



EL PROCÉS DEL PROGRAMARI

Tema 2

Enginyeria del Programari

ETS Enginyeria Informàtica

DSIC – UPV

Objectius

- Definir el terme “Procés del Programari”
- Presentar els principals models de procés de desenvolupament proposats al llarg de la història
- Introduir la noció de metodologia








Continguts

1. Introducció. El Procés del Programari
 2. Cicles de Vida
 - Clàssic o en Cascada
 - Clàssic amb Prototipat
 - Programació Automàtica
 - Incremental
 - Espiral
 3. Metodologies
-




Annex:

RUP
Metodologies Àgils

Bibliografia bàsica

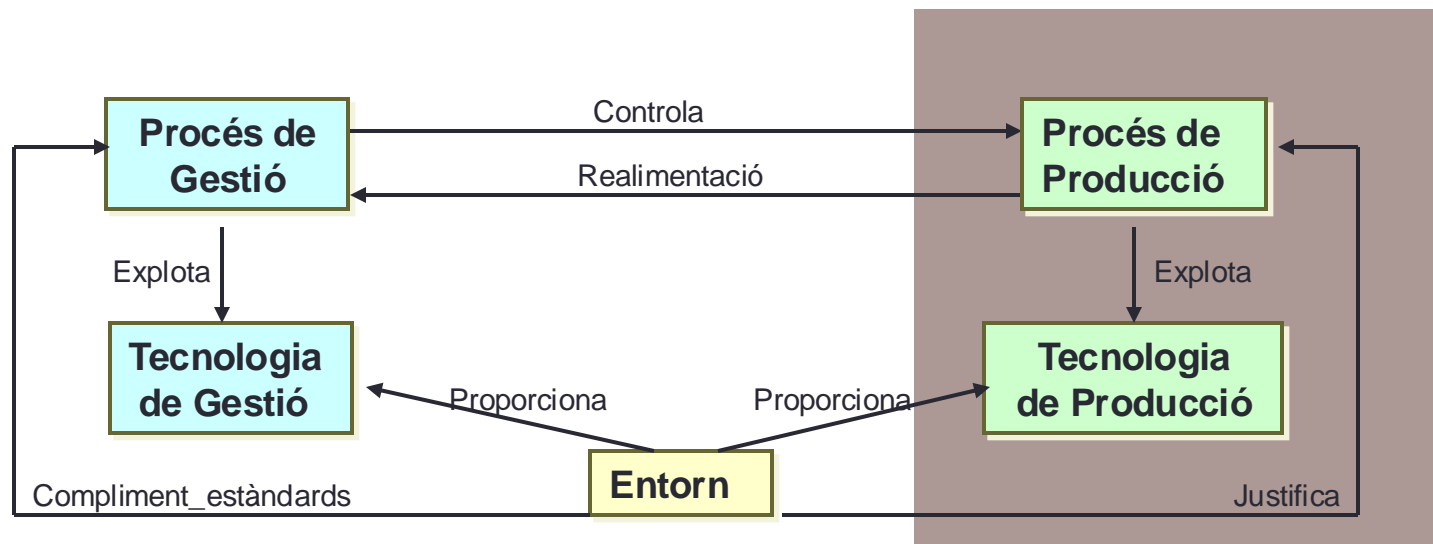
-  Sommerville, I. Ingeniería del Software. (8ª ed.). Addison-Wesley, 2008
-  Presman, R.S., Ingeniería del Software: un enfoque práctico (6ª ed.), McGraw-Hill, 2005
-  Royce, W.W., Managing the Development of Large Software Systems: Concepts and Techniques. Proc WESCON, 1970
-  Agresti, W.W. Tutorial: New Paradigms for Software Development. IEEE Computer Society Press, 1986
-  Balzer, R., Cheatman, T.E. and Green, C., Software Technology in the 1990's: Using a New Paradigm, IEEE Computer, Nov. 1983, pp. 39-45
-  McDermid, J. And Rook, P., Software Development Process Models. Software Engineer's Reference Book, CRC Press, 1993
-  Boehm, B.W.. A Spiral Model of Software Development and Enhancement, IEEE Computer, pages 61-72, May 1988.

Bibliografia metodologies

-  Krutchen, P., *The Rational Unified Process- An Introduction*. Addison – Wesley, 1998
-  Jacobson, G. Booch and J. Rumbaugh., *The Unified Software Development Process*, Addison-Wesley, 1999
-  Beck, K., *Extreme Programming Explained: Embrace Change*. The XP Series. Addison-Wesley, 2000

El Procés del Programari

- Estableix un marc per al desenvolupament de programari
- En general el terme “Procés del Programari” s'associa al *procés de producció*.... però inclou també el *procés de gestió*



El Procés de Desenvolupament

- Conjunt d'activitats la meta de les quals és el desenvolupament o evolució de programari

- Conegut també com a **Cicle de Vida**



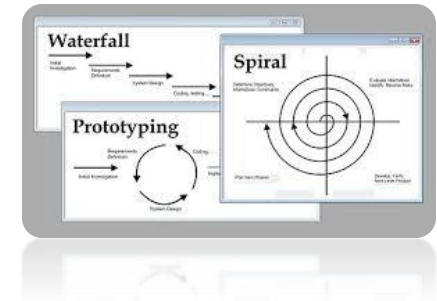
- **Activitats** genèriques que apareixen sempre:

- Especificació
- Desenvolupament
- Validació
- Evolució

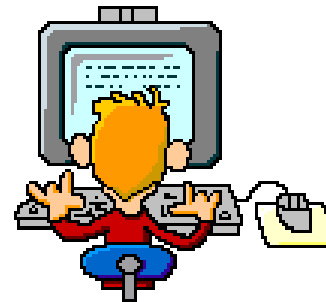
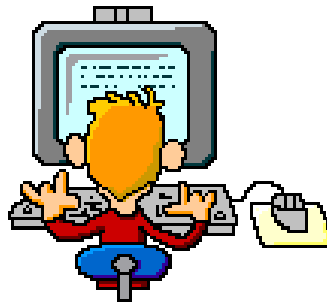
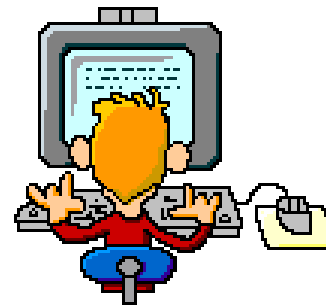
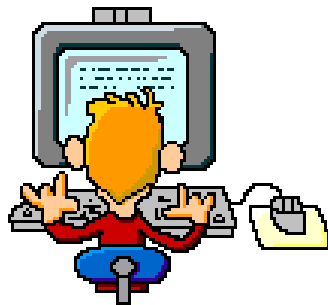
- Anàlisi
- Disseny
- Implementació
- Proves
- Manteniment

Models de Cicle de Vida

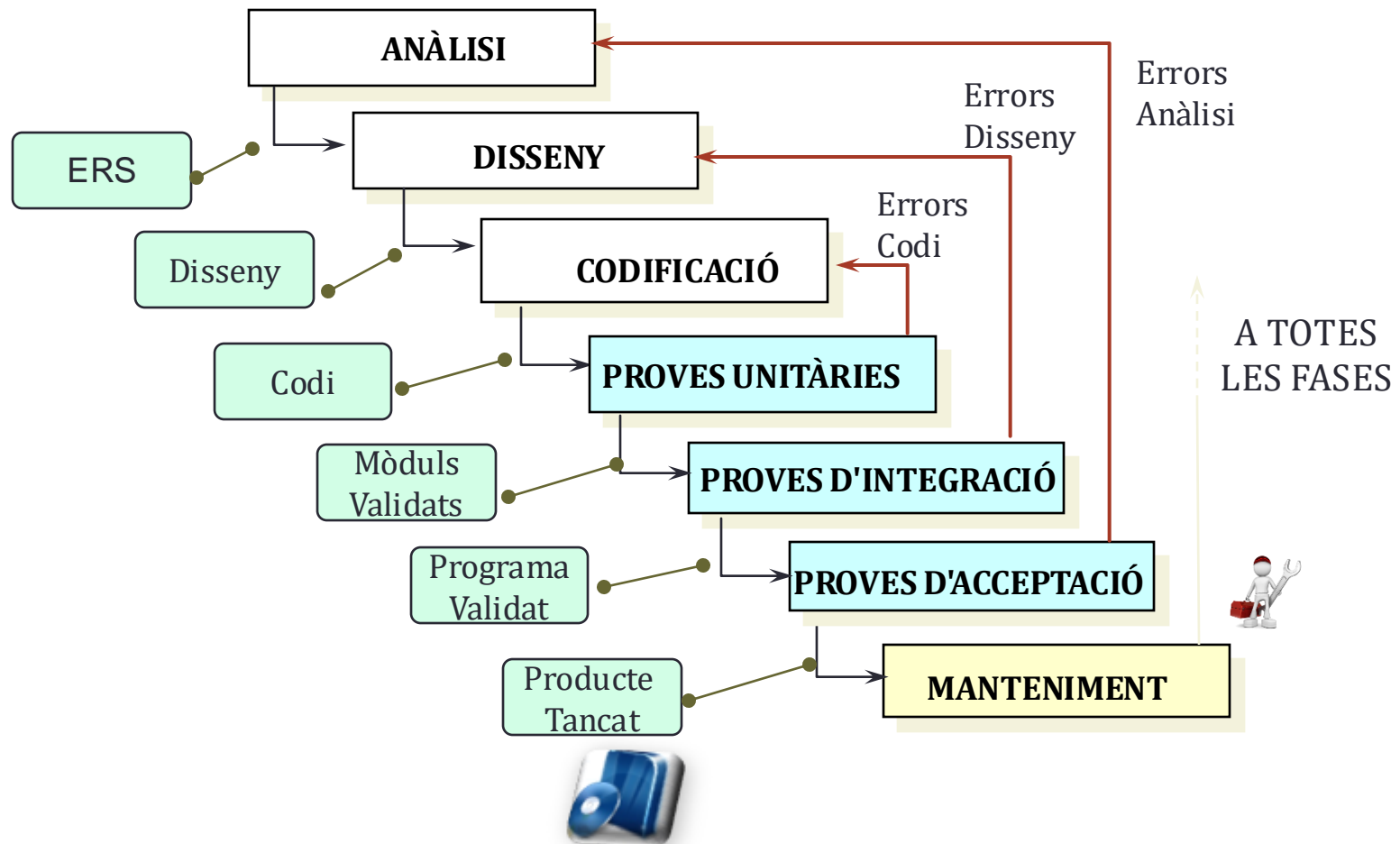
- Codificar i corregir (*code-and-fix*)
- Clàssic o en cascada
- Clàssic amb prototipat
- Programació Automàtica
- Models evolutius:
 - Incremental
 - En espiral



