Il Modello Microsoft[1]

- La Microsoft, come altre organizzazioni che sviluppano software commerciale, ha dovuto affrontare, fin dalla metà degli anni 80, problemi di:
  - incremento della qualità dei prodotti software
  - riduzione di tempi e costi di sviluppo
- Per cercare di risolvere tali problemi si è adottato un processo che è al tempo stesso *iterativo*, *incrementale* e *concorrente* e che permette di esaltare le doti di creatività delle persone coinvolte nello sviluppo di prodotti software

[1] M.A. Cusumano and R.W. Selby, How Microsoft Builds Software, *Communications of the ACM*, vol. 40, n. 6, June 1997.

Approccio *synch-and-stabilize*

- L'approccio usato attualmente da Microsoft è noto come "***synchronize-and-stabilize***"
- Tale approccio è basato su:
  - **sincronizzazione** quotidiana delle attività svolte da persone che lavorano sia individualmente che all'interno di piccoli team (da 3 a 8 persone), mediante assemblaggio dei componenti software sviluppati (anche parzialmente) in un prodotto (*daily build*) che viene testato e corretto
  - **stabilizzazione** periodica del prodotto in incrementi (*milestone*) successivi durante l'avanzamento del progetto, piuttosto che un'unica volta alla fine

Ciclo di sviluppo a 3 fasi

- *Planning* phase
  - Define product vision, specification and schedule
- *Development* phase
  - Feature development in 3/4 sequential subprojects, each resulting in a milestone release
- *Stabilization* phase
  - Comprehensive internal and external testing, final product, stabilization and ship

Planning phase

Planning Phase Define product vision, specification, and schedule

- **Vision Statement** Product and program management use extensive customer input to identify and priority-order product features.
- **Specification Document** Based on vision statement, program management and development group define feature functionality, architectural issues, and component interdependencies.
- **Schedule and Feature Team Formation** Based on specification document, program management coordinates schedule and arranges feature teams that each contain approximately 1 program manager, 3–8 developers, and 3–8 testers (who work in parallel 1:1 with developers).

Development phase

Development Phase Feature development in 3 or 4 sequential subprojects that each results in a milestone release

Program managers coordinate evolution of specification. Developers design, code, and debug. Testers pair with developers for continuous testing.

- **Subproject I** First 1/3 of features (Most critical features and shared components)
- **Subproject II** Second 1/3 of features
- **Subproject III** Final 1/3 of features (Least critical features)

Stabilization phase

Stabilization Phase Comprehensive internal and external testing, final product stabilization, and ship

Program managers coordinate OEMs and ISVs and monitor customer feedback. Developers perform final debugging and code stabilization. Testers recreate and isolate errors.

- **Internal Testing** Thorough testing of complete product within the company

- **External Testing** Thotough testing of complete product outside the company by "beta" sites, such as OEMs, ISVs, and end users

- **Release preparation** Prepare final release of "golden master" disks and documentation for manufacturing

