Il metodo Essence

Paolo Ciancarini

Capitolo 1

Essence: elementi di base

Essence è un metodo di descrizione e analisi di processi e attività, in particolare di processi e attività di sviluppo del software.

Il metodo è uno standard internazionale dell'Object Management Group - OMG (ricordiamo che OMG è responsabile della standardizzazione di UML) [4]. Essence è basato su SEMAT, una teoria scientifica dello sviluppo software presentata in [2] e riassunta in [1].

Essence propone una notazione grafica per descrivere le attività di costruzione del software. In questo modo Essence - come in parte anche UML, per la parte di diagrammi che supportano la descrizione di attività e processi - permette la modellazione e l'analisi di processi di sviluppo del software.

L'ipotesi alla base di Essence è che qualsiasi metodo di sviluppo è un insieme di *pratiche*, cioè di prassi operative. Ad esempio, la Fig.1.1 mostra un metodo che include tre pratiche: *Backlog Driven*, *User Story*, e *TDD* (Test Driven Development), più altre non meglio specificate.

Essence si concentra sugli aspetti essenziali dello sviluppo, cioè sulle buone pratiche. Ad esempio la Fig.1.2 mostra le (classi di) pratiche principali dell'approccio agile.

La Fig.1.2 pone giustamente al centro, in verde - che è il colore dell'area *Customer*, la pratica del Product Ownership. Questa pratica è centrale perché determina la caratteristiche del prodotto che serviranno a soddisfare gli Stakeholder. Le pratiche in blu - il colore dell'area *Endeavor* (ovvero "iniziativa progettuale"), quella del processo di sviluppo - riguardano come opera il team. Le pratiche in giallo - l'area *Solution* - riguardano il sistema in costruzione, che diventerà il prodotto.

Essence è un metodo concepito come supporto per i team agili, in particolare durante l'esecuzione delle retrospettive, per esempio nel modello agile Scrum. In sostanza l'uso di Essence permette allo Scrum Master - anche se poco esperto - di gestire le retrospettive del team in modo coerente con le buone pratiche di sviluppo agile. Il metodo intende appunto essere utile anche ai team con poca esperienza delle pratiche agili.

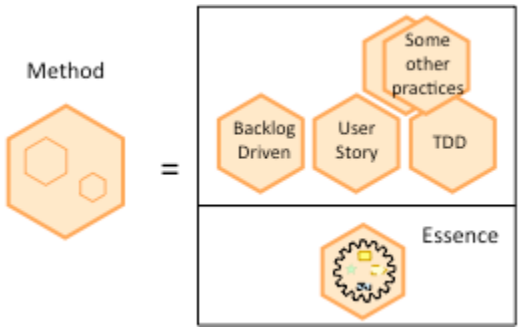

Figura 1.1: Un metodo di sviluppo è un insieme strutturato di *pratiche* (prassi di sviluppo) e può essere *essenzializzato*, cioè descritto con Essence. Nella figura è mostrato un metodo - *Method* - che include le seguenti pratiche: il backlog, le user story, il Test-Driven Development, più altre pratiche al momento indefinite. Il *kernel* di Essence è rappresentato sotto le pratiche

Gli elementi di base della notazione sono riportati in Tab.1.1: Alpha, Artefatti, Com-

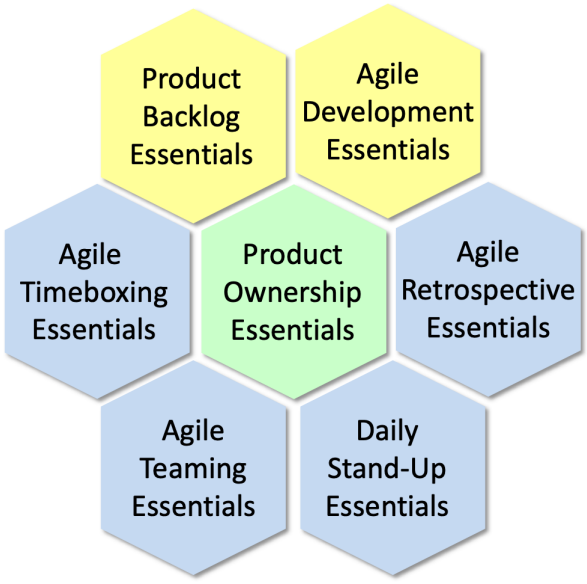

Figura 1.2: Le pratiche agili principali [1]: il backlog di prodotto (*Product Backlog*), lo sviluppo agile, la gestione del tempo con la tecnica del *timeboxing*, la gestione del prodotto (*Product Ownership*), le retrospettive, la gestione del team agile, gli incontri quotidiani (*Daily Stand-Up*). Il significato dei colori è spiegato nella fig.1.4

petenze, Attività, Spazi di attività e Pattern.

Tabella 1.1: Gli elementi di base di Essence

elemento	**simbolo**	**spiegazione**
Alpha		elemento essenziale di uno sviluppo, rilevante per valutarne lo stato
Artefatto		documento prodotto dallo sviluppo
Competenza		conoscenza o abilità necessaria per fare un certo tipo di attività
Attività		operazioni fatte dagli sviluppatori
Spazio di attività		nome che denota un insieme di più attività nello spazio Endeavour
Pattern		insieme composito di elementi di questa tabella

Gli Alpha sono gli elementi principali - o *primitivi* - di un processo, ad esempio il *Team di sviluppo* e il *Sistema* da sviluppare.

Gli *Artefatti* sono i documenti che vengono prodotti dalle *Attività*, le quali richiedono certe *Competenze*. Un esempio di Artefatto è il codice sorgente di un Sistema; un esempio di Attività è la scrittura del codice sorgente; un esempio di Competenza è la conoscenza di un linguaggio di programmazione, ad esempio Python, adatto a scrivere il codice del sistema.

Un *Pattern* è un elemento non primitivo ma composito, che descrive una particolare soluzione ad un problema di sviluppo. Ad esempio il Product Owner, lo Scrum Master e lo Scrum Team (i tre ruoli definiti nel processo Scrum) sono tutti elementi compositi - ovvero pattern - perché hanno bisogno di più Alpha per essere definiti compiutamente. La Fig.1.3 mostra la descrizione dei rispetivi pattern corrispondenti ai tre ruoli.

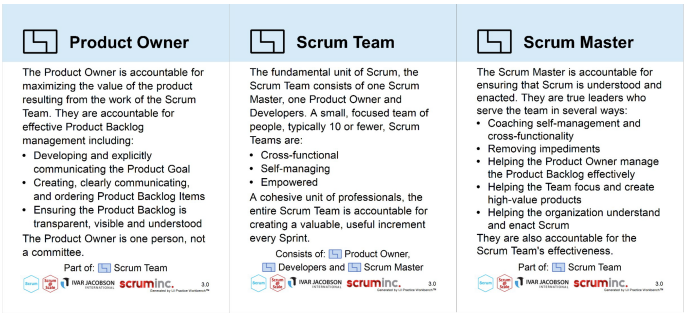

Figura 1.3: I ruoli Scrum sono pattern: si osservi l'icona in alto a sinistra su ciascuna carta