Codificar i Corregir

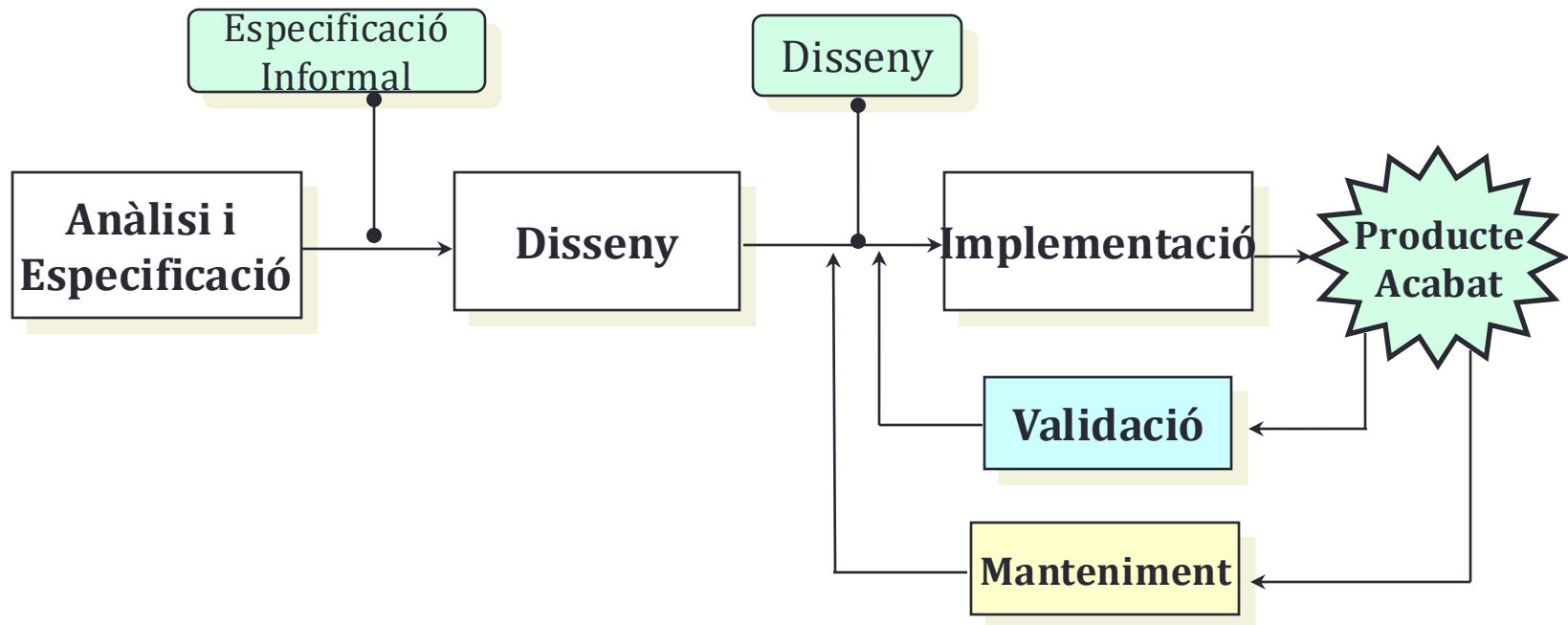


Model Clàssic o en Cascada



Model Clàssic o en Cascada

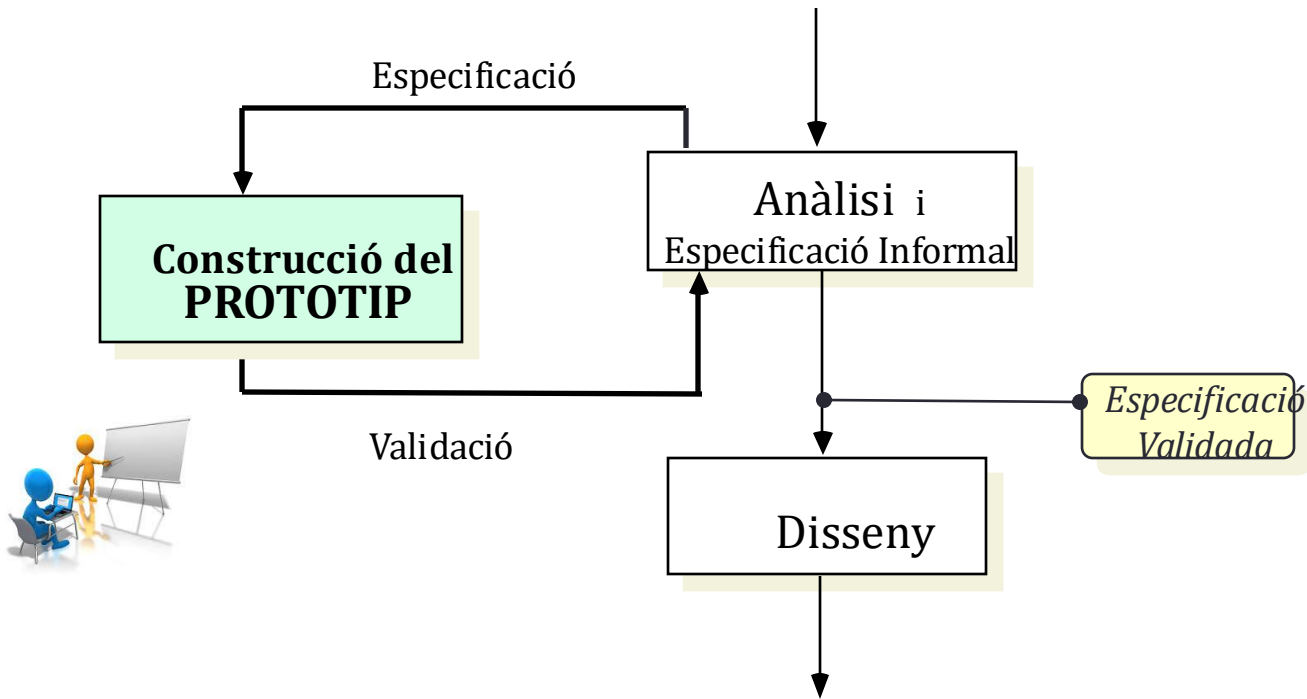
- En la pràctica el model tendeix a *deformar-se*, i tot el pes de la validació i manteniment recau, en la seua major part, sobre el *codi font*.



Model Clàssic amb Prototipat

- **Prototip:** Primera versió d'un producte, en el qual s'han incorporat només algunes característiques del sistema final, o no s'ha realitzat completament
- **Classes de Prototips:**
 - Vertical: desenvolupa completament alguna de les facetes del programa.
 - Horitzontal: desenvolupa en part totes les facetes

Model Clàssic amb Prototipat



- Ajuda als clients a establir clarament els requisits
- Ajuda als desenvolupadors a millorar els seus productes

Model Clàssic amb Prototipat

- Crítica:

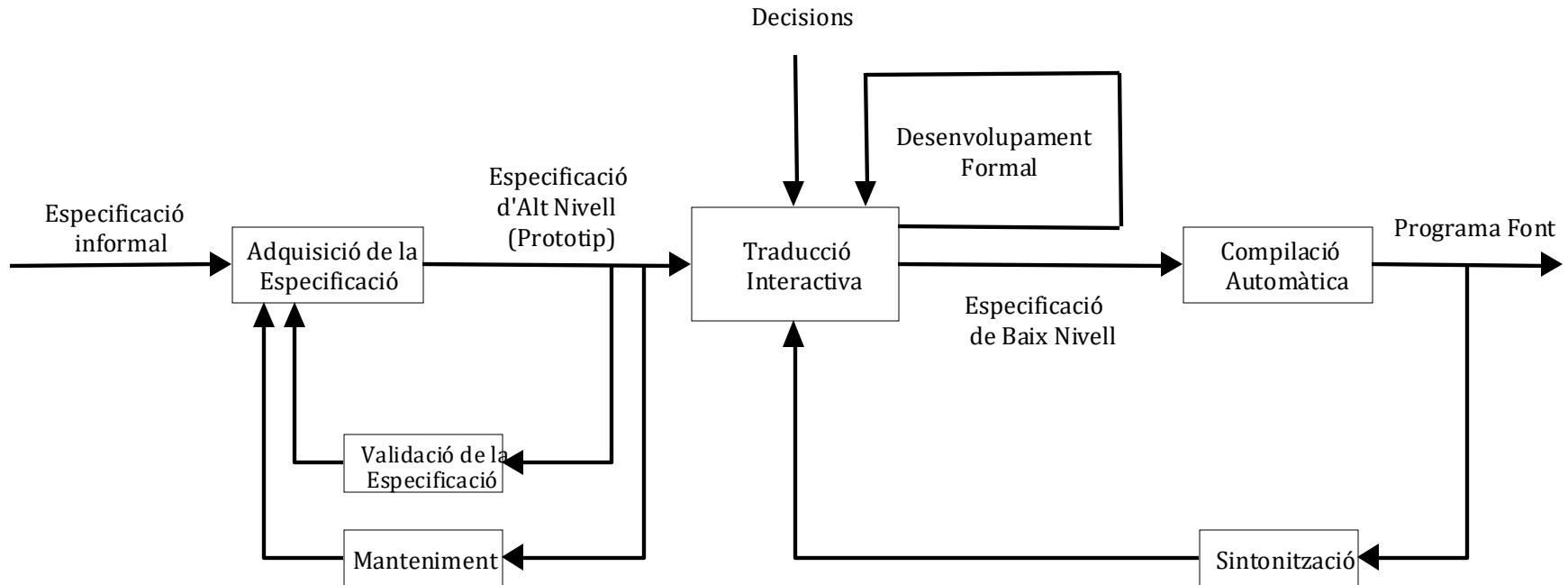
- ☺ Redueix el risc de “*parches*” sobre el producte final (Encara que no elimina el manteniment sobre el codi)
- ☺ Ajuda a entendre els requisits tant a clients com a desenvolupadors
- ☹ El client veu una versió del que serà el programa final, sense assumir que no és robusta ni completa
- ☹ Suposa una inversió addicional que pugues no ser rendible; a més, el temps invertit pot fer que el producte perdi oportunitat
- ☹ És freqüent arrossegar males decisions que només eren apropiades per a l'obtenció ràpida del prototip

Model de Programació Automàtica

(R. Balzer, 1983)

- Objectiu
 - Introduir l'automatització en el procés de construcció del programari
- Característiques bàsiques:
 - ✓ Ús de llenguatges formals d'especificació
 - ✓ L'especificació és un prototip del producte
 - ✓ Els requisits es perfilen animant l'especificació
 - ✓ El programa es deriva (semi)automàticament

Model de Programació Automàtica



Model de Programació Automàtica

- Comparació

CLÀSSIC Prototipat

- Especificació informal
- Prototipat no usual
- El prototip es crea manualment
- El prototip es rebutja
- Implementació manual
- El codi ha de provar-se
- Manteniment sobre el codi

P. AUTOMÀTICA

- Especificació formal
- Prototipat estàndard
- L'especificació és el prototip
- Evoluciona cap al producte final
- Implementació automàtica
- Sense proves
- Manteniment sobre l'especificació

Model de Programació Automàtica

- Crítica
 - ☺ Ajuda a reduir errors humans
 - ☺ Redueix el cost de desenvolupament
 - ☹ Dificultat d'ús dels llenguatges formals

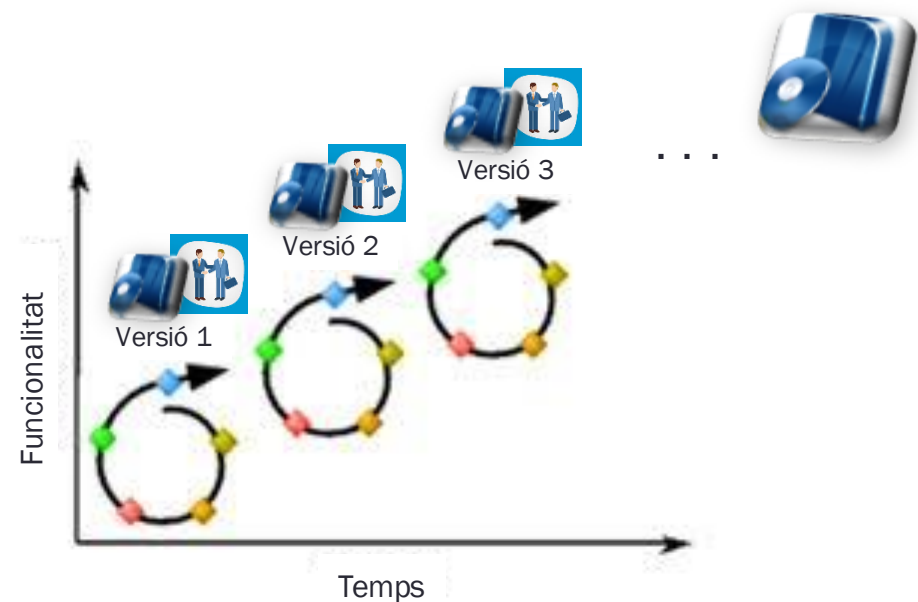
➡ *És l'antecessor de MDE - MDA*

Desenvolupament Evolutiu

- Adaptable a requisits canviants
- S'elaboren versions cada vegada més completes del programari