Strategie e Principi

1. Strategia per definire prodotto e processo: "*considerare la creatività come elemento essenziale*"

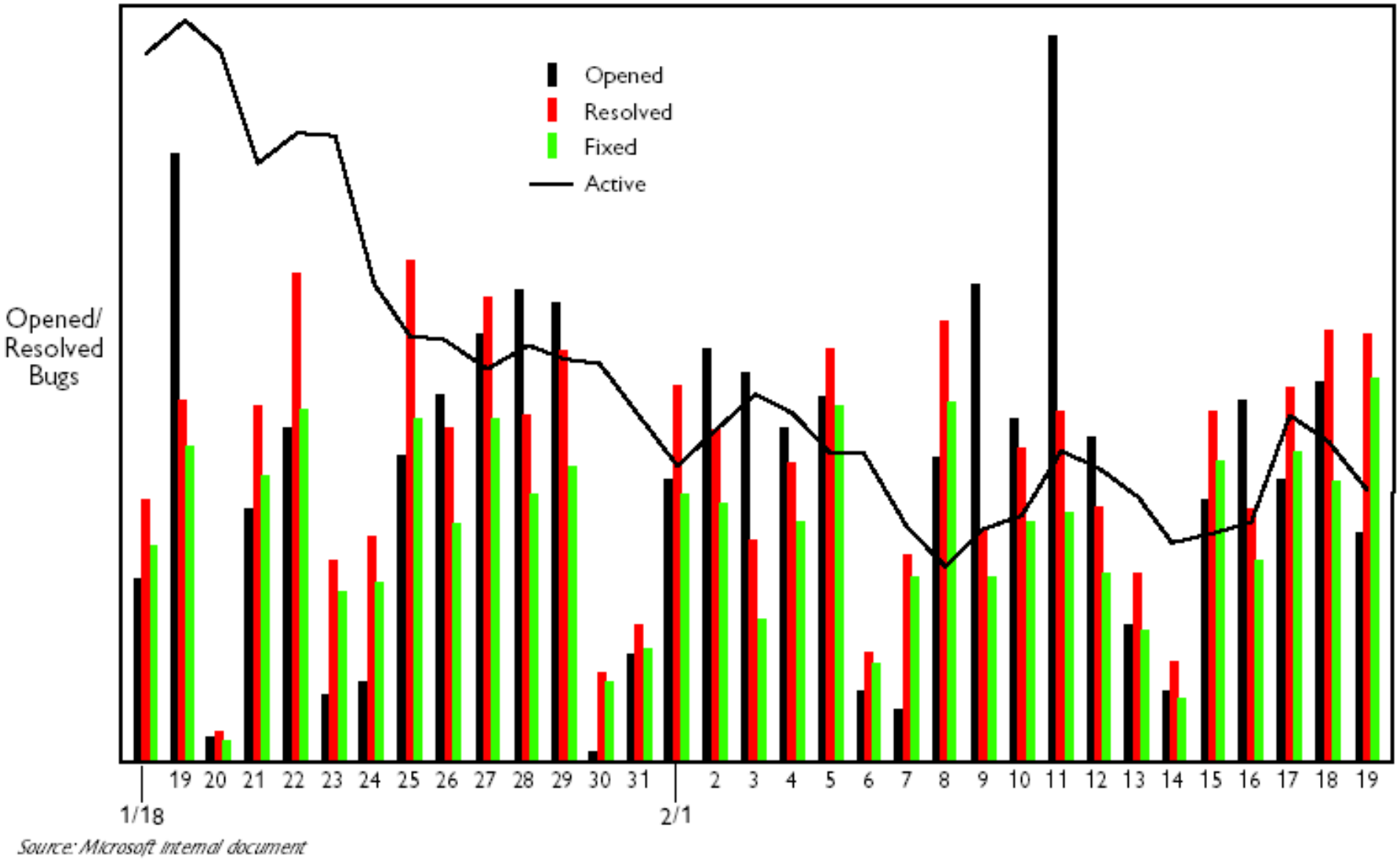
   Principi di realizzazione:

   a. Dividere il progetto in milestone (da 3 a 4)
   b. Definire una "product vision" e produrre una specifica funzionale che evolverà durante il progetto
   c. Selezionare le funzionalità e le relative priorità in base alle necessità utente
   d. Definire un'architettura modulare per replicare nel progetto la struttura del prodotto
   e. Assegnare task elementari e limitare le risorse

Strategie e Principi (2)