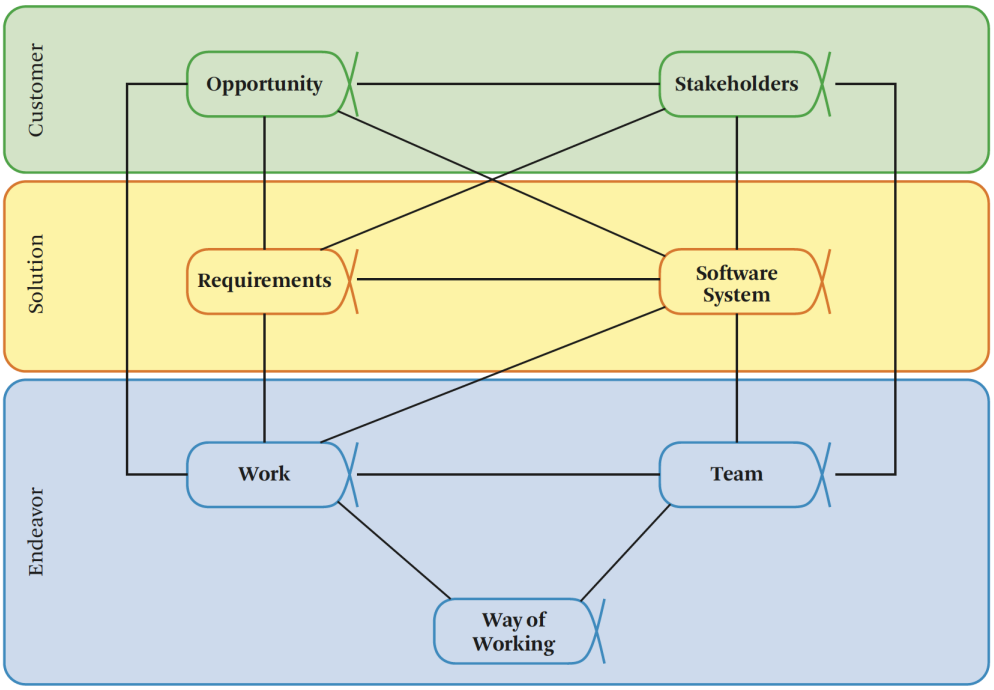

Figura 1.4: I sette Alpha principali di Essence [1]. Si notino i colori: verde per il *Customer*, giallo per la *Solution*, blu per l'*Endeavour*

Il metodo Essence si basa sull'impiego di *checklist* - in forma di mazzi di carte - per l'analisi del processo da seguire. Le carte sono liberamente disponibili e scaricabili (si vedano i link a corredo di questo documento).

Un team che scelga con cura le carte che descrivono lo stato del proprio lavoro ed ne esamini i contenuti può analizzare e cercare di migliorare lo stato e l'evoluzione del proprio processo di sviluppo.

La Fig.1.4 mostra i sette elementi principali di un processo (gli *Alpha*, nel gergo Essence). Sono inclusi i seguenti elementi: gli *Stakeholders*, l'*Opportunity*, i *Requirements*, il *Software System*, il *Team*, le attività (*Work*), il metodo di sviluppo (*Way of Working*).

Ciascun *Alpha* è un componente essenziale di ogni processo di sviluppo. Ogni Alpha evolve attraverso diversi stati. La Fig.1.5 mostra gli stati possibili dei sette Alpha principali.

Quindi un insieme di sette carte, uno per ciascun alpha, permette di definire uno stato del processo che sta eseguendo il team. L'analisi dello stato del processo permette al team di comprendere come sta progredendo nello sviluppo.

Ogni processo viene articolato da Essence su tre aree di interesse: l'area del Cliente (*Customer*, in verde), l'area del Sistema (*Solution*, in giallo); l'area del Progetto (*Endeavour*, in blu). Vediamo più in dettaglio cosa rappresentano gli Alpha e cosa significano.

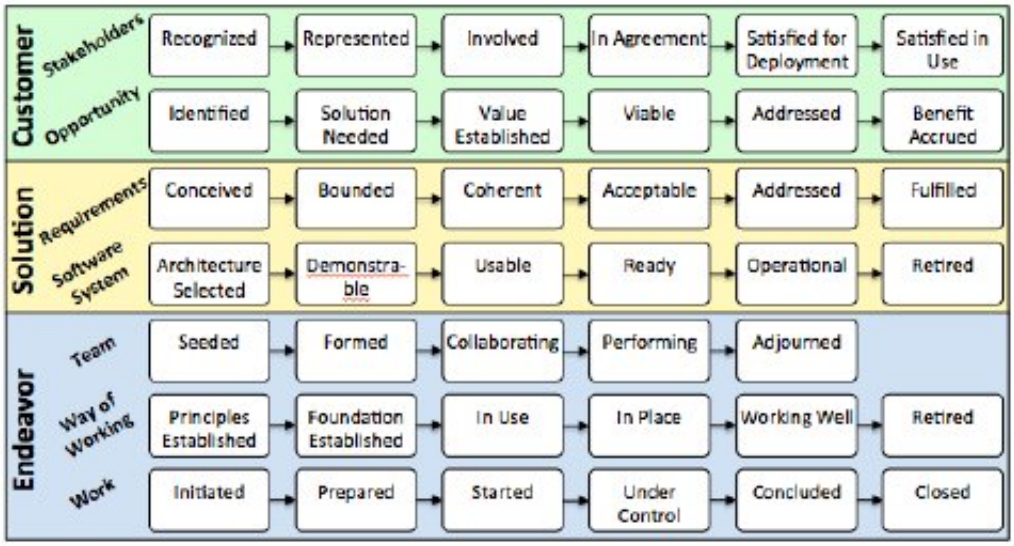

Figura 1.5: Gli stati possibili dei sette Alpha principali

Gli *Stakeholders* sono le persone interessate al progetto. Includono sia i ruoli interni al progetto - i "pig", ovvero PO, SM, sviluppatori - sia quelli esterni - i "chicken" ovvero gli utenti finali, i decisori, i manager, e chiunque sia interessato al prodotto finale.

I possibili stati della carta Stakeholders sono: *Recognized*, *Represented*, *Involved*, *In Agreement*, *Satisfied for Deployment*, *Satisfied in Use*.

Questi stati descrivono il livello di coinvolgimento e soddisfazione degli stakeholder lungo il ciclo di vita del progetto:

1. *Recognized* (Riconosciuti): Gli stakeholder principali sono stati identificati e sono noti al team. In questo stato, si ha una lista preliminare delle persone o gruppi che hanno interesse nel progetto o possono influenzarlo, anche se il loro coinvolgimento non è ancora definito.

2. *Represented* (Rappresentati): È stato assegnato un rappresentante per ciascun gruppo di stakeholder. Questo stato implica che i bisogni e le aspettative degli stakeholder siano chiariti e abbiano un referente nel team (es. Product Owner o altri ruoli) che possa mediare tra le loro esigenze e le decisioni del progetto.

3. *Involved* (Coinvolti): Gli stakeholder partecipano attivamente al processo e sono informati sullo sviluppo. Sono coinvolti regolarmente in riunioni, review, e attività di feedback, contribuendo alle decisioni e garantendo che il progetto tenga conto delle loro aspettative.

4. *In Agreement* (In Accordo): Gli stakeholder concordano sugli obiettivi del progetto, sulle funzionalità principali e sulle scelte di progettazione. In questo stato, gli stakeholder supportano il piano di progetto e le decisioni strategiche, riducendo il rischio di conflitti in fase di sviluppo.

5. *Satisfied for Deployment* (Soddisfatti per il Rilascio): Gli stakeholder hanno approvato il sistema per il rilascio. Ritengono che il prodotto risponda ai requisiti stabiliti e sia pronto per essere distribuito o messo in produzione. Questo stato è una conferma che il team ha rispettato le aspettative fondamentali e può procedere al rilascio.

6. *Satisfied in Use* (Soddisfatti in Uso): Gli stakeholder sono soddisfatti del prodotto nel contesto reale di utilizzo. Il sistema non solo è stato rilasciato, ma sta funzionando come previsto e risponde efficacemente alle necessità operative e strategiche degli stakeholder, raggiungendo i risultati attesi.

L'*Opportunity* è il problema percepito dagli Stakeholders, ovvero il bisogno da soddisfare col prodotto che verrà sviluppato. I possibili stati sono dell'Opportunity sono: *Identified*, *Solution needed*, *Value Established*, *Viable*, *Addressed*, *Benefit Accrued*.

Gli stati dell'*Opportunity* riflettono la maturazione di un'opportunità di progetto dall'identificazione fino alla realizzazione dei benefici:

1. *Identified* (Identificata): È stato individuato un potenziale bisogno o opportunità. In questo stato, il team ha identificato un problema o un'opportunità di mercato, ma senza ancora un'analisi dettagliata del valore o delle soluzioni possibili.

