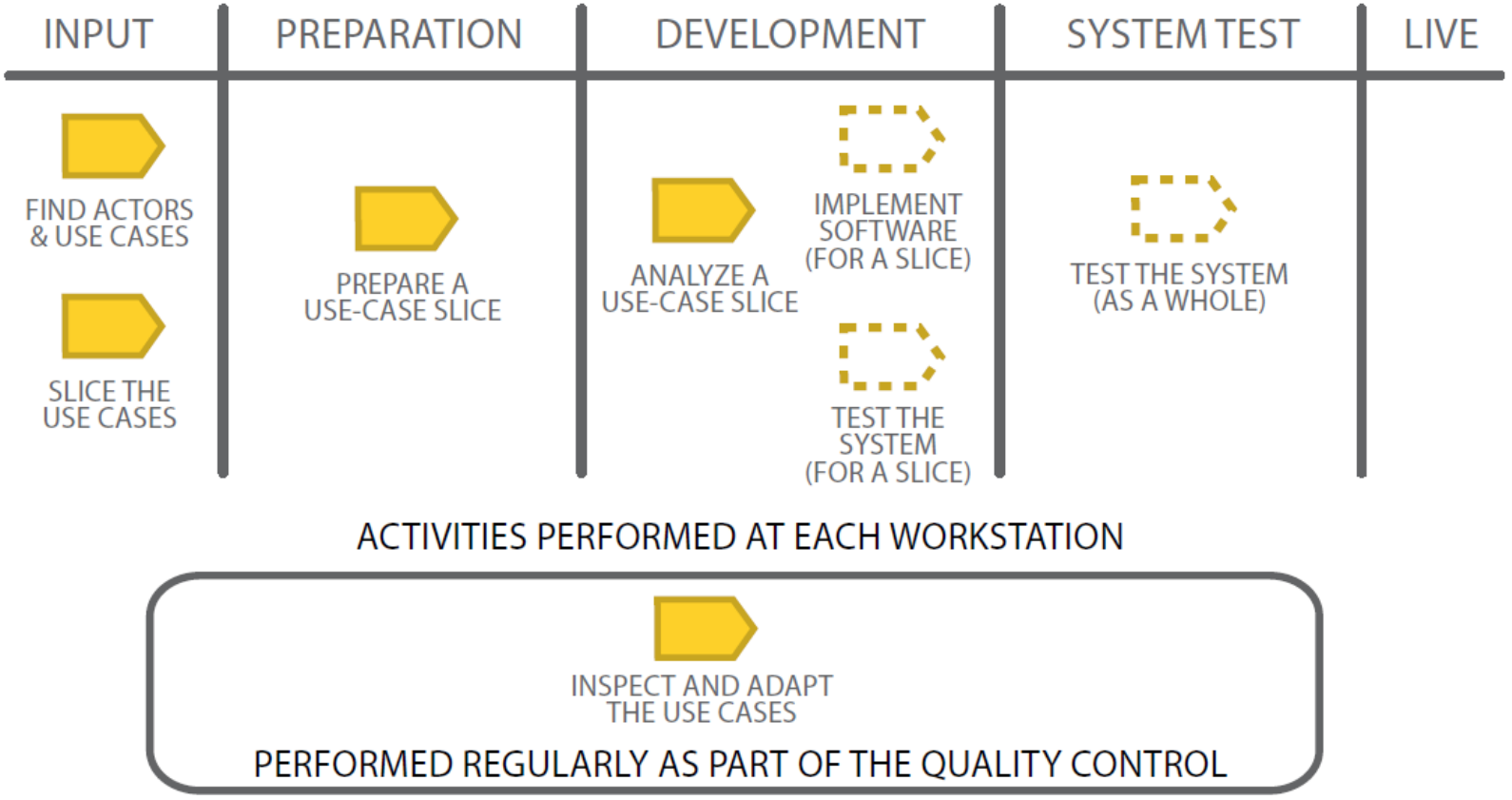


FIGURE 22: USE-CASE 2.0 ACTIVITIES FOR WATERFALL APPROACHES

ONE-PIECE FLOW

UC 2.0 : Things to do

Dezvoltare waterfall

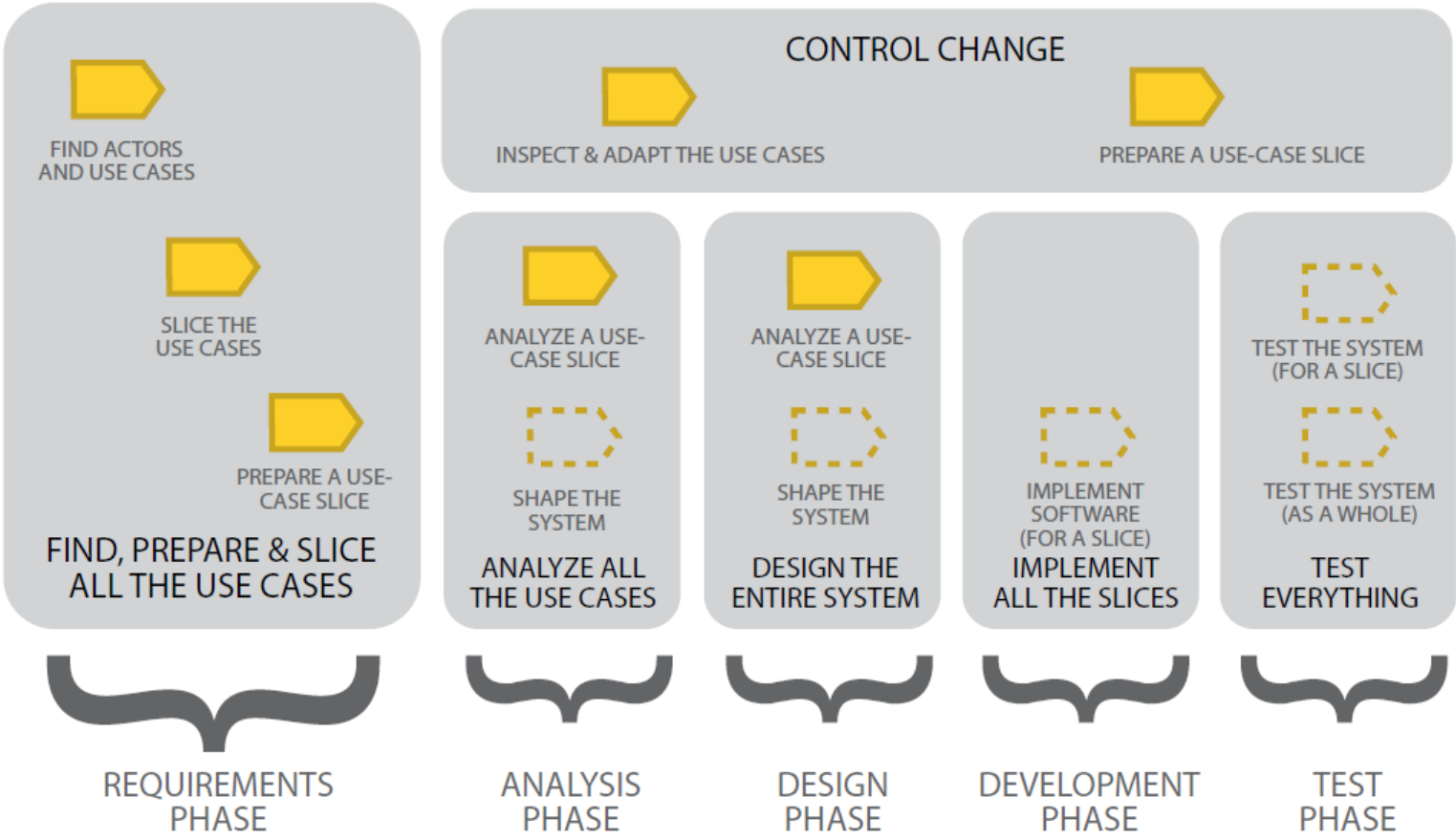


FIGURE 24: USE-CASE 2.0 ACTIVITIES FOR WATERFALL APPROACHES

UC 2.0

Dezvoltare waterfall