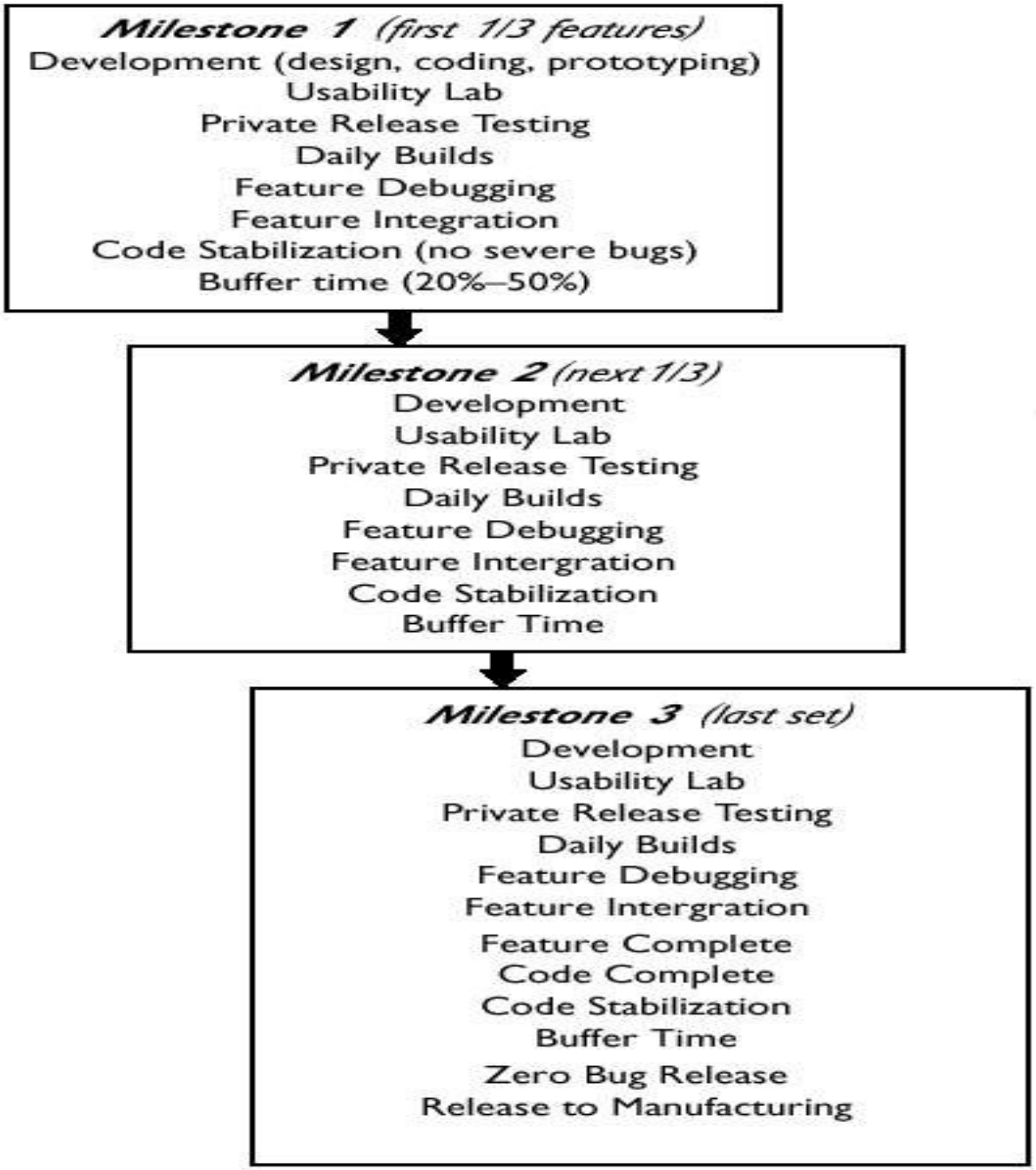
2. Strategia per lo sviluppo e la consegna dei prodotti: "*lavorare in parallelo con frequenti sincronizzazioni*"
   Principi di realizzazione:
   a. Definire team paralleli ed utilizzare *daily build* per la sincronizzazione
   b. Avere sempre un prodotto da consegnare, con versioni per ogni piattaforma e mercato
   c. Usare lo stesso linguaggio di programmazione all'interno dello stesso sito di sviluppo
   d. Testare continuamente il prodotto durante il suo sviluppo
   e. Usare metriche per il supporto alle decisioni