➡ Model incremental

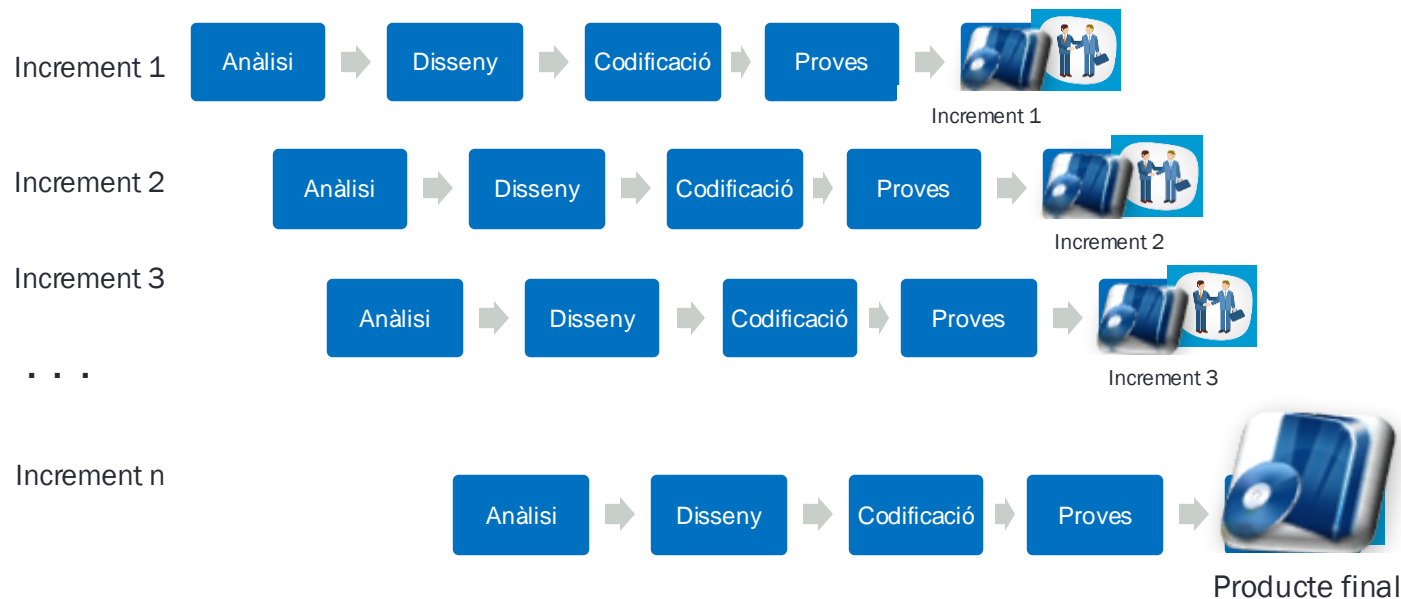
➡ Model en espiral



Model Incremental

(McDermind, 1983)

- Seqüència d'aplicacions del cicle clàssic
- Cada iteració produeix un increment del producte
- Finalitza quan es lliura el producte final



Model Incremental

- Crítica

😊 Útil quan no es disposa de personal per a una implementació completa

😊 Cada lliurament pot ser avaluada per l'usuari → alta interactivitat

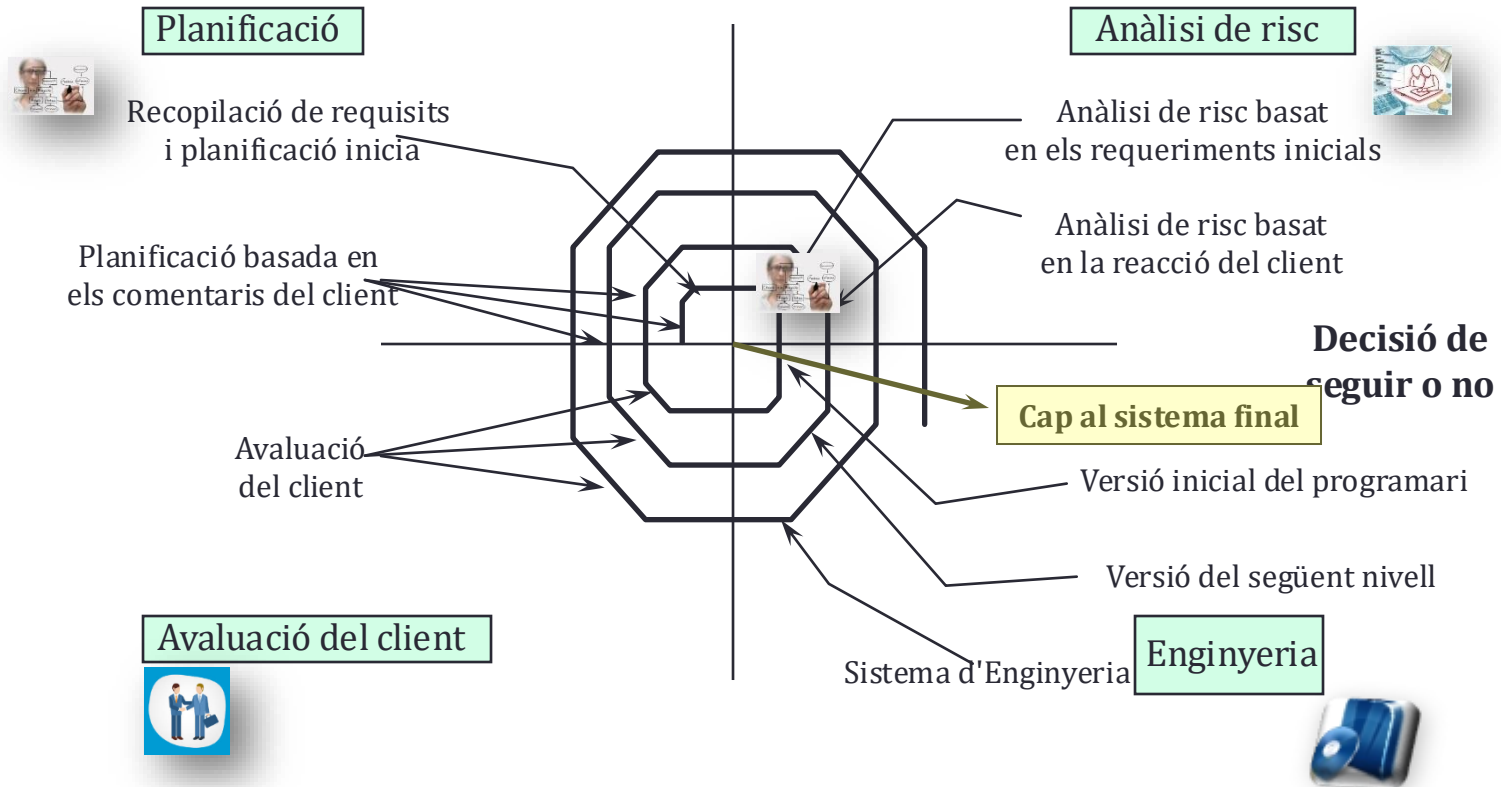
😞 Dificultat de determinar l'increment requerit en cada iteració

Model En Espiral

(B. Boehm, 1988)

- Enfocament:
 - Iteratiu.
 - Interactiu.
 - Evolutiu
- Introdueix l'anàlisi de riscos en el procés de desenvolupament

Model En Espiral

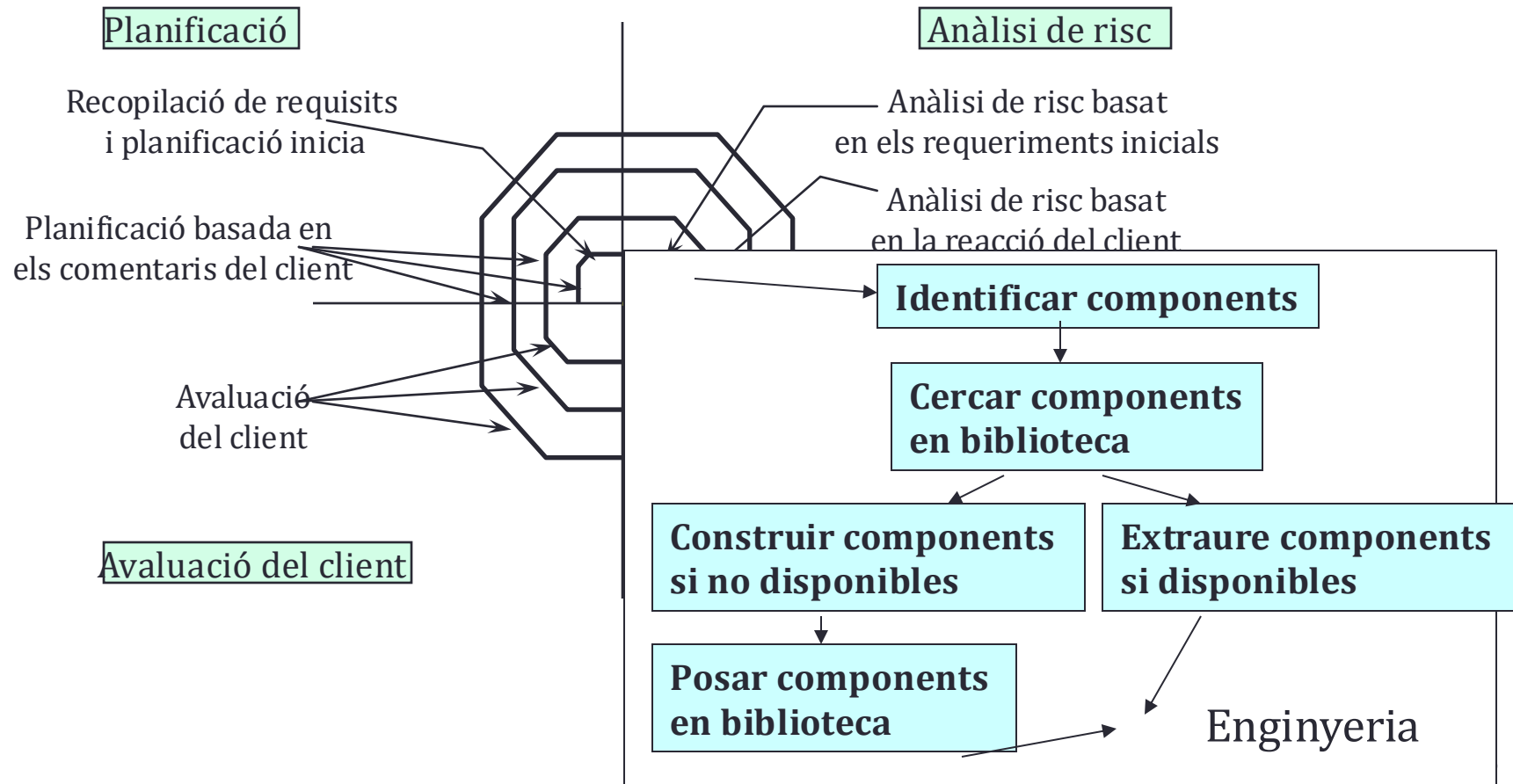


Model En Espiral

- Crítica
 - 😊 Cada vegada s'obtenen versions més completes del producte.
 - 😊 Cada versió és avaluada per l'usuari → alta interactivitat
 - 😞 Dificultat a avaluar els riscos
 - 😞 Assegurar que s'avança cap al producte final