2. *Solution Needed* (Necessaria una Soluzione): È confermato che esiste una chiara necessità di una soluzione. Qui, il problema o l'opportunità sono definiti con maggior precisione, e vi è consenso sul fatto che affrontarli sia rilevante, anche se la soluzione specifica non è ancora stata decisa.

3. *Value Established* (Valore Stabilito): È stato determinato che l'opportunità offre un valore significativo. In questo stato, il team ha quantificato o dimostrato il potenziale valore della soluzione per gli stakeholder, evidenziando come la soluzione possa soddisfare bisogni chiave o portare vantaggi concreti.

4. *Viable* (Realizzabile): La soluzione proposta è fattibile e sostenibile. Sono stati considerati gli aspetti pratici della realizzazione e i rischi associati, confermando che il progetto è tecnicamente ed economicamente fattibile e ha buone possibilità di successo.

5. *Addressed* (Affrontata): L'opportunità è stata concretamente affrontata attraverso lo sviluppo e il rilascio della soluzione. Il prodotto o sistema risponde all'opportunità identificata e agli obiettivi concordati, ed è pronto per l'uso o l'implementazione.

6. *Benefit Accrued* (Beneficio Raggiunto): I benefici attesi sono stati realizzati e misurati nel contesto reale. In questo stato, la soluzione ha dimostrato di apportare il valore promesso, e gli stakeholder vedono risultati tangibili e vantaggi concreti derivanti dall'opportunità inizialmente identificata.

Questi stati aiutano a tracciare il ciclo di vita di un'opportunità, assicurando che ci sia sempre un chiaro valore e una sostenibilità nel perseguirla e che i benefici finali siano effettivamente conseguiti.

Ad esempio supponiamo che uno stakeholder voglia un nuovo prodotto digitale vendere biglietti di una lotteria. Potremmo ottenere questa sequenza di stati per l'analisi dei bisogni del cliente (*Opportunity*):

1. Un problema identificato (*Opportunity* in stato *identified*) : occorre definire cos'è un biglietto, e a cosa serve. Serve per l'estrazione dei premi? serve per un controllo? Serve come ricevuta? Serve come combinazione di queste o altre funzioni?

   Descrizione: È stata identificata un'opportunità per sviluppare un prodotto di vendita di biglietti della lotteria, rispondendo a una domanda di mercato per semplificare l'acquisto online e ampliare il pubblico.

   Esempio concreto: Durante una riunione di pianificazione strategica, il team di marketing ha presentato dati di mercato che mostrano una crescita della domanda per piattaforme digitali che vendono biglietti di lotteria. È stata avviata una discussione preliminare per esplorare la fattibilità.

2. Un problema analizzato (*Opportunity* in stato *Solution needed*): il biglietto sarà cartaceo o digitale? Deve essere "smarcato"? Deve essere conservato per quanto tempo?

   Descrizione: È stato chiarito che c'è un bisogno reale di una soluzione specifica per vendere biglietti di lotteria in modo sicuro, semplice e conforme alla normativa. È necessario progettare una piattaforma che consenta una buona esperienza utente, la gestione delle transazioni e l'integrazione con il sistema dei distributori di lotterie.

   Esempio concreto: Sono stati svolti alcuni focus group con potenziali utenti e partner per comprendere i requisiti specifici del prodotto. I risultati indicano che gli utenti vogliono acquistare e visualizzare i biglietti in digitale, ricevere notifiche sui numeri vincenti e avere un sistema di pagamento sicuro. Il team tecnico ha iniziato a definire i requisiti funzionali e non funzionali del sistema.

3. Valore confermato (*Opportunity* in stato *Value established*): abbiamo definito un prodotto che permette di vendere vantaggiosamente i biglietti digitali online.

   Descrizione: Il valore dell'opportunità è stato confermato: il prodotto per la vendita online di biglietti di lotteria offre vantaggi sia per gli utenti finali sia per il distributore, con potenziale per aumentare la base clienti e le vendite.

Esempio concreto: È stato completato uno studio di fattibilità che dimostra che la piattaforma non solo può attrarre nuovi utenti ma anche ridurre i costi operativi grazie all'automazione. Il business case è stato approvato dal management, con proiezioni di guadagno che supportano l'investimento iniziale e dimostrano la sostenibilità del progetto.

I *Requirements* sono i requisiti, che di solito nello sviluppo agile si rappresentano mediante epiche e user stories. I *Requirements* possono essere nei seguenti stati: *Conceived*, *Bounded*, *Coherent*, *Acceptable*, *Addressed*, *Fulfilled*.

1. *Conceived* (Concepiti): I requisiti sono stati inizialmente individuati o definiti. Questa è la fase embrionale in cui si ha solo un'idea generale delle necessità o funzionalità da realizzare, senza dettagli o conferme.

2. *Bounded* (Delimitati): I requisiti sono stati delineati e hanno confini chiari. È stata stabilita l'area di applicazione, distinguendo ciò che è incluso nel progetto da ciò che non lo è. Questo stato è fondamentale per evitare ambiguità e tenere sotto controllo lo scope.

3. *Coherent* (Coerenti): I requisiti sono stati organizzati e verificati per assicurare coerenza interna, senza conflitti. Qui si verifica che i requisiti siano allineati tra loro e che non vi siano incongruenze che potrebbero causare problemi durante la progettazione o lo sviluppo.

4. *Acceptable* (Accettabili): I requisiti sono stati validati e accettati dagli stakeholder chiave. Ciò implica che i requisiti soddisfano i bisogni degli utenti e le aspettative del cliente e che sono pronti per essere implementati.

5. *Addressed* (Affrontati): I requisiti sono stati considerati durante la progettazione e lo sviluppo del sistema. Questo significa che il team di sviluppo ha effettivamente lavorato sui requisiti, traducendoli in specifiche funzionali e implementazioni.

6. *Fulfilled* (Soddisfatti): I requisiti sono stati pienamente implementati e verificati nel prodotto finale. Questo stato implica che i requisiti sono stati non solo implementati ma anche testati per garantire che il sistema risponda pienamente a quanto richiesto.

Il *Software System* è il prodotto da costruire. Gli stati possibili di questo alpha sono: *Architecture Selected*, *Demonstrable*, *Usable*, *Ready*, *Operational*, *Retired*.

1. *Architecture Selected* (Architettura Selezionata): È stata scelta un'architettura per il sistema che supporti i requisiti e risponda ai vincoli tecnici. In questo stato, le decisioni architetturali di alto livello sono state prese, dando una direzione chiara per lo sviluppo del sistema.

2. *Demonstrable* (Dimostrabile): Il sistema è in uno stato tale da poter mostrare le prime funzionalità principali. Anche se incompleto, è già possibile dimostrare alcune capacità del sistema, ad esempio tramite un prototipo o una demo iniziale, per ottenere feedback anticipato dagli stakeholder.

3. *Usable* (Utilizzabile): Il sistema è ora sufficientemente sviluppato da poter essere usato in contesti di test o di utilizzo limitato. Le funzionalità chiave sono presenti e funzionali, ma potrebbero mancare componenti secondari o miglioramenti delle prestazioni necessari per un uso esteso.

4. *Ready* (Pronto): Il sistema è completo e pronto per essere rilasciato in produzione. Questo stato implica che il software è stato testato, ottimizzato e approvato dagli stakeholder, ed è idoneo per un'implementazione su larga scala.

5. *Operational* (Operativo): Il sistema è attivo e utilizzato in un ambiente di produzione. In questo stato, il software supporta gli utenti finali in scenari di utilizzo reale e può essere soggetto a monitoraggio e manutenzione per garantire affidabilità e risolvere eventuali problemi operativi.

6. *Retired* (Ritirato): Il sistema è stato ufficialmente dismesso e non è più utilizzato in produzione. Questo stato indica che il software è stato sostituito o non è più necessario, e che le risorse ad esso dedicate sono state riassegnate o disattivate.