Esempio di metriche collezionate

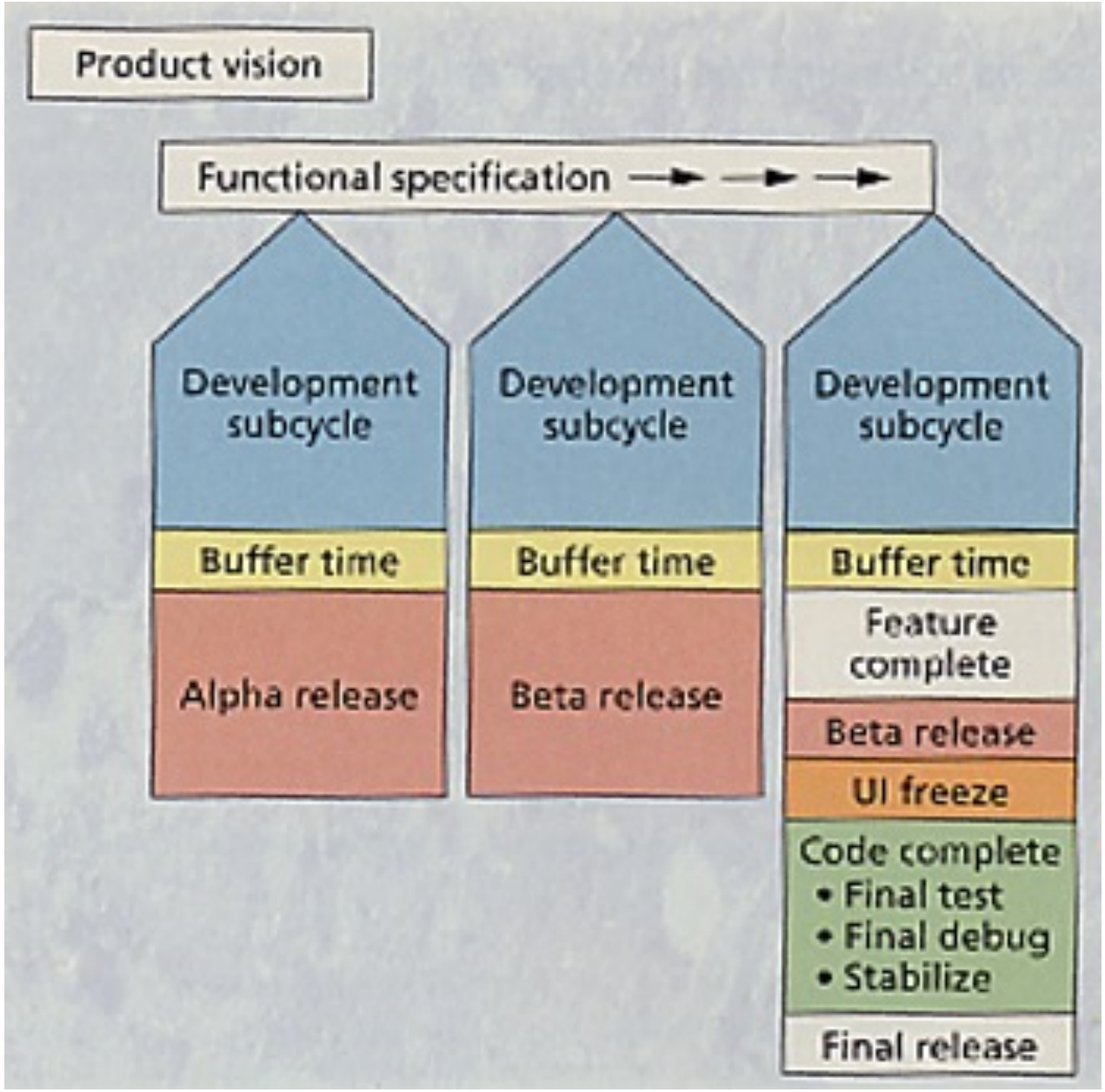


Source: Microsoft internal document

Milestones


Milestone 1 (first 1/3 features)
Development (design, coding, prototyping)
Usability Lab
Private Release Testing
Daily Builds
Feature Debugging
Feature Integration
Code Stabilization (no severe bugs)
Buffer time (20%–50%)
Milestone 2 (next 1/3)
Development
Usability Lab
Private Release Testing
Daily Builds
Feature Debugging
Feature Intergration
Code Stabilization
Buffer Time
Milestone 3 (last set)
Development
Usability Lab
Private Release Testing
Daily Builds
Feature Debugging
Feature Intergration
Feature Complete
Code Complete
Code Stabilization
Buffer Time
Zero Bug Release
Release to Manufacturing

Modello del ciclo di sviluppo *synch-and-stabilize*



Confronto tra modelli *synch-and-stabilize* e *waterfall*

Synch-and-Stablize	Sequential Development
Product development and testing done in parallel	Separate phases done in sequence
Vision statement and evolving specification	Complete "frozen" specification and detailed design before building the product
Features prioritized and built in 3 or 4 milestone subprojects	Trying to build all pieces of a product simultaneously
Frequent synchronizations (daily builds) and intermediate stabilizations (milestones)	One late and large integration and system test phase at the project's end
"Fixed" release and ship dates and multiple release cycles	Aiming for feature and product "perfection" in each project cycle
Customer feedback continuous in the development process	Feedback primarily after development as inputs for future projects
Product and process design so large teams work like small teams	Working primarily as a large group of individuals in a separate functional department

Il Modello Netscape[2]

- Anche alla Netscape si è adottato un modello di tipo *synchronize-and-stabilize*, con opportuni adattamenti allo sviluppo di applicazioni Internet (browser e prodotti server):
  - dimensione dello staff
    - in media 1 tester ogni 3 sviluppatori (ma stessa produttività di Microsoft nello sviluppo di prodotti comparabili, ad es. *IE* vs. *Communicator*)
  - processo
    - scarso effort di pianificazione (tranne che su prodotti server)
    - documentazione incompleta
    - scarso controllo sullo stato di avanzamento del progetto (lasciato all'esperienza e all'influenza dei project manager)
    - scarso controllo su attività di ispezione del codice (code review)
    - pochi dati storici per il supporto alle decisioni

[2] M.A. Cusumano and D.B. Yoffie, Software Development on Internet Time, *IEEE Computer*, October 1999.

Staffing

Table 2. Allocation of staff in Netscape's client and server development divisions, mid-1998.

	Client products	Server products	Total
Software engineering	110	200	310
Testing (QA)	50	80	130
Product management	50	42	92
Subtotal	210	322	533
Other*	30	98	128
Total	240	420	660

*Persons in activities such as documentation, user interface design, OEM porting, internationalization and localization, and special product support.