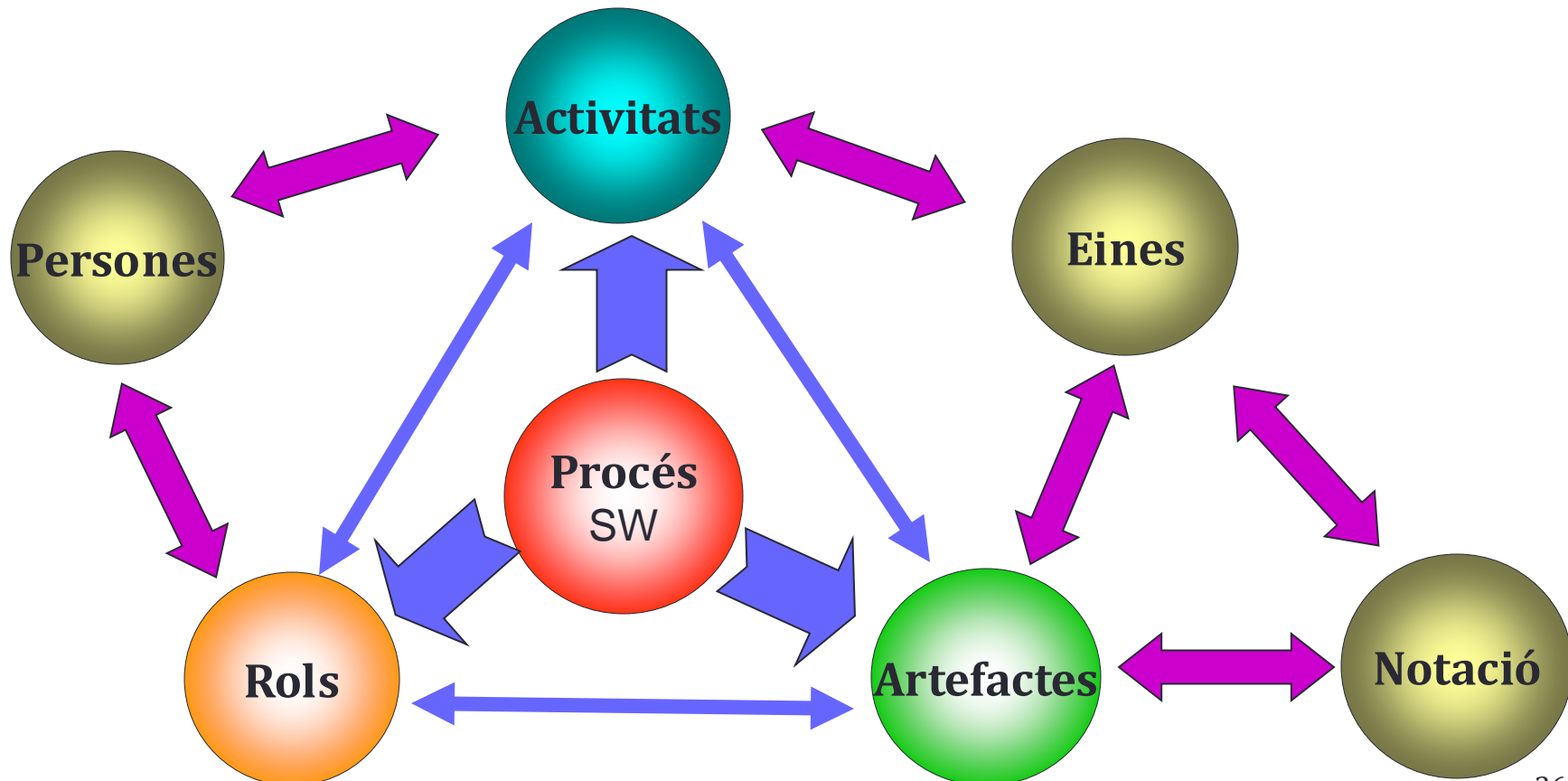
Model d'Assemblatge de Components

- El model en espiral es pot adaptar la fase d'enginyeria a noves propostes.



Metodologia

- En un projecte de desenvolupament de programari, la metodologia defineix: **Qui** / **Qué** / **Com** / **Quan**



Metodologia

- Defineix un procés explícit de desenvolupament de programari

(el seu objectiu és la formalització de les activitats relacionades amb l'elaboració de sistemes informàtics)

- Aquest procés ha de ser:
 - Reproduïble
 - Definit
 - Mesurable quant a rendiment
 - Optimitzable
 - ...

Metodologia

No existeix una metodologia de programari universal.

Metodologies estructurades

Metodologies orientades a objectes

RUP

Metodologies Tradicionals vs. *Metodologies Àgils*

RUP

...

XP

Kanban

SCRUM ...

ANNEX:

Metodologies

- . (RUP) Rational Unified Process
- . Metodologies Àgils / Pràctiques Àgils

Procés Unificat de Rational (RUP)



Procés de Desenvolupament de Programari
(Rational – IBM)

Utilitza **UML**, com a llenguatge de modelatge

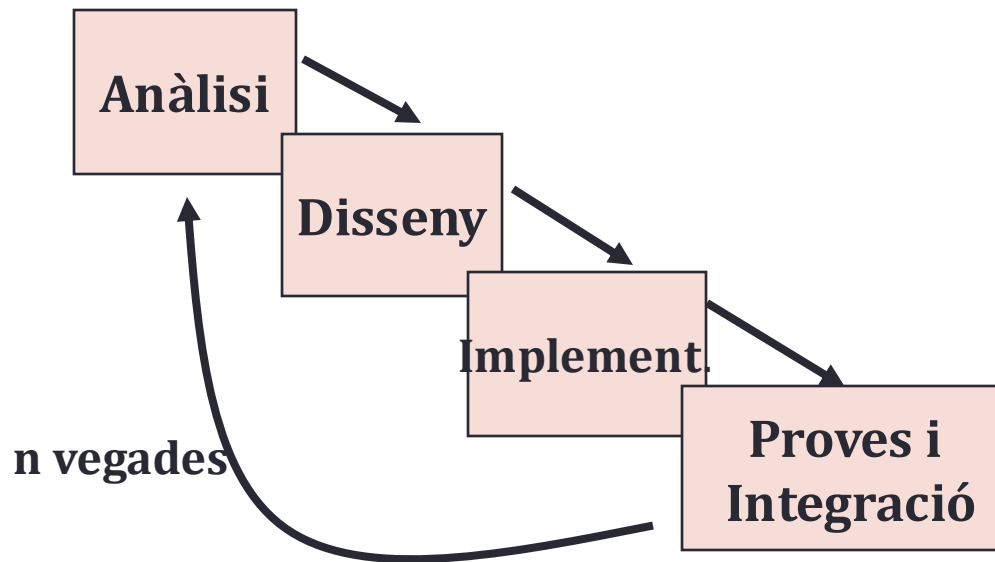
Característiques:

- *Procés Dirigit pels Casos d'Ús*: des de l'especificació fins al manteniment
- *Procés Iteratiu i Incremental*: les iteracions en funció de la importància dels casos d'ús i l'estudi de riscos.
- *Procés Centrat en l'Arquitectura*: reutilitzable i com a guia fins a la solució

RUP

- Iteratiu i Incremental

Les activitats es realitzen en una mini-cascada amb un abast limitat pels objectius de la iteració

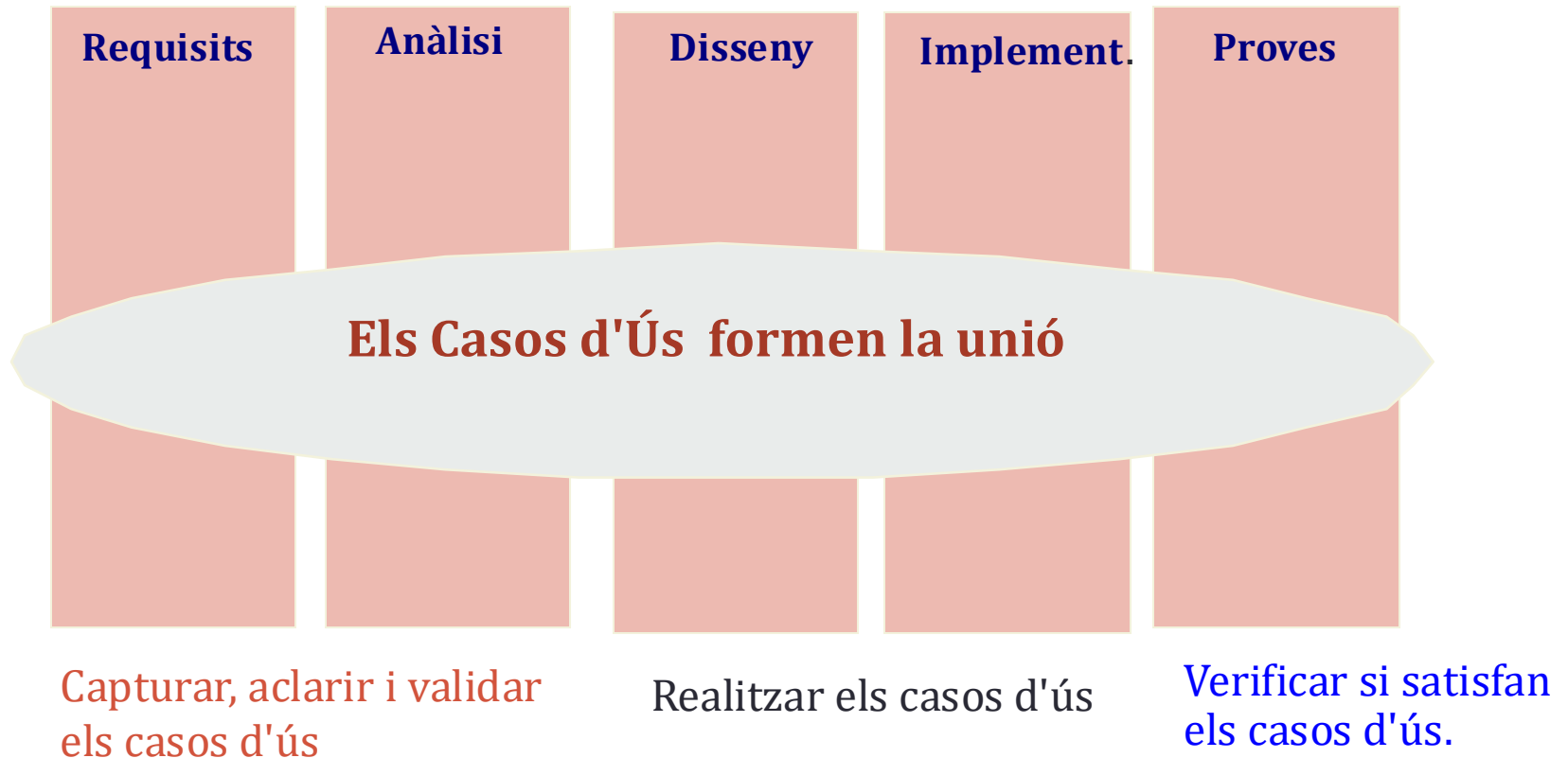


ACTIVITATS DE LA ITERACIÓ

- Planificar la iteració (riscos)
- Anàlisi de **Casos d'Ús** i Escenaris
- Dissenye Opcions Arquitectòniques
- Implementació
- Proves
- Integració
- Avaluació del lliurament
- Preparació del lliurament