Requirements
Analysis and Design
Implementation
Testing

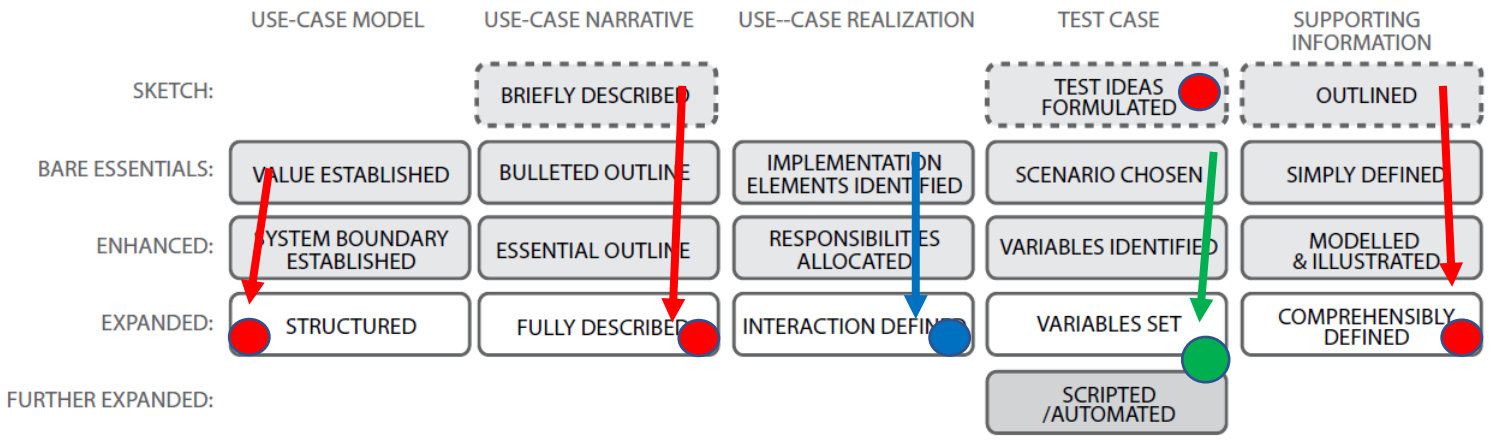


FIGURE 25: LEVELS OF DETAIL FOR THE WORK PRODUCTS USING A WATERFALL APPROACH

UC slices :

- oferă control foarte fin, ceea ce permite clarificarea domeniului acoperit de aplicație.
- ajută la gestionarea modificărilor de ultimă oră din domeniul acoperit de aplicație, rezultate din graficul de timp sau de probleme de calitate.

CONCLUZII

USE-CASE 2.0 - Practică esențializată (bine definită, conform standardului Essence)

‘things to work with’ și ‘things to do’ *nu* sunt negociabile.