Netscape Development Process (1)

Step 1: Product requirements and project proposal

Advance planning meeting (APM) held to brainstorm ideas (marketing, development, executives)

Product vision generated, initially by senior engineers, now mainly by product managers

Some design and coding by engineers to explore alternative technologies or feature ideas

Product requirements document compiled by product managers, with help from developers

Informal review of this preliminary specification by engineers

Functional specification begun by engineers, sometimes with help from product managers

Schedule and budget plan compiled by marketing and engineering, and informally discussed with executives

Step 2: First executive review

Executives review product requirements document and schedule and budget proposal

Plan adjusted as necessary

Netscape Development Process (2)

Step 3: Start of development phase
Design and coding of features, architecture work as necessary
Daily integration of components as they are created and checked in (builds)
Bug lists generated and fixes initiated

Step 4: Interim executive review (if necessary)
Functional specification should be complete at this point
Midcourse corrections in specification or project resources, as necessary
Coordination issues with other products or projects discussed, as necessary
Development continues

Step 5: First internal (alpha) release (takes approximately six weeks)
Development stops temporarily
Intensive debugging and testing of existing code
Alpha release for internal feedback (or possibly a developer's release)
Development continues
User feedback incorporated
Feature-complete target (rarely met, though servers especially try to be as complete as possible)
One week to stabilize beta release

Netscape Development Process (3)

Step 6: Public beta 1 or field test 1 (takes approximately six weeks)
Repeat development and testing steps in Step 5
Server groups moving to "field tests" with limited customers rather than public betas

Step 7: Public beta 2 and 3 (each beta takes approximately six weeks)
Repeat development steps as in Step 5
UI freeze milestone
Feature-complete status "mandatory," although some minor changes still allowed

Step 8: Code complete
No more code added except to fix bugs; features are functionally complete

Step 9: Final testing and release
Final debugging and stabilization of release candidate
Certification meeting(s) with senior executives for ship decision
Release to manufacturing (RTM) and commercial release

Agile Methods

- In the early 2000's, a reaction was witnessed against the importance of carefully planned software processes, stating that such processes are too restrictive on the developers
- The term *agile method* was introduced to extend the original concept of iterative and incremental development to concepts such as intensive communication within the project, fast feedback, few external rules for the way of working, etc.
- The common values and principles of agile methods are summarized in the *Agile Manifesto*

Agile Manifesto (2001)

- The Agile Manifesto defines *agile values*

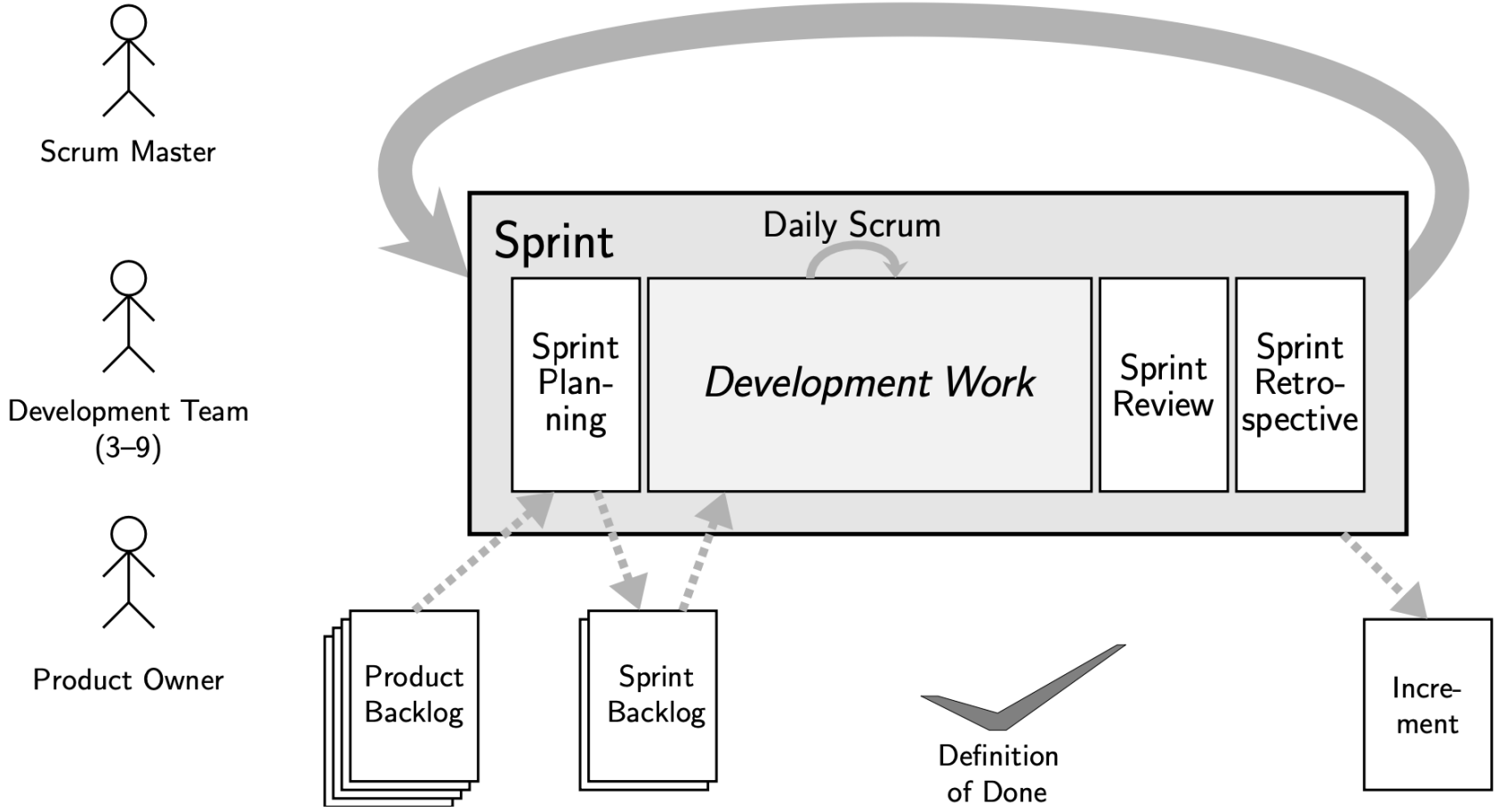
  "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