Questi stati del sistema aiutano a tracciare il progresso del prodotto dallo sviluppo iniziale fino alla dismissione, assicurando che il sistema sia pronto, stabile e affidabile in ogni fase della sua vita operativa.

Il *Team* è il gruppo di persone che si occupano della costruzione, deployment e manutenzione del Software System. Gli stati possibili del Team sono quelli di Tuckman: *Seeded*, *Formed*, *Collaborating*, *Performing*, *Adjourned*.

Work è un insieme di attività del Team per costruire il Software System. Stati: *Initiated*, *Prepared*, *Started*, *Under Control*, *Concluded*, *Closed*.

Ecco come una tipica istanza di processo Scrum, in particolare il primo sprint, può rappresentare l'evoluzione degli stati del Work secondo Essence:

1. *Initiated* (Iniziato): Il progetto è stato approvato e il team Scrum è stato formato. È stato identificato uno Scrum Master, un Product Owner e i membri del Development Team. È stato anche stabilito un primo obiettivo di prodotto, insieme alla definizione delle linee guida iniziali per il lavoro.

2. *Prepared* (Preparato): Il Product Backlog è stato creato e popolato con le User Stories prioritarie. Durante la pianificazione dello Sprint, il team ha selezionato gli elementi del backlog da completare, ha definito obiettivi di sprint e ha stabilito un piano di lavoro per le prossime due settimane. Il team ha anche preparato gli strumenti

necessari per il tracciamento del lavoro e la comunicazione, come la scrum board o il software di gestione.

3. *Started* (Avviato): Il primo Sprint è iniziato, e il team sta eseguendo le attività in base al piano dello sprint. Ogni giorno si svolge il Daily Stand-Up per aggiornare il team sul progresso del lavoro, discutere gli impedimenti e pianificare le attività giornaliere, assicurando che tutti siano allineati.

4. *Under Control* (Sotto Controllo): Lo Sprint è a metà del suo ciclo e il lavoro è monitorato per assicurare che i task procedano come previsto. Lo Scrum Master gestisce e risolve impedimenti, mentre il Product Owner può adattare la priorità delle storie in caso di nuove informazioni. Questo stato viene mantenuto fino alla fine dello sprint, con il team che continua a monitorare e correggere il lavoro secondo necessità.

5. *Concluded* (Concluso): Lo Sprint è stato completato. Il team ha terminato tutte le User Stories selezionate o ha identificato quelle che devono essere riprese nello sprint successivo. È stata condotta una Sprint Review con gli stakeholder per mostrare i risultati e raccogliere feedback, seguito dalla Sprint Retrospective per riflettere sulle prestazioni del team e proporre miglioramenti di processo.

6. *Closed* (Chiuso): Dopo la retrospettiva, lo Sprint è formalmente chiuso. I dati raccolti (come ad esempio il Burndown Chart) vengono archiviati per la reportistica e il miglioramento continuo delle prestazioni del Team, mentre le lezioni apprese sono documentate e condivise. Il Team chiude ufficialmente il lavoro di questo sprint, rendendosi pronto per il prossimo Sprint.

Questa evoluzione mostra come uno Sprint evolve, tracciando ogni aspetto del lavoro da pianificazione e attivazione a conclusione e miglioramento continuo, seguendo un ciclo iterativo.

Way of Working è il modello del processo di sviluppo che determina le attività da effettuare. Stati: *Principles Established*, *Foundation Established*, *In Use*, *In Place*, *Working Well*, *Retired*.

1. *Principles Established* (Principi Stabiliti): I principi e i valori che guideranno il modo di lavorare sono stati definiti. Questo stato fornisce una base comune di credenze e linee guida che il team userà per prendere decisioni operative e mantenere un approccio coerente nel progetto.

2. *Foundation Established* (Fondamenti Stabiliti): Gli strumenti, le pratiche di base e le metodologie principali sono stati scelti e definiti. Questo stato implica che sono stati scelti framework, strumenti di collaborazione, gestione delle attività e altre risorse essenziali per supportare il team.

3. *In Use* (In Uso): Il modo di lavorare è stato adottato e messo in pratica dal team. Ovvero, il team sta applicando le pratiche, gli strumenti e i processi scelti nella loro attività quotidiana, testando l'efficacia della WoW e adattandola alle loro necessità operative.

4. *In Place* (Consolidato): Il modo di lavorare è completamente integrato nel flusso di lavoro del team. A questo punto, le pratiche sono state adattate e ottimizzate in base ai feedback e alle esperienze del team, assicurando che la WoW sia efficiente e funzionale.

5. *Working Well* (Funzionante): Il modo di lavorare è ben ottimizzato e dimostra di supportare efficacemente gli obiettivi del progetto. In questo stato, il team opera in modo fluido e può rispondere rapidamente a cambiamenti e sfide, dimostrando che la WoW è stata perfezionata e fornisce risultati concreti.

6. *Retired* (Dismessa): La WoW corrente è stata dismessa perché il progetto è terminato o è stata adottata una nuova metodologia. Il team non utilizza più le pratiche e gli strumenti precedenti e potrebbe aver documentato le lezioni apprese per eventuali futuri progetti.

Questi stati descrivono l'evoluzione del modo di lavorare del team, assicurando che sia funzionale e adatto alle necessità del team in ogni fase del progetto.

Le attività che un team esegue possono essere definite anch'esse in modo grafico, come in Fig.1.6. Questa figura mostra tre righe di diversi colori, che includono sequenze di attività, lette da sinistra verso destra.

L'area Customer sulla riga verde, in alto, include queste attività:

1. *Explore possibilities*: Esplorare le opportunità – Consiste nell'analisi di idee e opzioni disponibili per identificare soluzioni che potrebbero essere utili al cliente, valutando anche fattibilità e rischi.

2. *Understand Stakeholder Needs*: Comprendere le necessità degli stakeholder – Si tratta di raccogliere e chiarire le esigenze, i desideri e le aspettative degli stakeholder, assicurando che siano ben comprese da tutti i membri del team.

3. *Ensure Stakeholder Satisfaction*: Garantire la soddisfazione degli stakeholder – Questa attività mira a verificare che le soluzioni sviluppate rispondano alle esigenze degli stakeholder e siano conformi agli standard qualitativi attesi, ricevendo riscontri positivi.

4. *Use the System*: Utilizzare il sistema – Gli stakeholder iniziano a utilizzare il sistema, verificandone l'utilità e il valore in contesti reali, offrendo feedback che può essere utilizzato per ulteriori miglioramenti.

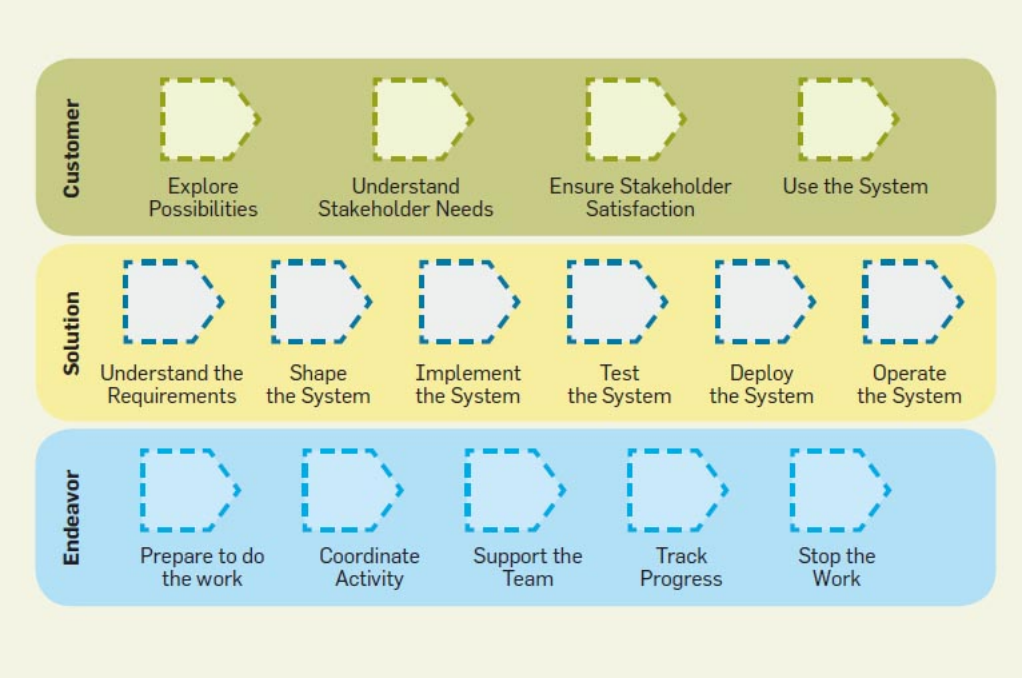

Figura 1.6: Attività tipiche dello sviluppo, organizzate per colori

L'area Solution include le attività sulla riga gialla, al centro:

1. *Understand the Requirements*: Comprendere i requisiti – Analizzare e interpretare i requisiti, sia funzionali che non funzionali, per garantire che il team di sviluppo abbia una visione chiara di ciò che il sistema deve realizzare.

2. *Shape the System*: Modellare il sistema – Definire l'architettura e la struttura del sistema, progettando componenti e interfacce per rispondere ai requisiti in modo efficace e sostenibile.

3. *Implement the System*: Implementare il sistema – Sviluppare il sistema costruendo codice e componenti, traducendo il progetto in una soluzione funzionante e verificabile.

4. *Test the System*: Testare il sistema – Verificare che il sistema funzioni come previsto attraverso attività di test, individuando e risolvendo eventuali difetti per garantire la qualità e l'affidabilità.

5. *Deploy the System*: Rilasciare il sistema – Pubblicare il sistema nell'ambiente di produzione o in un contesto reale, rendendolo disponibile per gli utenti e predisponendo quanto necessario per una distribuzione efficace.

6. *Operate the System*: Gestire il sistema – Assicurarsi che il sistema funzioni correttamente una volta in uso, monitorando le sue prestazioni, risolvendo eventuali problemi operativi e aggiornando il sistema secondo necessità.

Infine, sulla riga in basso, blu, l'area Endeavor include le attività del team:

1. *Prepare to do the Work*: Prepararsi al lavoro – Consiste nel predisporre tutto ciò che è necessario per avviare l'attività, inclusa la definizione degli obiettivi, la pianificazione delle risorse e l'identificazione di potenziali rischi.

2. *Coordinate Activity*: Coordinare le attività – Assicurarsi che le attività del team siano allineate e ben coordinate, facilitando la collaborazione tra i membri e gestendo le dipendenze tra compiti e obiettivi.

3. *Support the Team*: Supportare il team – Offrire supporto al team fornendo risorse, strumenti e assistenza per superare ostacoli e mantenere alta la motivazione e l'efficienza.

4. *Track Progress*: Monitorare i progressi – Seguire l'avanzamento del lavoro rispetto agli obiettivi e alle scadenze, identificando e risolvendo eventuali deviazioni o impedimenti.

5. *Stop the Work*: Concludere il lavoro – Terminare l'attività quando il lavoro è completato o quando non è più necessario, garantendo una chiusura ordinata e, se necessario, documentando i risultati e le lezioni apprese.

1.1 Il kernel di Essence

Il linguaggio kernel di Essence è riassunto nella Fig.1.7. Questa figura mostra le astrazioni principali del metodo e le loro relazioni, utili per modellare processi di sviluppo di software.

La figura include:

a sinistra, i materiali di supporto, che sono in genere conoscenze in forma di pattern;

a destra in alto: le cose con cui lavorare, in particolare gli Alpha, i loro stati, e gli artefatti (Work product);

a destra al centro: le cose da fare, che sono principalmente attività che possono avere delle alternative in forma di pattern; le attività possono aver luogo in uno "spazio di attività";

a destra in basso: le competenze necessarie (simbolo stella a cinque punte) ed i pattern che descrivono i ruoli e l'organizzazione interna dei team;

in basso: le aree in cui si struttura il metodo Essence, ovvero Customer in verde, Solution in giallo, e Endeavor in blu.

Questa figura definisce una grammatica visuale interpretabile da chi conosce i simboli di Essence e ha competenze di sviluppo software. Ad esempio, si evince che: *un Alpha ha*

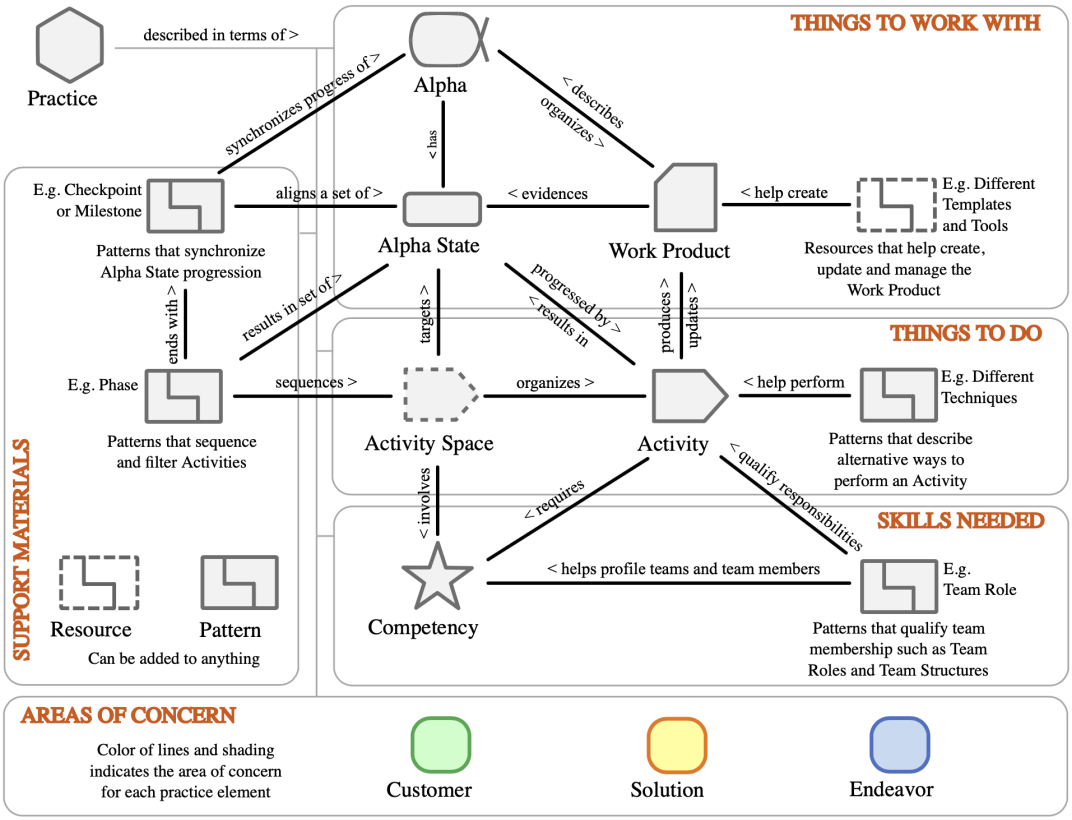

Figura 1.7: Il linguaggio kernel di Essence

uno Stato che risulta in un Work Product che è il prodotto di un'Attività che richiede una Competenza.

1.2 Un esempio

Facciamo un esempio per capire come un team usa le carte durante una retrospettiva.

La fase preliminare prima del primo sprint viene definita "Sprint zero". Possiamo usare le carte "Alpha" per definire uno stato del processo, obiettivo finale dello sprint zero. La Fig. 1.8 mostra una configurazione di alpha utilizzabile come obiettivo dello sprint zero.