RUP

- Dirigit pels casos d'ús



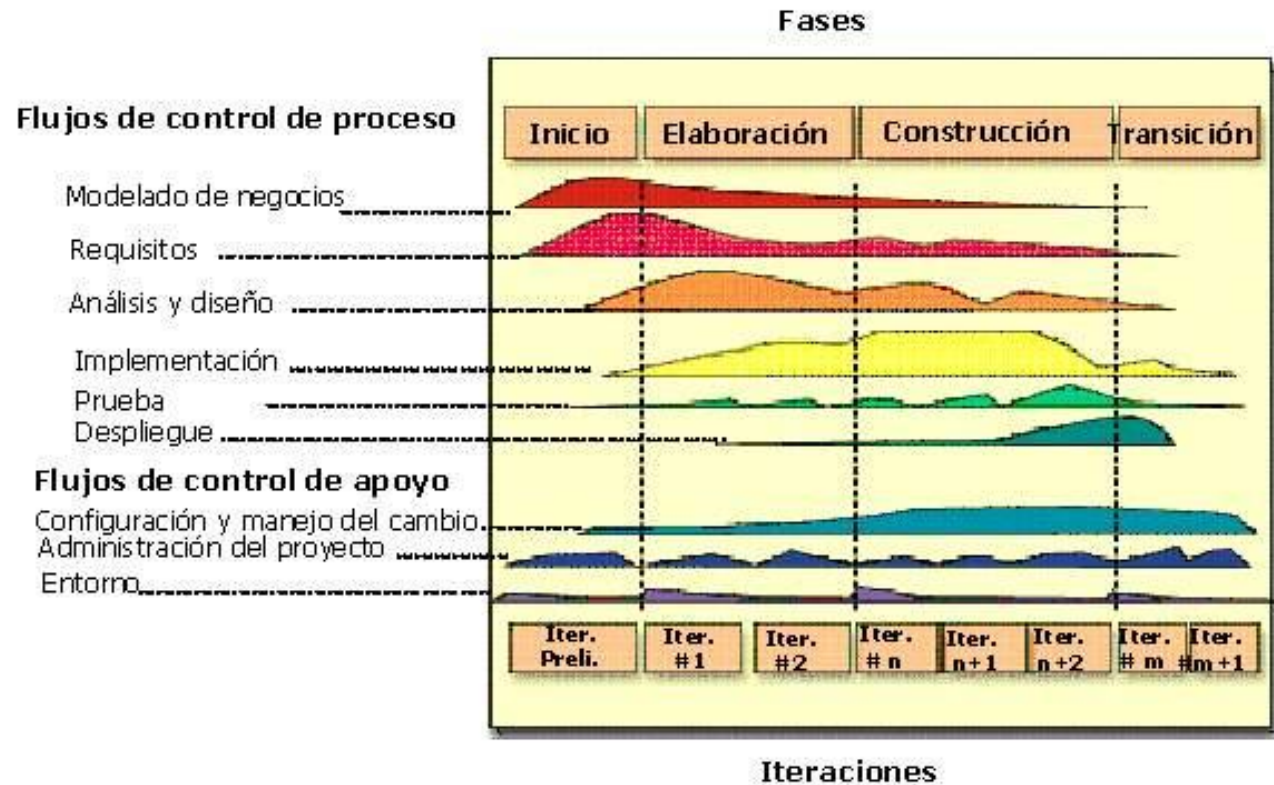
RUP

Vista dinàmica

Vista estàtica

Eje Vertical:
Organización
a lo largo
del contenido

Eje Horizontal: Organización a lo largo del tiempo



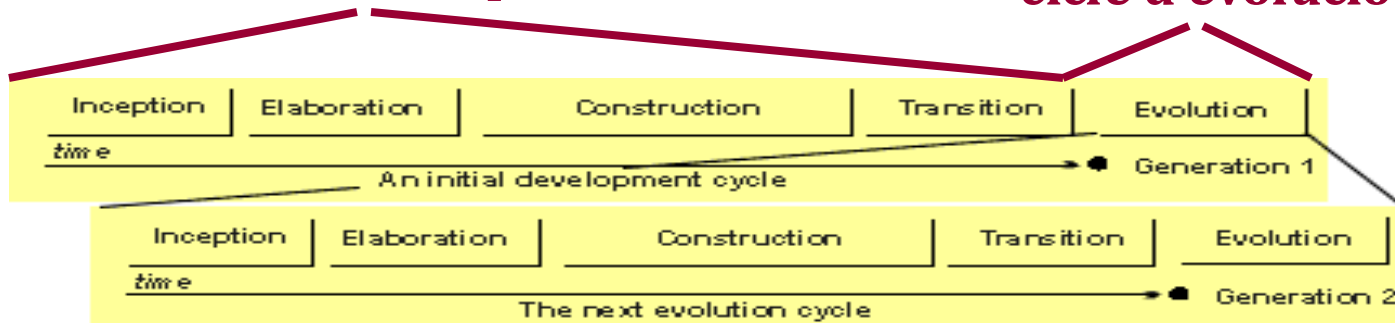
RUP

Vista dinàmica

- Cicles, Fases, Iteracions i Lliuraments

cicle de desenvolupament

cicle d'evolució



Fases



Iteracions

Objectius
(Vision)

Arquitectura

Capacitat
Operacional
Inicial

Producte

Lliuraments

RUP

Vista dinàmica

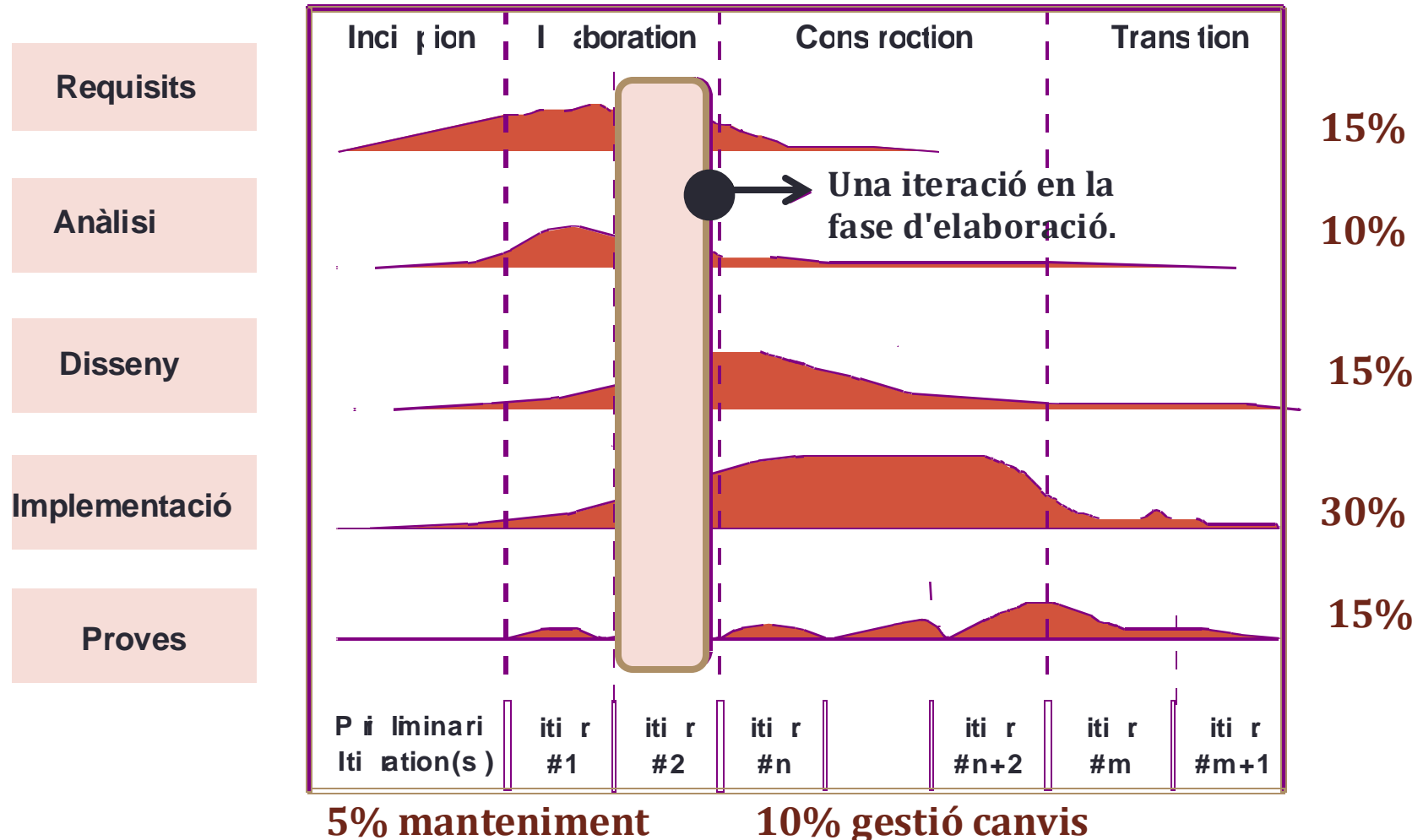
- Fases
 - *Inici (Estudi d'Oportunitat)*
 - Es defineix l'àmbit i objectius del projecte
 - Es defineix la funcionalitat i capacitats del producte
 - *Elaboració*
 - El domini del problema i la funcionalitat desitjada s'estudien en profunditat
 - Es defineix una arquitectura bàsica
 - Es planifica el projecte considerant els recursos disponibles

RUP

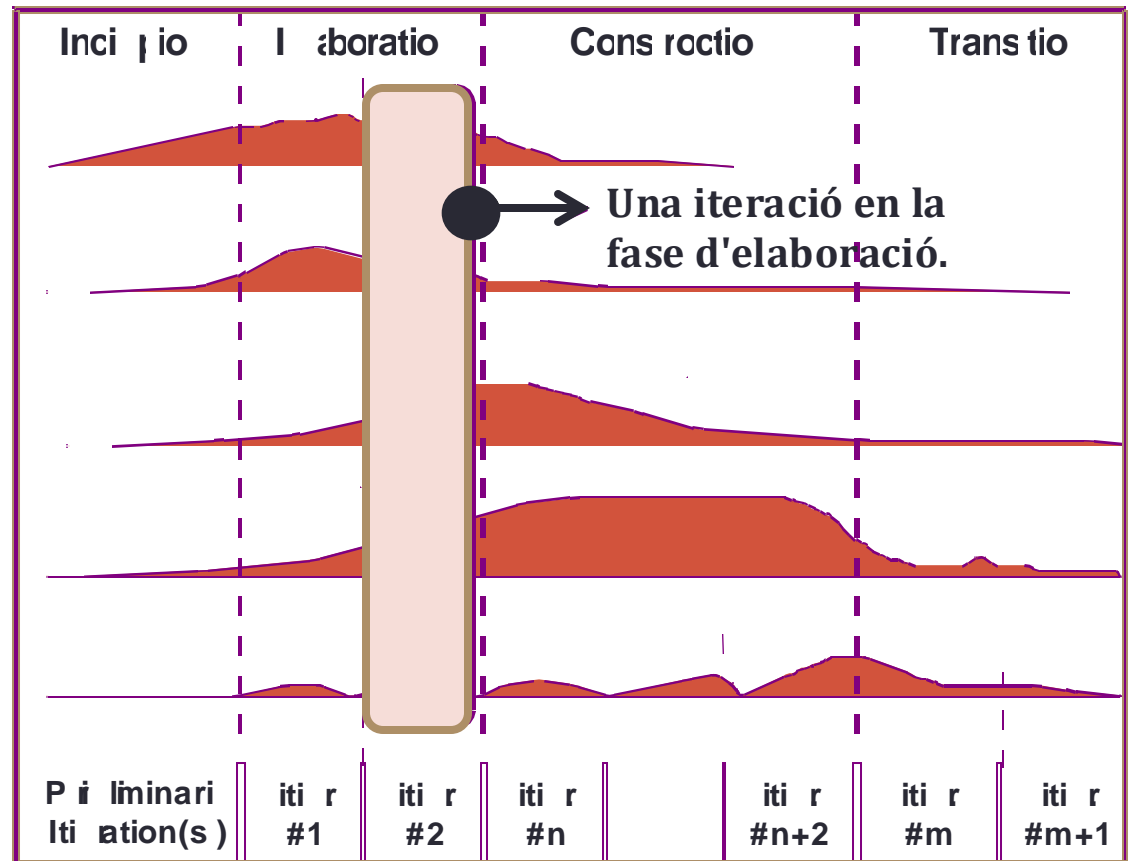
Vista dinàmica

- *Construcció*
 - En cada iteració es realitzen tasques d'anàlisi, disseny i implementació
 - Es refina l'arquitectura
 - Una part important del treball es dedica a programació i proves
 - Es documenta tant el sistema construït com la utilització del mateix
 - Aquesta fase proporciona un producte construït i una documentació
- *Transició*
 - Es lliura a l'usuari per al seu ús real
 - Es realitzen tasques de màrqueting, empaquetat, instal·lació, configuració, entrenament, suport, manteniment, ...
 - Els manuals d'usuari, instal·lació, ... es completen i refinen

Vista dinàmica

RUP - *Distribució d'esforços respecte activitats*

RUP - *Distribució d'esforços respecte fases*



Esforç:

5%

10%

20%

65%

Durada:

10%

10%

30%

50%

RUP

Vista estàtica

- Fluxos de Treball (Workflows)

<i>Flux de Treball</i>	<i>Descripció</i>
Modelatge del Negoci	Els processos del negoci es modelen utilitzant casos d'ús del negoci
Requisits	Es defineixen els actors que interactuen en el sistema i es desenvolupen casos d'ús per a modelar els requisits del sistema
Anàlisi i disseny	Es crea i documenta un model de disseny utilitzant models arquitectònics, models de components, models d'objectes i models d'interacció.
Implementació	S'implementen i estructuren en subsistemes els components del sistema. La generació automàtica de codi dels models de disseny ajuda a accelerar aquest procés.
Proves	Les proves són un procés iteratiu que es duen a terme conjuntament amb la implementació. Quan finalitza la implementació es realitzen les proves del sistema.
Desplegament	Es crea una “ <i>release</i> ” (versió) del producte, es distribueix als usuaris i s'instal·la en el seu lloc de treball.

RUP

Vista estàtica

- Fluxos de Treball (Workflows)

<i>Flux de Treball</i>	<i>Descripció</i>
Configuració i gestió de canvis	Gestiona els canvis en el sistema
Gestió del projecte	Gestiona el desenvolupament del sistema
Entorn	Desenvolupament d'eines programari apropiades per als equips de desenvolupament de programari.

Metodologies Àgils

Les Metodologies Àgils valoren:

- A l'individu i les interaccions en l'equip de desenvolupament més que a les activitats i les eines
- Desenvolupar programari que funciona més que aconseguir una bona documentació \Rightarrow Minimalisme respecte del modelatge i la documentació del sistema
- La col·laboració amb el client més que la negociació d'un contracte
- Respondre als canvis més que seguir estrictament una planificació

<http://www.agilealliance.com>

Metodologies Àgils

Principis de les Metodologies Àgils (1/2)

- 1.- La prioritat principal és satisfer al client mitjançant primerenques i continus lliuraments de programari utilitzable.
- 2.- Donar la benvinguda als canvis. Els processos àgils apliquen els canvis perquè el client siga competitiu.
- 3.- Lliurar el programari desenvolupat freqüentment amb el menor interval de temps possible entre un lliurament i la següent
- 4.- La gent de negocis i els desenvolupadors treballen junts a través d'un projecte
- 5.- Construir projecte espentats per motivacions personals. Donar l'entorn que necessiten les persones i confiar en ells.