  – **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."

- In addition to values, the Agile Manifesto contains the *twelve agile principles* that provide additional guidance on the use of agile methods
- Together, these values and principles define the basic concepts that are today known as *agile development*

Scrum


Scrum Master
Development Team
(3–9)
Product Owner
Sprint
Daily Scrum
Sprint Plan-ning
Development Work
Sprint Review
Sprint Retro-spective
Product Backlog
Sprint Backlog
Definition of Done
Incre-ment

Scrum Roles

- The *Scrum master*:
  - ensures that the methodology is understood and properly implemented by the development team and the product owner
  - supports the team by helping everyone else to interact with the team according to the Scrum rules
- The *product owner* manages and helps prioritizing the requirements to be implemented as documented in the product backlog
- The *development team* is responsible for developing the product, including all relevant tasks (e.g., designing, coding and testing)

Sprints

- A ***sprint*** is carried out to deliver a new increment of working software, and typically takes 2 to 4 weeks
- It is started with the *sprint planning* meeting where the Scrum team transfers the items to be developed from the product backlog and transfers them into the sprint backlog
- During the sprint, the development team works on the increment, with short *daily Scrum* meetings (aka stand-up meeting) of the development team to synchronize work and address any problems
- At the end of a sprint, the increment is presented to the product owner and other stakeholders in a *sprint review*
- Finally, the *sprint retrospective* meeting is performed to identify and plan any improvements for the next sprint

Definition of *Done*

- Scrum requires a “definition of done”, where the development team defines for itself what it means for a work item to be done (i.e., before it is allowed to be integrated into the main branch)
- Typical minimum requirements included in this “definition of done” include an adequate number of test cases, as well as checking the integration of the new code to ensure that it does not break the main development branch
- The definition of done also requires that the code has been documented adequately, where the team defines for itself what ”adequate” means