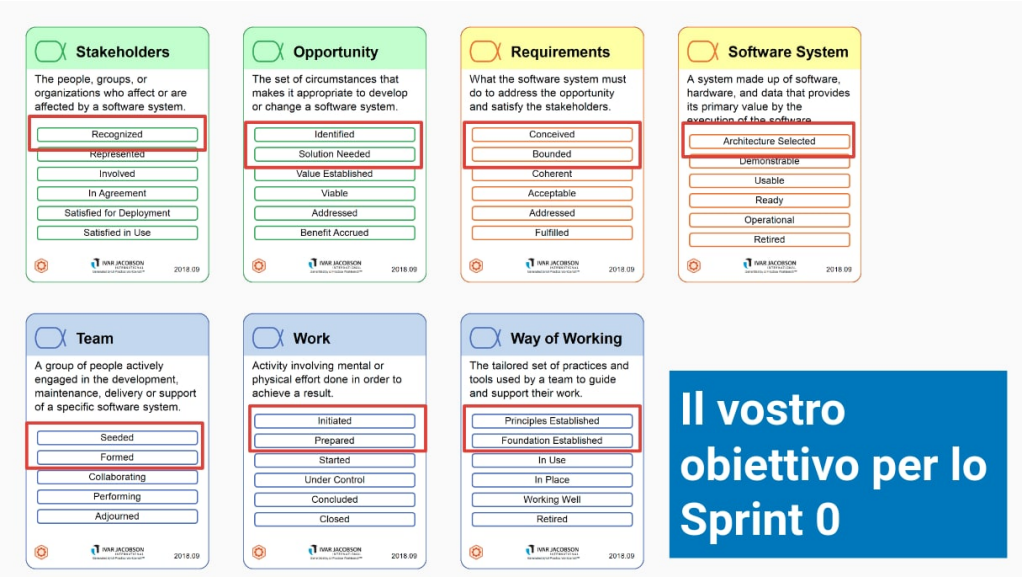

Figura 1.8: Un obiettivo per lo sprint zero - sono mostrate le sette carte "Alpha" principali

Esistono inoltre alcune carte specializzate per processi di tipo Scrum, con elementi quali "I valori Scrum", l'"auto-organizzazione del team", lo "sprint", eccetera. Queste carte possono essere utilizzate per analizzare aspetti specifici del metodo agile prescelto.

1.3 Esempio: autovalutazione del team

Una delle discussioni che il team deve solitamente affrontare durante la retrospettiva del primo sprint riguarda la qualità della collaborazione tra i membri del team: occorre capire se il team sta funzionando bene oppure no, e cosa si può fare per migliorare. L'approccio Essence propone la carta mostrata in Fig.1.9, che riporta la definizione di team ed i possibili stati secondo Tuckman.

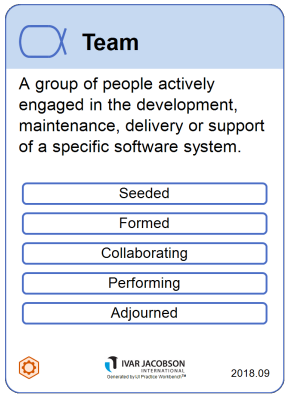

Figura 1.9: Carta dei possibili stati del team secondo Tuckman

Questa è una carta "alpha" dello sviluppo software, cioè un concetto primitivo di primo livello nel gergo SEMAT. Esistono poi, associate a quella, altre cinque carte derivate, di secondo livello, che definiscono meglio i diversi stati del team; le mostriamo in Fig.1.10.

Le cinque carte riguardano i possibili stati del team (secondo la teoria di Tuckman): *seeded*, *formed*, *collaborating*, *performing*, *adjourned*.

Ciascuna carta riporta quali caratteristiche dovrebbe avere il team per classificarsi nello stato che descrive. Ad esempio la terza carta, *collaborating*, si applica se il team "funziona come unica unità, comunica apertamente e sinceramente, è focalizzato sulla propria missione, i membri hanno imparato a conoscere pregi e difetti di ciascun collega".

In pratica le carte Essence costituiscono un riferimento semplice e immediato che permette al team durante la retrospettiva di autovalutarsi rispetto ad alcune nozioni chiave delle pratiche agili. Per esempio la Fig.1.11 mostra che in che misura uno specifico team si riteneva al secondo sprint in stato *collaborating*.

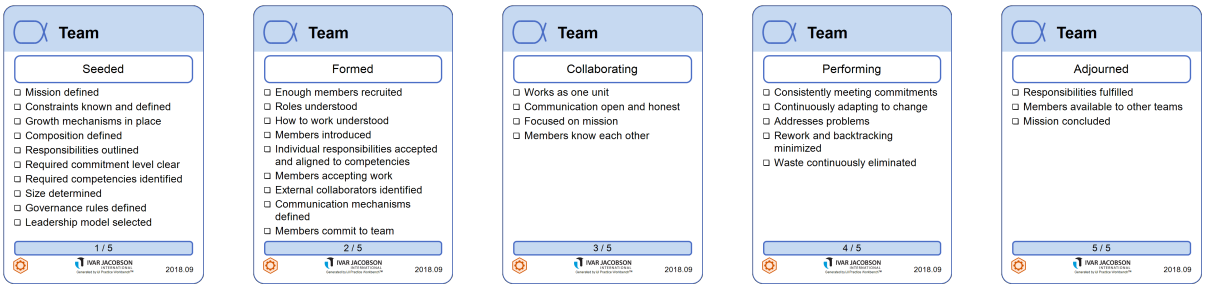

Figura 1.10: Carte che dettagliano i diversi stati di un team secondo il modello di di Tuckman

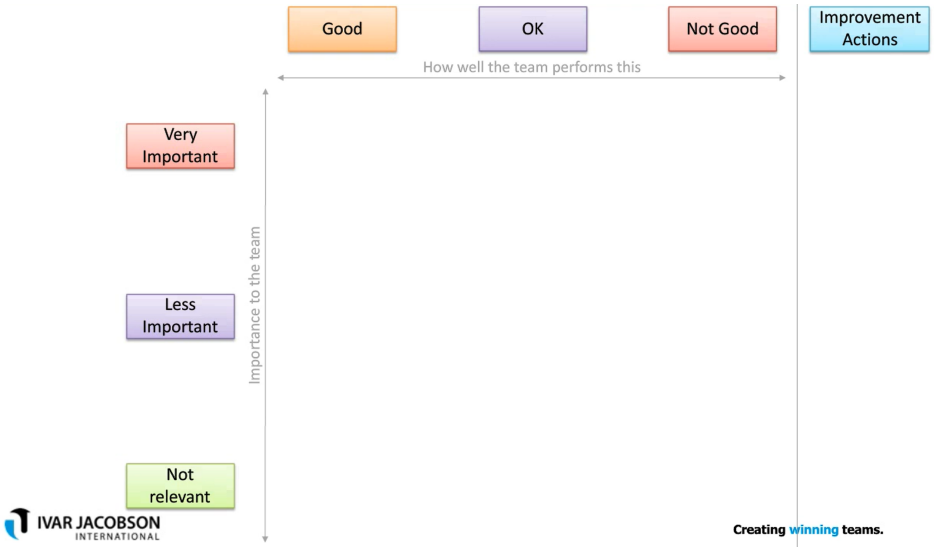

Figura 1.12: Immagine tratta dal video *Essence in Action* al minuto 21:28 url: https://youtu.be/qCCso-a1iC8?t=1288, di Ivar Jacobson e Brian Kerr

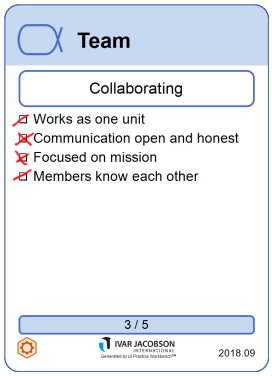

Figura 1.11: Carta che permette di analizzare lo stato *collaborating* di un team

Si osservino i segni rossi: le crocette segnalano che il corrispondete punto obiettivo è stato raggiunto in pieno, mentre le barre mostrano che il punto obiettivo è stato conseguito solo parzialmente. Dunque la carta annotata in figura 1.11 significa:

Works as one unit - parziale, ovvero: funziona parzialmente bene come unità, *Communication open and honest* - pieno, ovvero: il team comunica bene e in modo franco, *Focussed on mission* - pieno, ovvero: il team è focalizzato sui suoi obiettivi di missione, *Members know each other*- parziale, ovvero: i membri non si conoscono ancora abbastanza (e quindi presumibilmente la fiducia reciproca ne soffre).