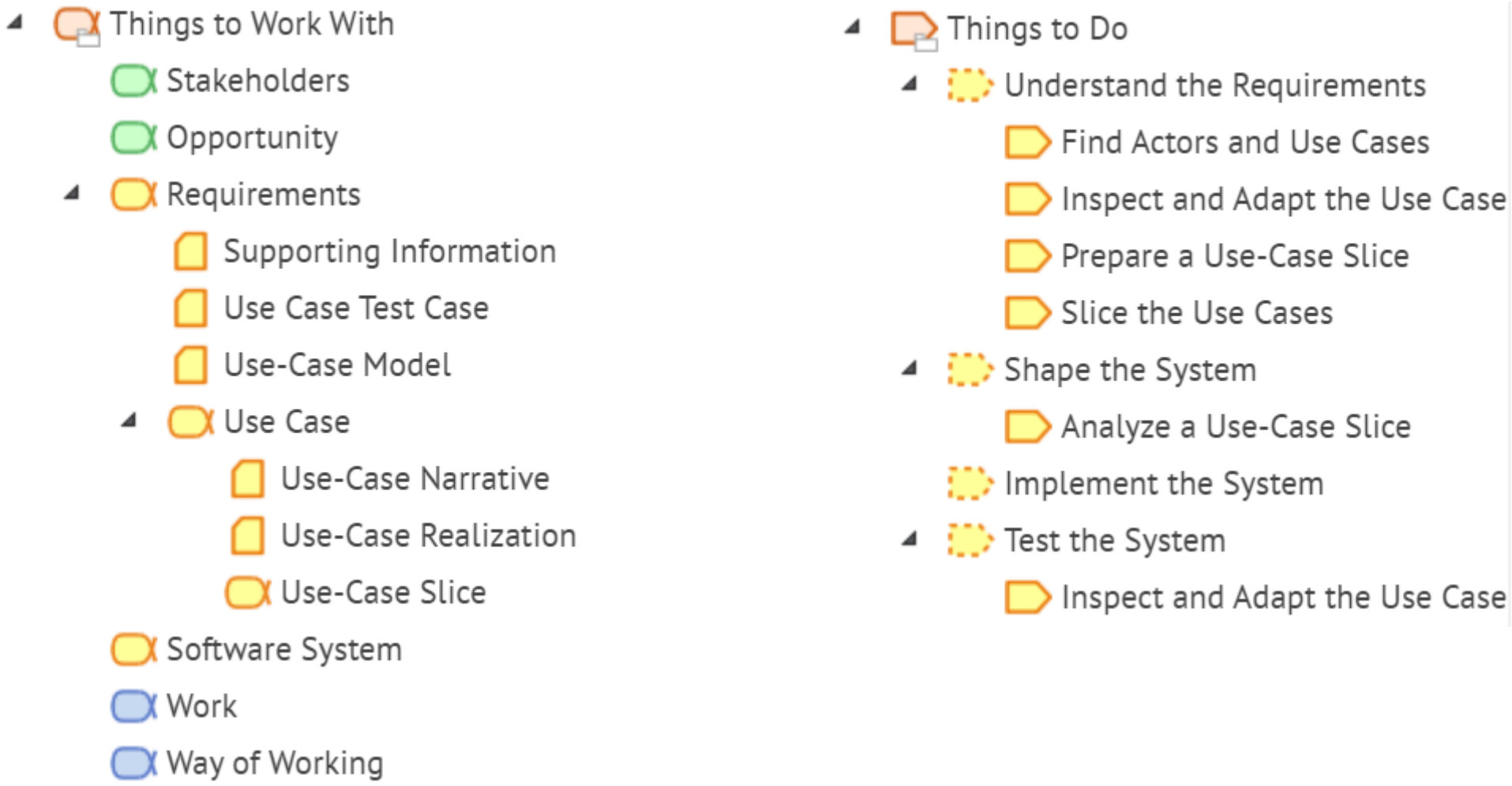
Caracteristici :

- Simplă – atât ca definiție cât și ca aplicare.
- Scalabilă – potrivită pentru echipe și pentru sisteme de orice dimensiune
- Versatilă – toate tipurile de sisteme și de abordări de dezvoltare
- Ușor de folosit – modelele rapid puse în practică, slice-urile create conform nevoilor echipelor

Compatibilă cu multe alte practici :

- Continuous Integration,
- Intentional Architecture,
- Test-Driven Development
- toate practicile de management populare:
  - Lejeritatea și flexibilitatea - utile în manieră agilă
  - Completitudinea și rigoarea - necesară lucrului în contexte mai formale sau de tip watterfall.

‘things to work with’ și ‘things to do’ *nu* sunt negociabile.



https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

Bibliografie

https://www.ivarjacobson.com/publications/white-papers/use-case-20-e-book

https://queue.acm.org/detail.cfm?id=2912151

https://www.ivarjacobson.com/resources?subject=276

Use Case 2.0 Essentials

https://practicelibrary.ivarjacobson.com/content/use-case-20-essentials-publication

The cards

https://practicelibrary.ivarjacobson.com/sites/default/files/embedded-practices/Use_Case_2.0_Essentials_HTML_guidelines6/html/_TbRKwSwHEeGlJawYgZSHHw/Use%20Case%202.0%20Essentials_card.html