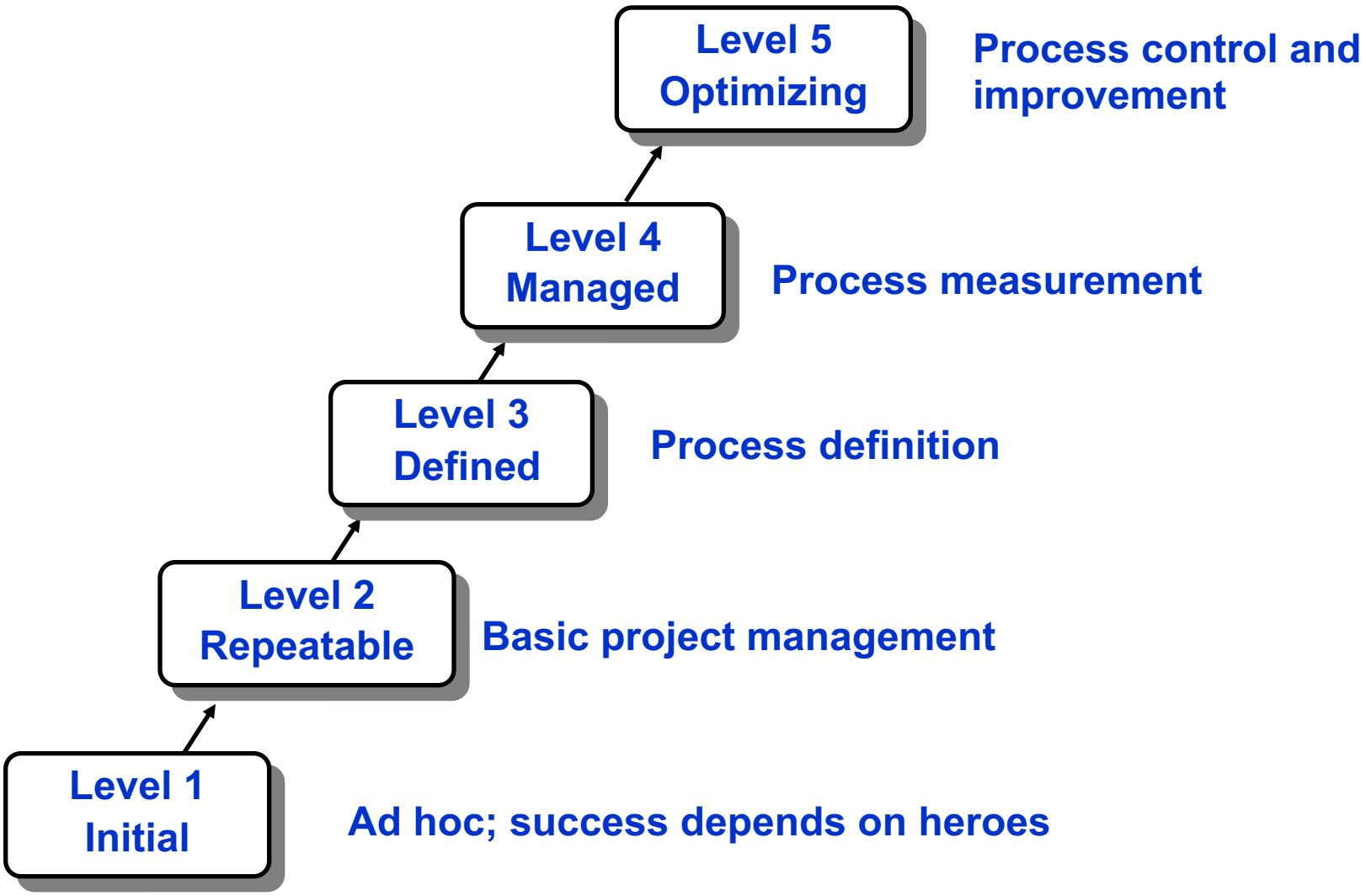
User Stories

- A common practice, used in agile development, often in combination with Scrum (but not defined in the Scrum Guide)
- A user story:
  - is a format for describing user requirements as a "story". Is should be short, typically just one sentence, and described from the user point of view
  - uses a common template, such as
    - *As a <role>, I want <goal> so that <benefit>*
      - Example: "*As a process engineer, I want to see the dependencies between different process steps so that I can easily verify and validate them*"
- Large user stories which are later broken down into smaller ones are often called *epics*

Capability Maturity Model (CMM)

- Il SEI (Software Engineering Institute) ha predisposto, a partire dal 1993, un modello per determinare il livello di maturità del processo software di un'organizzazione (ovvero una misura dell'efficacia globale dell'applicazione di tecniche di ingegneria del software)
- Il modello è basato su un questionario ed uno schema valutativo a cinque livelli
- Ogni livello comprende tutte le caratteristiche definite per il livello precedente

I 5 livelli del CMM

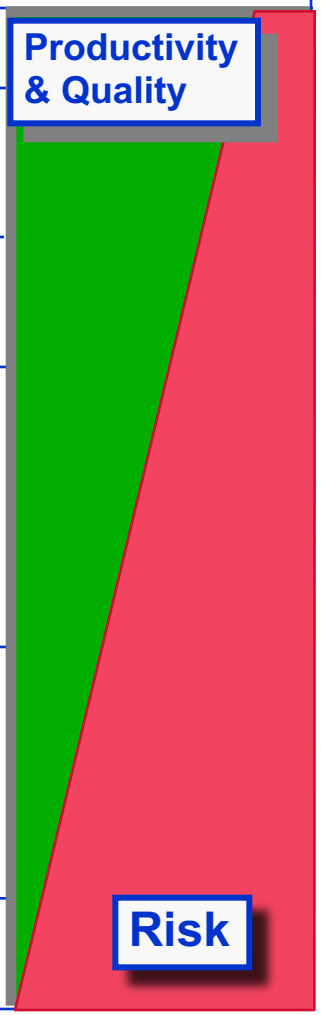


Key Process Areas

- Il CMM associa a ogni livello di maturità alcune KPA (Key Process Area), tra le 18 definite, che descrivono le funzioni che devono essere presenti per garantire l'appartenenza ad un certo livello.
- Ogni KPA è descritta rispetto a:
  - obiettivi
  - impegni e responsabilità da assumere
  - capacità e risorse necessarie per la realizzazione
  - attività da realizzare
  - metodi di "monitoring" della realizzazione
  - metodi di verifica della realizzazione