L'uso di carte Essence durate la retrospettiva è stato suggerito dagli autori del metodo [5]. Nel nostro case study la riflessione sul processo seguito durante lo sprint nella retrospettiva fu formalizzata dal team utilizzando un card board simile a quello mostrato in Fig.1.12.

La disposizione verticale delle carte rispecchia il grado di priorità che i team conferiscono

al concetto espresso dalla carta, mentre la disposizione orizzontale il grado di soddisfazione della realizzazione del concetto.

L'esempio include sulla destra una colonna (intitolata "Improvement Actions") dove il team segna le azioni da intraprendere nel successivo sprint per migliorarsi. Questa è una prassi raccomandabile, che aiuta lo Scrum Master a controllare sprint per sprint se il team dà seguito alle proprie promesse

Nel nostro case study la retrospettiva dei team avveniva come segue. Il team riunito online sceglieva alcune carte tra quelle disponibili (ogni membro una o più carte in modo indipendente), per commentare lo stato del processo scegliendo gli aspetti soggettivamente piu interessanti.

Lo Scrum Master registrava la "mano" risultante dalla discussione in un verbale di retrospettiva, eventualmente aggiungendo commenti.

La Fig.1.13 mostra alcune carte scelte da uno dei team, con alcuni commenti - cioè *Improvement Actions* - in forma di post-it.

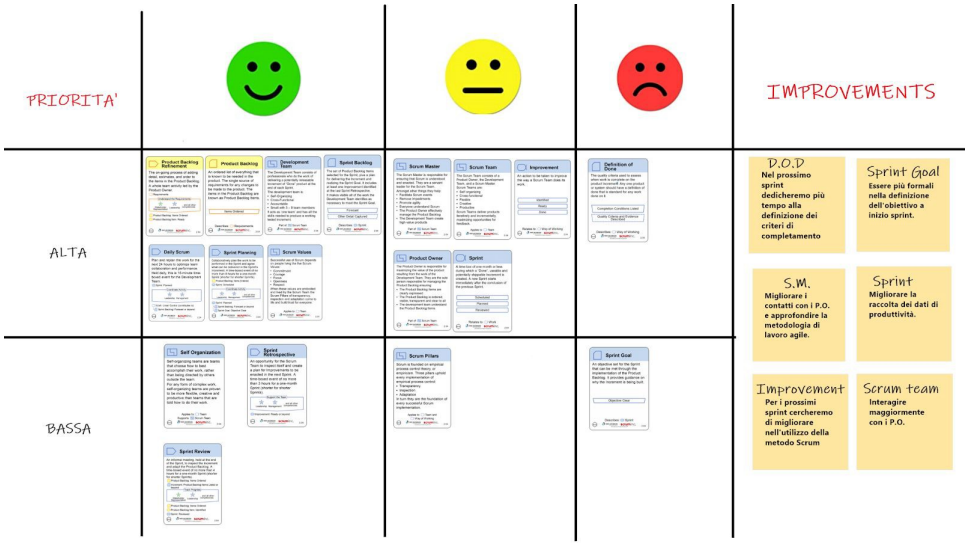

Figura 1.13: Schema che definisce il risultato di una retrospettiva alla fine di uno sprint. La colonna a sinistra, con in cima uno smiley verde, include gli elementi che secondo il team sono acquisiti positivamente. La colonna a destra, con lo smiley rosso, elenca gli elementi negativi che il team deve correggere per il futuro. La colonna mediana con lo smiley giallo riguarda gli elementi di processo che sono in uno stato intermedio tra positivo e negativo. Infine, i post-it nella colonna sulla destra segnalano le azioni che il team si propone di intraprendere nel prossimo sprint.

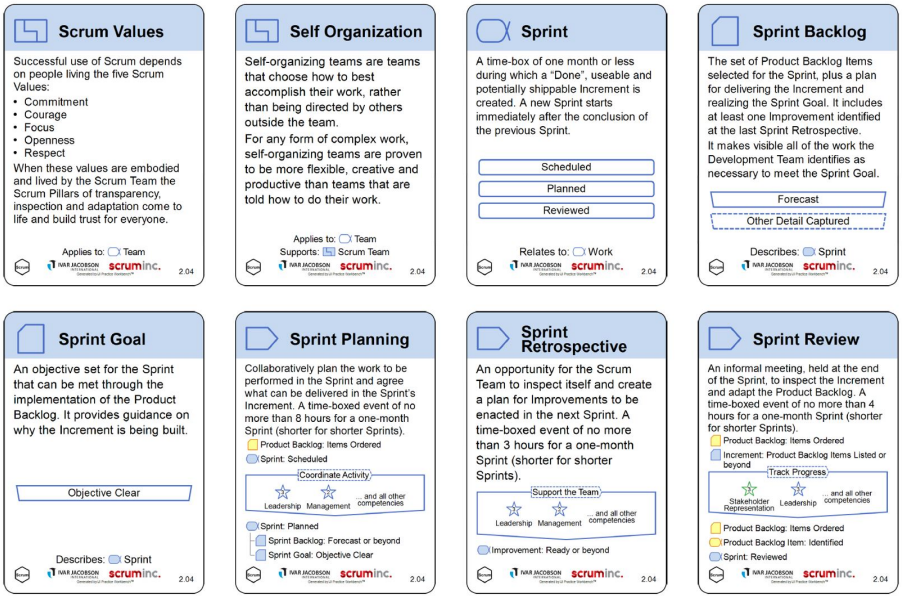

Figura 1.14: Carte per processi di tipo Scrum. Notare che a) sono tutte blu, b) la carta dello Sprint è un Alpha, mentre i valori Scrum e l'autoorganizzazione del team sono pattern.

Nelle Figg.1.14 e 1.15 riportiamo le immagini delle principali carte utilizzate dai team durante il case study.

1.4 Autovalutazione

A questo punto, prima di passare alla lettura del prossimo capitolo, il lettore dovrebbe poter:

- spiegare i concetti di alfa, stato alfa e prodotto di lavoro, e come vengono espressi in Essence;
- identificare i concetti di attività e spazio di attività;
- identificare i concetti di competenza e di pattern;
- spiegare la differenza tra un alfa e un prodotto di lavoro;
- spiegare il concetto di alfa con un esempio (ad esempio, l'alpha dei Requisiti e i suoi stati) e come questi vengono espressi;
- spiegare il tipo di informazioni presenti nelle carte Essence;
- spiegare i benefici delle checklist nell'analisi dei processi di sviluppo;
- elencare i passaggi necessari per essenzializzare un processo come Scrum o una pratica agile di sviluppo.