Metodologies Àgils

Principis de les Metodologies Àgils (2/2)

- 6.- El diàleg cara a cara és el mètode més eficient i efectiu per a comunicar informació dins d'un equip de desenvolupament
- 7.- Desenvolupar programari és la primera mesura de progrés.
- 8.- Els processos àgils promouen un desenvolupament suportable. Els patrocinadors, desenvolupadors i usuaris són capaços de mantenir una pau constant
- 9.- L'atenció contínua a la qualitat tècnica i al bon disseny incrementa l'agilitat
- 10.- La simplicitat és essencial
- 11.- Les millors architectures, requisits i dissenys sorgeixen de la pròpia organització de l'equip
- 12.- En intervals regulars, l'equip reflexiona en com arribar a ser més efectiu, sincronitzar i ajustar el seu comportament.

Metodologies Àgils

- Comparativa

Metodologia Àgil

El client és part de l'equip de desenvolupament (a més *in-situ*)

Grups xicotets (< 10 integrants)
i treballant en el mateix lloc

Pocs artefactes

Pocs rols

Menys èmfasi en l'arquitectura

Metodologia No Àgil

El client interactua amb l'equip de desenvolupament mitjançant reunions

Grups grans

Més artefactes

Més rols

L'arquitectura és essencial

Metodologies Àgils

- Comparativa

Metodologia Àgil	Metodologia No Àgil
Heurístiques	Rigoroses
Tolerant als canvis	Resistent als canvis
Imposades internament (per l'equip).	Imposades externament
Procés menys controlat, amb pocs principis	Procés molt més controlat, amb nombroses polítiques/normes
No existeix un contracte tradicional o almenys és bastant flexible	Existeix un contracte prefixat

Principals Metodologies Àgils

- ⇒ Extreme Programming (XP) <http://www.extremeprogramming.org>
- ⇒ SCRUM <http://www.controlchaos.com>
- ⇒ Crystal Methods <http://alistair.cockburn.us/Crystal+methodologies>
- ⇒ Adaptive Development Software (ADS) <http://www.adaptivesd.com>
- ⇒ Dynamic Systems Development Method (DSDM) <http://www.dsdm.org>
- ⇒ Feature-Driven Development (FDD)
<http://www.featuredrivendevelopment.com>
- ⇒ Lean Development (LD) <http://www.poppendieck.com>

Extreme Programming (XP)



Kent Beck, Ward Cunningham i Ron Jeffries

www.extremeprogramming.org

www.xprogramming.com

- Dissenyat per a entorns dinàmics
- Pensat per a equips xicotets (≤ 10 programadors)
- Orientat fortament cap a la codificació
- Èmfasi en la comunicació informal, verbal
- Altres valors: simplicitat, realimentació i coratge

XP

Cicle de Desenvolupament

Històries, Iteracions, Versions, Tasques i Casos de Prova

- ✓ El client selecciona la **següent versió** a construir, triant les **característiques funcionals** que considera més valuoses (cridades **Històries**) d'un conjunt possible d'històries, sent informat dels **costos** i del **temps** que costarà la seua implementació.
- ✓ Els programadors **converteixen les històries** en **tasques a realitzar** i a continuació converteixen les **tasques** en **un conjunt de casos de prova** per a demostrar que les tasques s'han finalitzat.
- ✓ Treballant amb un company el programador **executa els casos de prova** i **evoluciona el disseny** intentant mantenir la seua simplicitat.

XP

Pràctiques

El joc de la Planificació

Proves

Propietat Col·lectiva

Lliuraments Xicotets

Metàfora

Setmanes de 40 hores

Refactorització

Disseny Senzill

El Client sempre amb el Desenvolupador

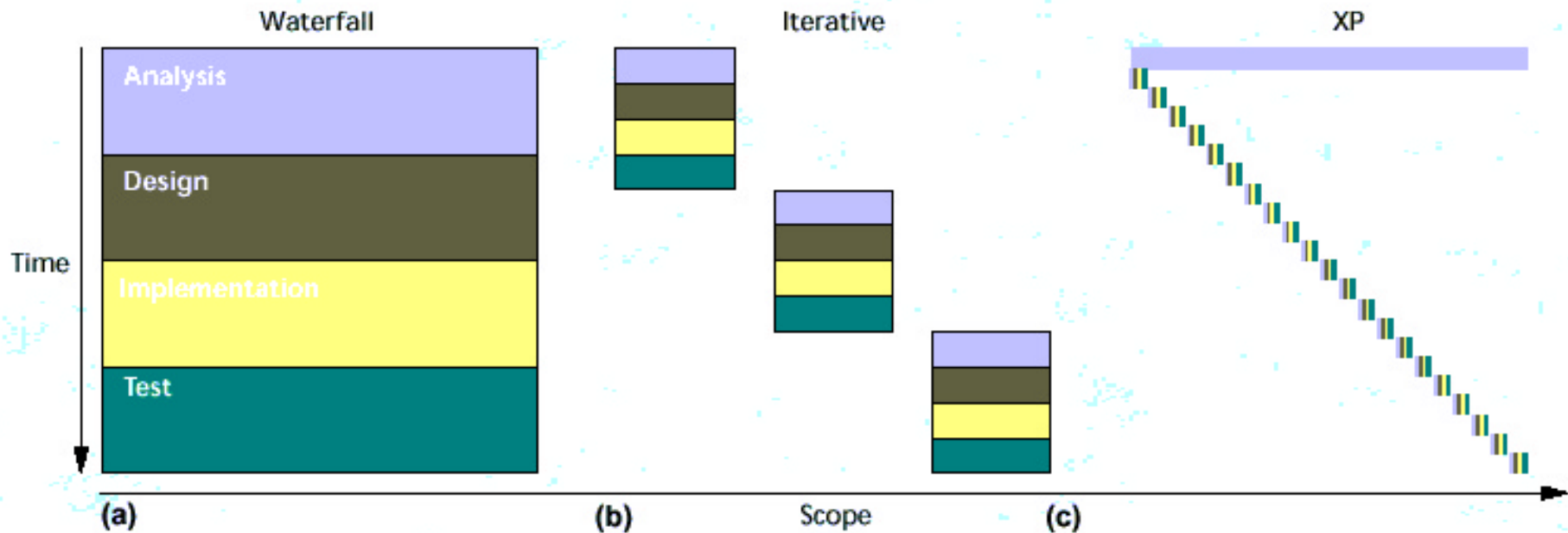
Programació en Parelles

Integració Contínua

Estàndards de Codificació

XP

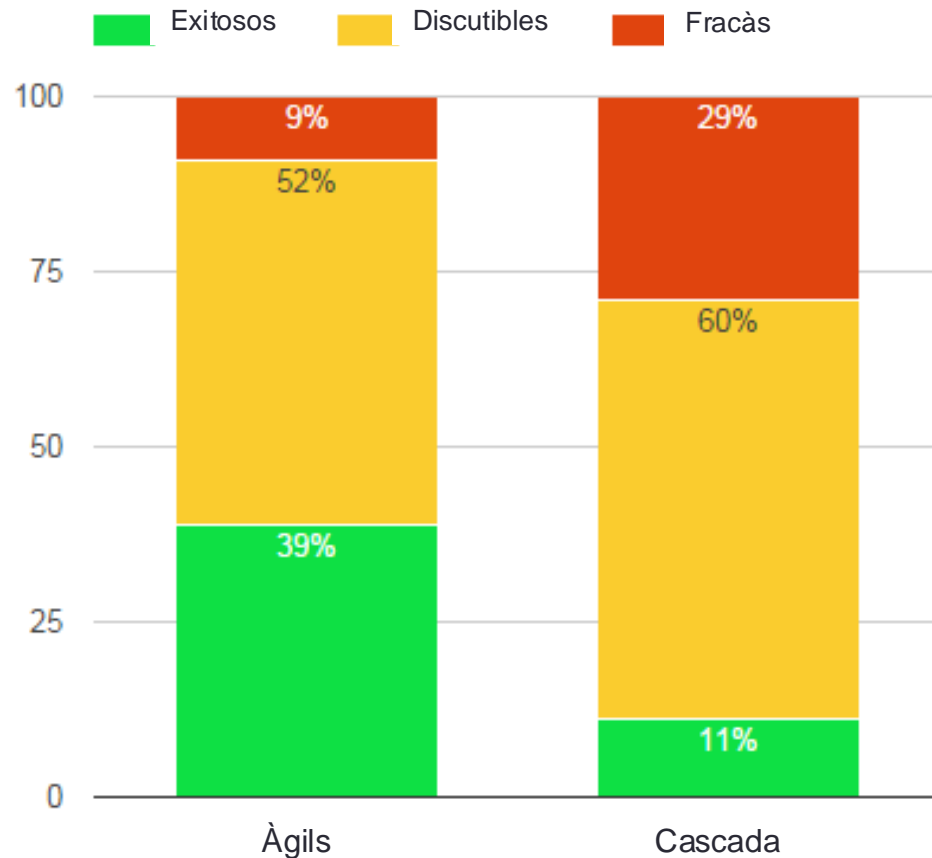
Comparativa



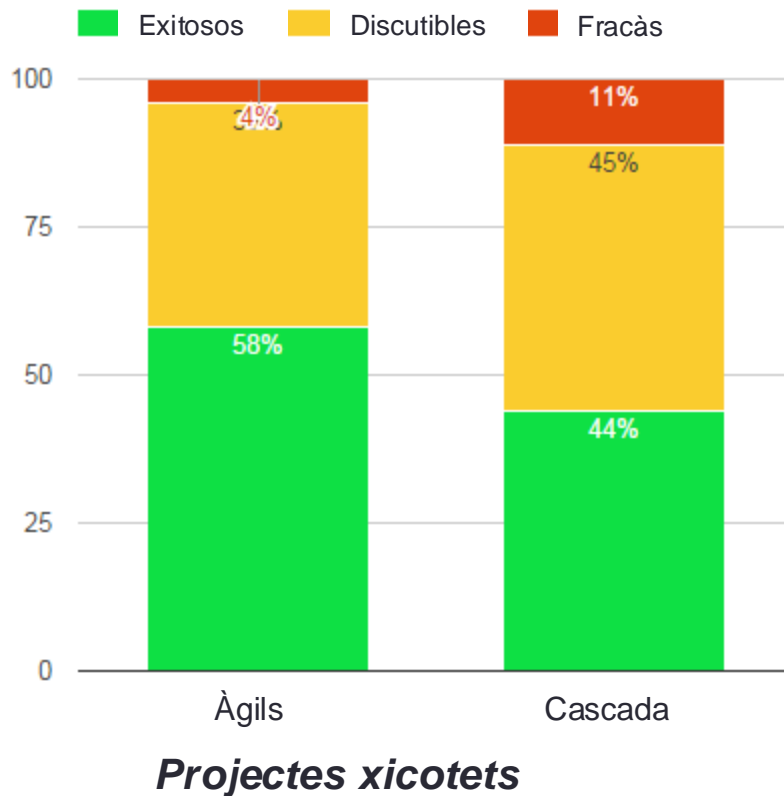
Àgils vs. Cascada

Comparativa

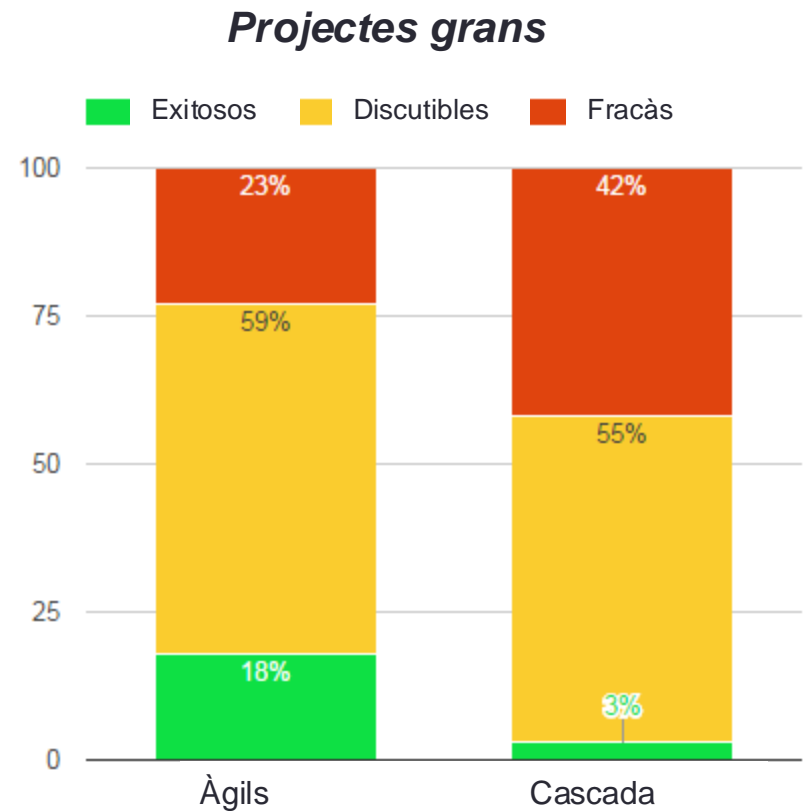
Comparativa de l'**èxit** dels projectes en funció de la **metodologia** seguida per al seu desenvolupament 2011-2015