CMM KPAs

Level	Characteristic	Key Process Areas
Optimizing (5)	Continuous process capability improvement	Process change management Technology change management Defect prevention
Managed (4)	Product quality planning; tracking of measured software process	Software quality management Quantitative process management
Defined (3)	Software process defined and institutionalized to provide product quality control	Peer reviews Intergroup coordination Software product engineering Integrated software management Training program Organization process definition Organization process focus
Repeatable (2)	Management oversight and tracking project; stable planning and product baselines	Software configuration management Software quality assurance Software subcontract management Software project tracking & oversight Software project planning Requirements management
Initial (1)	Ad hoc (success depends on heroes)	"People"

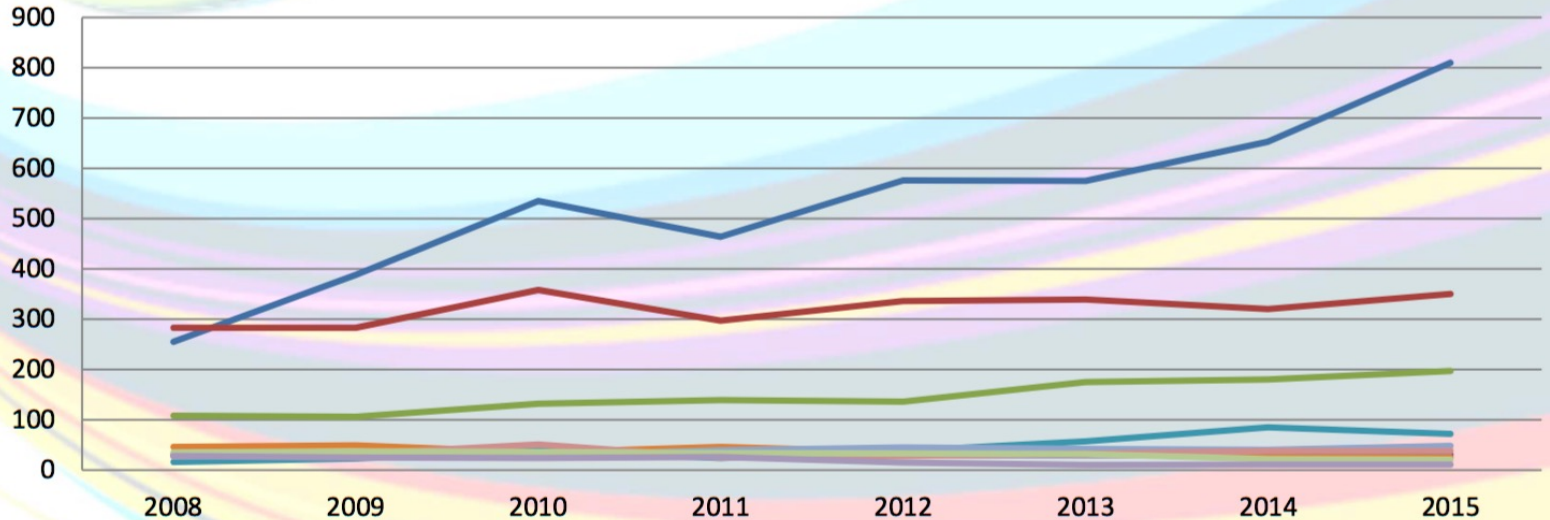


Statistiche a Febbraio 2000

La lista delle organizzazioni a livello 4 e 5 (*maturità elevata*) include:

- 71 organizzazioni negli USA
  - 44 organizzazioni a Livello 4 (tra cui Oracle, NCR, Siemens Info Systems, IBM Global Services)
  - 27 organizzazioni a Livello 5 (tra cui Motorola, Lockeed-Martin, Boeing, Honeywell)
- 25 organizzazioni al di fuori degli USA
  - 1 organizzazione a Livello 4 in Australia
  - 14 organizzazioni a Livello 4 in India
  - 10 organizzazioni a Livello 5 in India

Number of appraisals by country (06/15)



	2008	2009	2010	2011	2012	2013	2014	2015
China	255	388	535	464	576	575	653	810
United States	283	283	358	297	336	339	320	350
India	108	106	132	139	136	175	180	197
Spain	30	30	30	30	30	30	30	30
Mexico	16	23	39	24	38	57	85	72
Japan	46	49	35	46	36	32	26	25
Korea, Republic Of	27	36	29	39	45	42	40	48
Brazil	28	32	51	25	28	33	38	38
France	35	37	36	34	33	32	22	21
United Kingdom	28	25	24	26	15	10	11	11

Trends (as of June 2015)