Àgils vs. Cascada



Comparativa



SCRUM

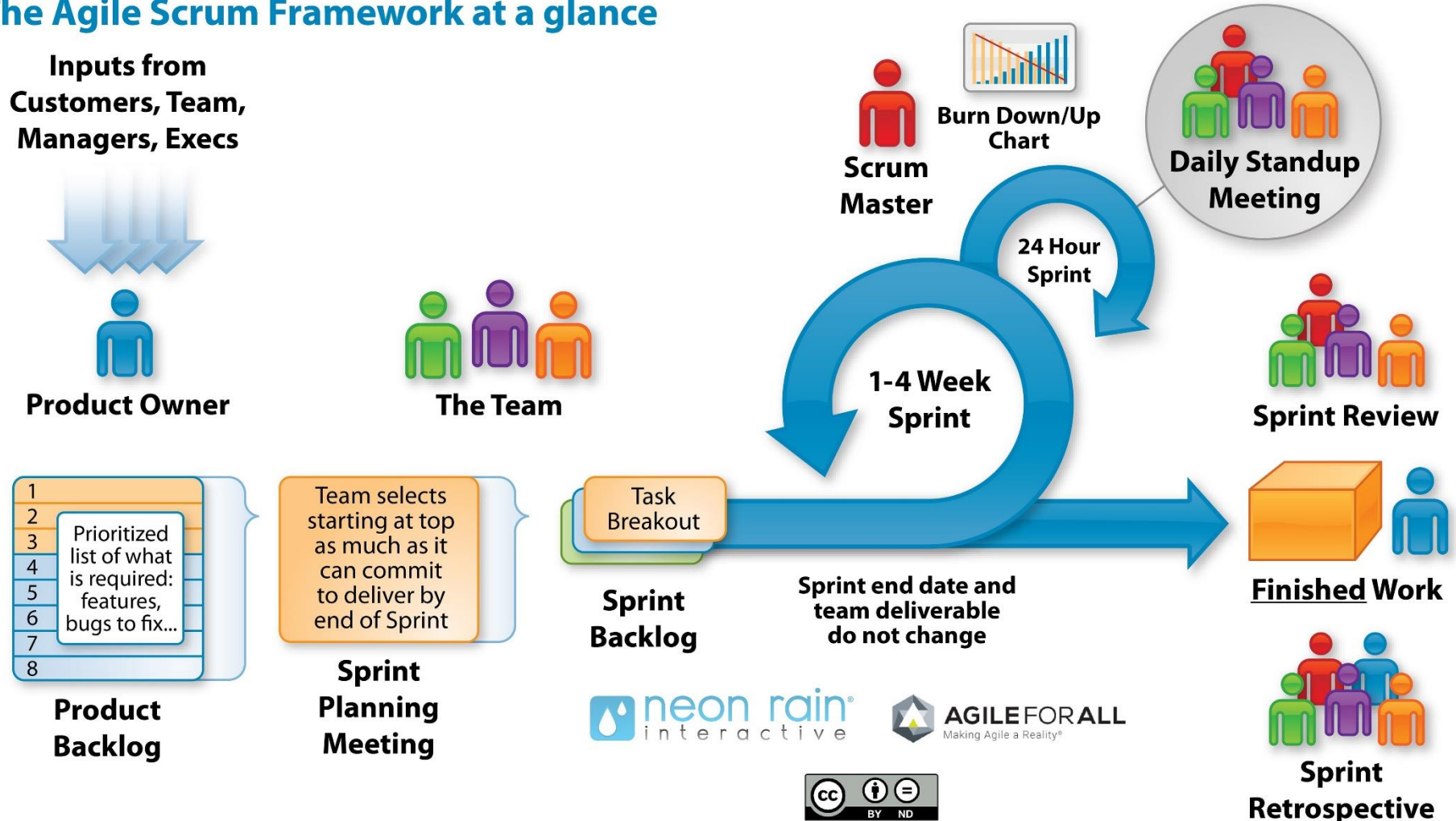


Jeff Sutherland, Ken Schwaber

<https://scrumguides.org/>

- Marc per a la gestió àgil de projectes
- Suport a la col·laboració
- Equips auto-dirigits i auto-organitzats
- El client rep versions funcionals del producte final contínuament al finalitzar cada **sprint** (1 a 4 setmanes)
- Defineix el seu propi cicle de desenvolupament i nomenclatura

The Agile Scrum Framework at a glance



Kanban

https://resources.kanban.university/wp-content/uploads/2021/08/The-Official-Kanban-Guide_Spanish_A4.pdf



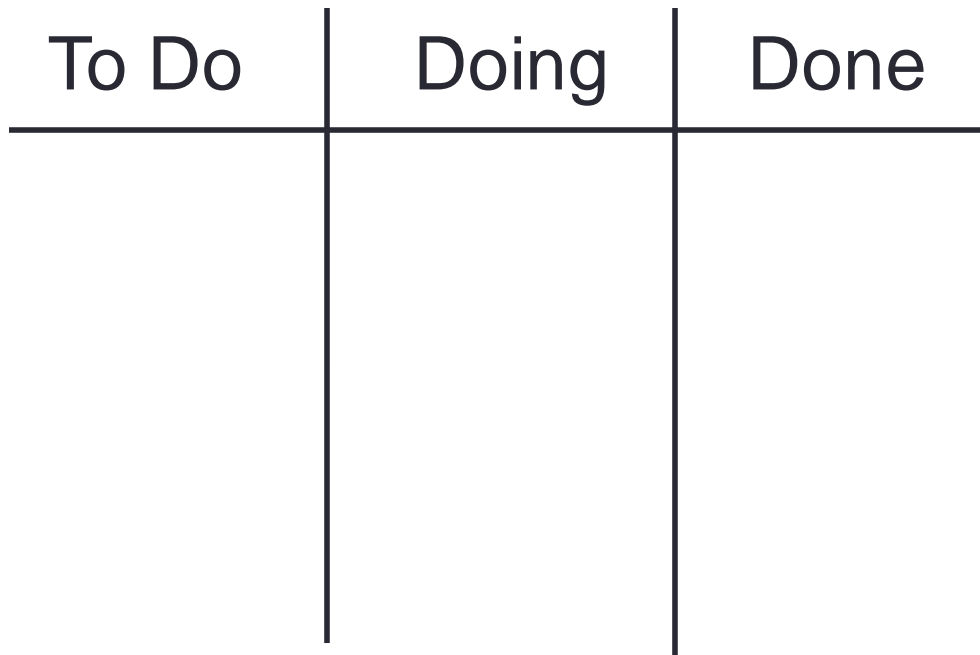
- Gestionant el treball en equip



Kanban elemental

A

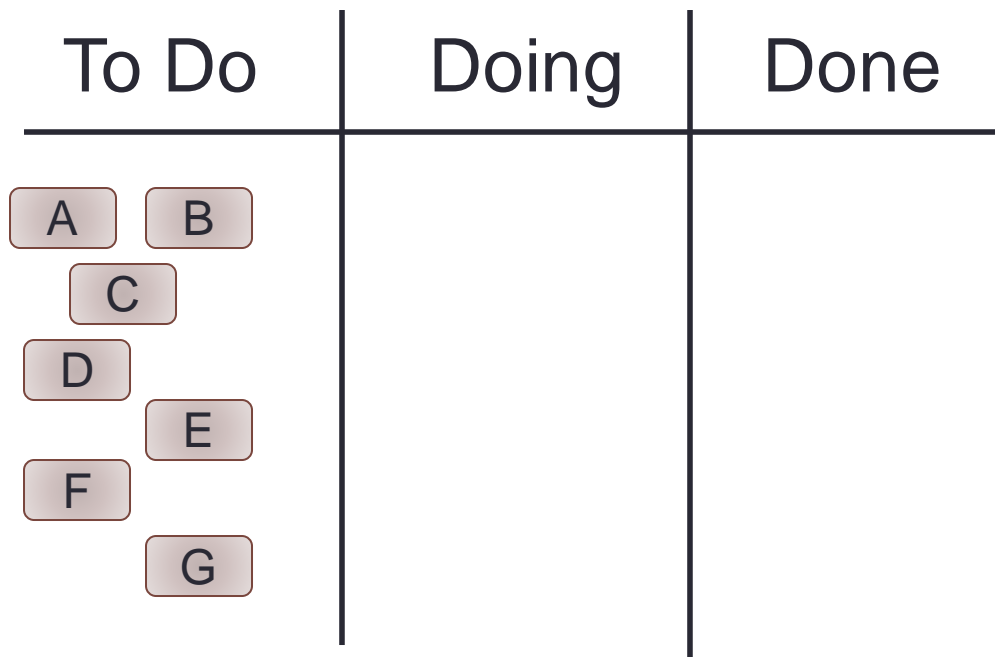
Tasca // Work Unit (WU)



Kanban elemental

A

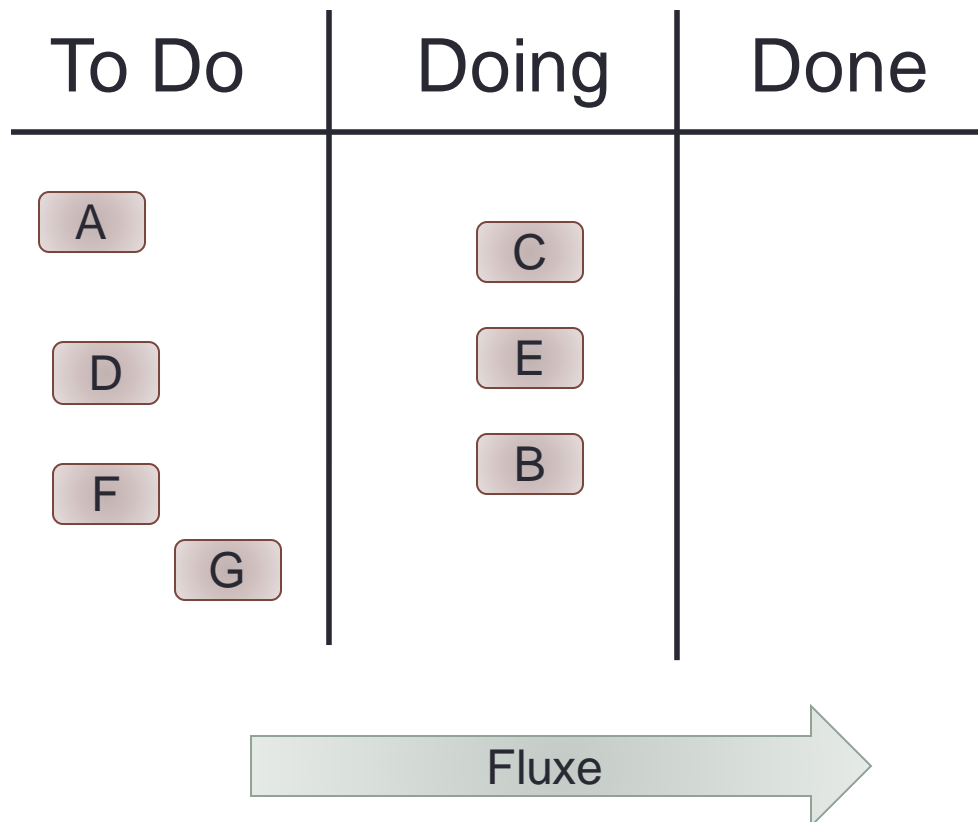
Tasca // Work Unit (WU)



Kanban elemental

A

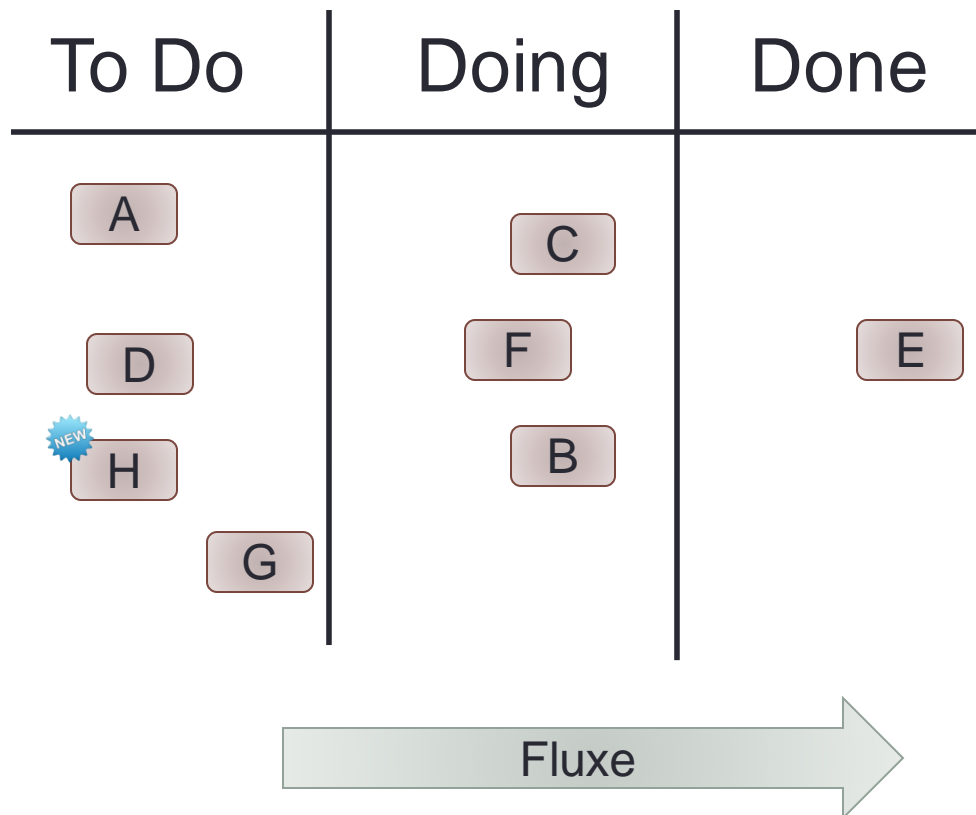
Tasca // Work Unit (WU)



Kanban elemental

A

Tasca // Work Unit (WU)

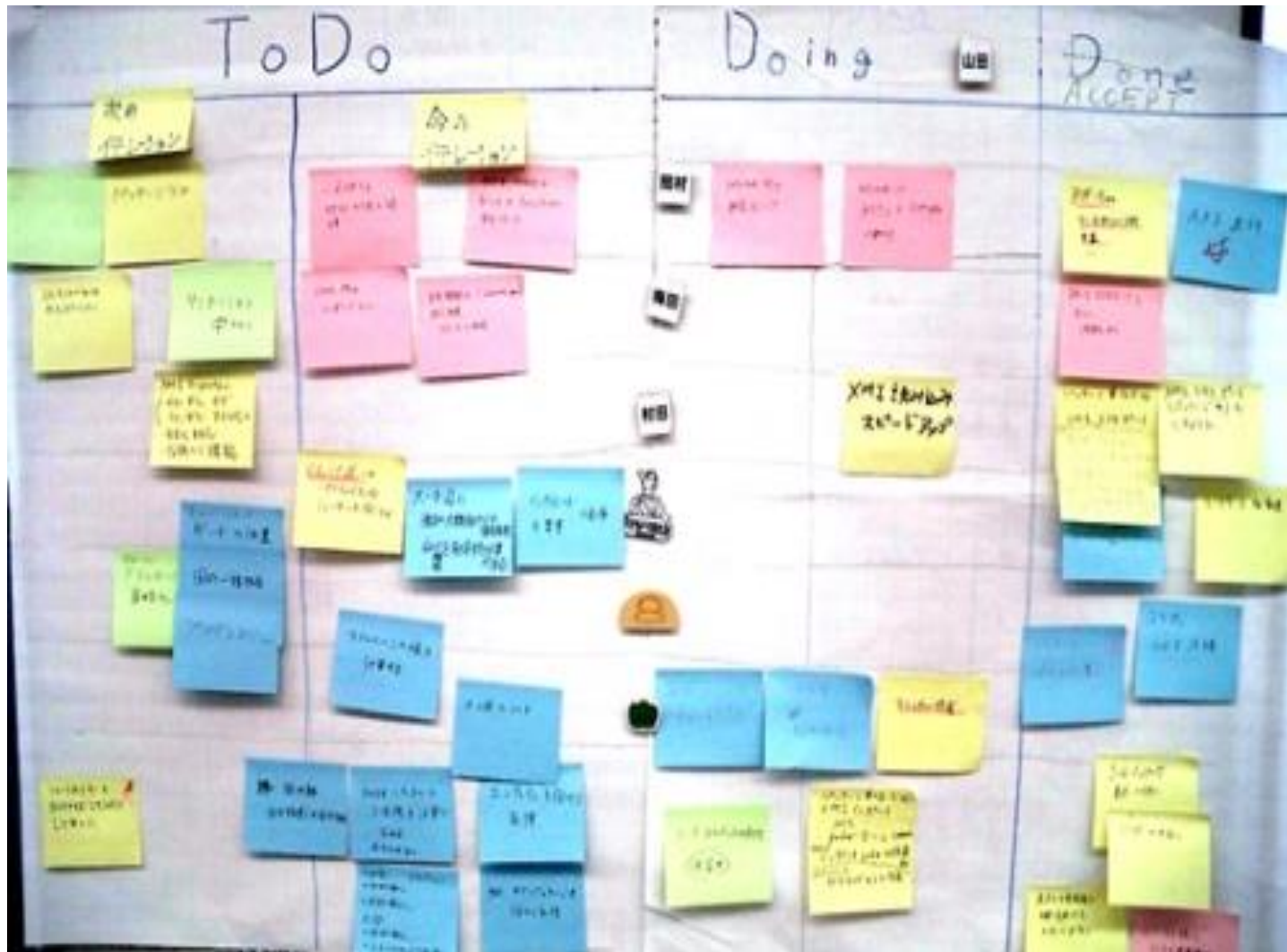


El tauler Kanban SEMPRE deberIA estar present per a visualitzar l'estat del treball

Kanban - Examples



Kanban - Exemples



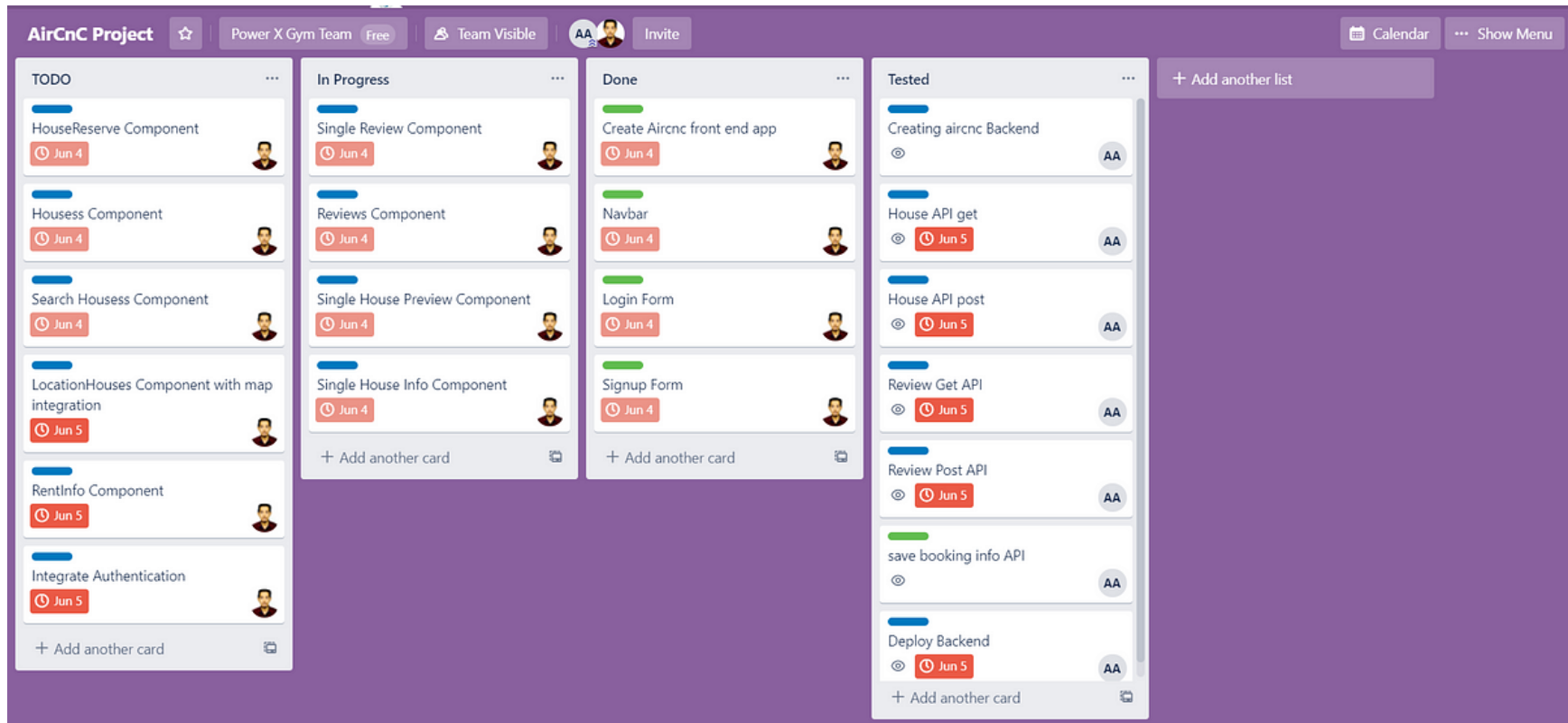
Kanban - Eines

- Eines de programari
- Gratuïtes (i en moltes ocasions sense instal·lació – accés al servidor)

<https://www.toptut.com/es/top-10-free-online-kanban-tools/>

- Trello (<https://trello.com/es>)
- Integrades en altres eines
 - AZURE DEVOPS (*Boards*) - *Seminari ISW **

Kanban - Eines



Kanban - Regles

1. Visualitzar el treball
2. Limitar el treball en curs (WIP)
3. Gestionar el flux de treball
4. Implementar polítiques de processos explícites
5. Implementar cicles de retroalimentació
6. Millorar col·laborant i evolucionar experimentalment

VISUALIZAR



Muestra el trabajo y su flujo.
Visualiza los riesgos.
Construye un modelo visual
que refleje cómo se trabaja.

LIMITAR EL TRABAJO EN CURSO



¡Deja de empezar, empieza a
terminar!
De derecha a izquierda. Limita el
trabajo en el sistema a la capacidad
disponible, basándote en los datos.

GESTIONAR EL FLUJO



El flujo es el movimiento del
trabajo.
Gestiona el flujo para ser
predecible y confiable.
Utiliza los datos.

HACER LAS POLÍTICAS EXPLÍCITAS



Ten políticas acordadas y visibles
para todos los involucrados:
- Criterios de *Pull* (tracción/arrastre)
- Límites de *WiP* (trabajo en curso)
- Clases de servicio
- Otras que correspondan

ESTABLECER CICLOS DE RETROALIMENTACIÓN



Establece ciclos de
retroalimentación con la
cadencia adecuada.
Fomenta la colaboración, el
aprendizaje y las mejoras.
Basado en datos.

MEJORAR COLABORATIVAMENTE, EVOLUCIONAR EXPERIMENTALMENTE



Usa el método científico.
Cambia basándote en hipótesis.
Ejecuta experimentos para
aprender (*safe-to-fail experiments*).

Kanban