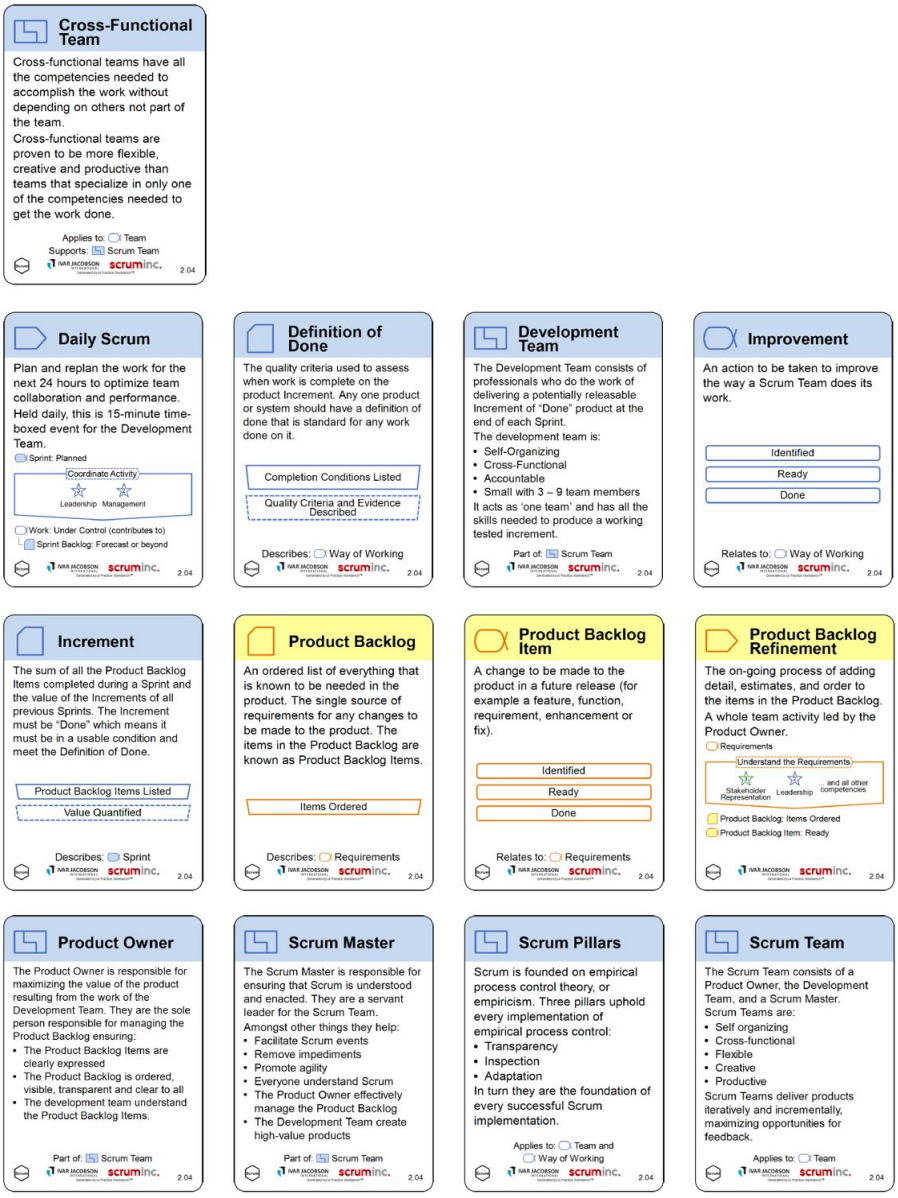

Figura 1.15: Carte per processi di tipo Scrum. Qui ci sono due alpha: l'Improvement ed il Product Backlog Item.

Capitolo 2

Giochi di retrospettiva con Essence

Gli elementi Essence riportati nella carte hanno lo scopo di facilitare la comunicazione, la collaborazione e la riflessione collettiva all'interno di un team di sviluppo. Ogni carta rappresenta un elemento chiave del processo di sviluppo e viene utilizzata nel corso di sessioni di pianificazione o retrospettiva per aiutare i membri del gruppo a discutere sugli aspetti critici in modo strutturato.

In questo capitolo verrà in particolare spiegata l'importanza dell'adozione di Essence e delle sue carte nel contesto di redazione di una retrospettiva. Quest'ultima prevede che il team analizzi lo sprint appena portato a termine, al fine di identificare perfezionamenti da apportare nel prossimo sprint. L'uso di Essence in questo contesto può migliorare notevolmente l'efficacia delle retrospettive, mettendo a disposizione un linguaggio condiviso e strumenti utili ad analizzare il processo di sviluppo agile.

Verranno illustrati una serie di Serious Games da impiegare come esercizi pratici per iniziare a capire come condurre sessioni di retrospettiva produttive utilizzando il modello Essence. Attraverso l'uso dei giochi, i team possono sperimentare in modo interattivo e coinvolgente le pratiche e i principi chiave di Essence, acquisendo familiarità con le carte.

L'utilizzo di Essence per coordinare la retrospettiva è utile ad approfondire la comprensione e l'applicazione dei principi Agili in un contesto ludico. Grazie ad esso si riescono a combinare elementi caratteristici di Scrum, attività utile a simulare le varie situazioni di sviluppo software in un ambiente controllato e di gioco, con l'approccio sistematico proposto da Essence per valutare e migliorare le pratiche di sviluppo.

Le carte fornite da Essence consentono di esplorare e conoscere i vari aspetti del lavoro del team, suddividendoli in categorie chiave come Alphas, Attività e Pattern. Questo facilita il processo di autovalutazione e di analisi durante la sessione di retrospettiva. Il modello inoltre incoraggia la partecipazione attiva di ogni membro del gruppo nell'analisi retrospettiva del proprio lavoro. Ogni carta mette a disposizione un punto di partenza per avviare una discussione, consentendo a tutti di esprimere le proprie opinioni e di contribuire al processo decisionale collettivo.

Usare il framework in corsi per l'educazione dei Product Owner ha mostrato come, mediamente, i team implementino correttamente solo un terzo delle componenti di Scrum, un terzo in modo scarso ed il restante non viene implementato affatto. Per questo motivo, grazie all'uso di giochi e attività che favoriscono l'auto-riflessione e l'apprendimento, l'adozione di Essence può portare ad un miglioramento significativo dell'efficacia con cui i gruppi implementano le task di Scrum [3].

2.1 Team Status Game

Team Status Game rappresenta il primo approccio al modello Essence. Questo gioco, infatti, è basato sull'uso delle carte fornite da Essence per condurre retrospettive efficaci, valutando collettivamente lo stato di avanzamento del team.
Durante la partita, ogni membro del gruppo sceglie una carta che rappresenta lo stato attuale del team. Se tutti i singoli individuano la stessa carta allora si può passare allo strato successivo, altrimenti si avvia una discussione dove ognuno espone le proprio motivazioni sulla scelta appena compiuta.
Una volta che si è raggiunta l'unanimità sulla carta, si va ad osservare la checklist presente in essa; in segreto, ogni partecipante deve selezionare un numero (inferiore al numero di elementi presenti nella checkbox della carta). Si scoprono, quindi, i numeri e, cominciando da chi ha quello più basso, ogni membro del gruppo indica quali compiti, tra quelli presenti nella carta, sono stati portati a termine.
Si prosegue finchè tutti hanno 0 voti. A questo punto, se ogni voce nella checkbox è stata valutata, significa che il team è già nello stadio successivo, altrimenti bisogna controllare cosa non è stato spuntato, avviando una riflessione per progettare un piano d'azione.

Team Status Game stimola la riflessione condivisa, portando il team a valutare le proprie prestazioni e promuovendo una comprensione condivisa degli obiettivi di miglioramento, rendendo il processo di retrospettiva più coinvolgente e produttivo.

2.2 Alpha State Cards Game

Il gioco Alpha State Card Game è stato progettato con lo scopo di guidare il team attraverso una valutazione strutturata dei vari aspetti del processo di sviluppo software rappresentati dalle carte Alpha, identificando le aree di forza e le opportunità di miglioramento.

Lo Scrum Master da inizio alla partita in quanto ha il compito di identificare delle carte rilevanti per il gruppo. Quindi, i membri del team scelgono, tra le carte selezionate, quelle che raffigurano gli aspetti critici del processo. Si avvia, in seguito, una discussione in cui vengono individuate le aree che necessitano di maggiore attenzione e quelle che, invece, rappresentano un punto di forza per il gruppo. A questo punto, il team sviluppa un

piano d'azione che include attività specifiche, responsabilità assegnate e scadenze, al fine di garantire il progresso e il monitoraggio delle azioni di miglioramento.

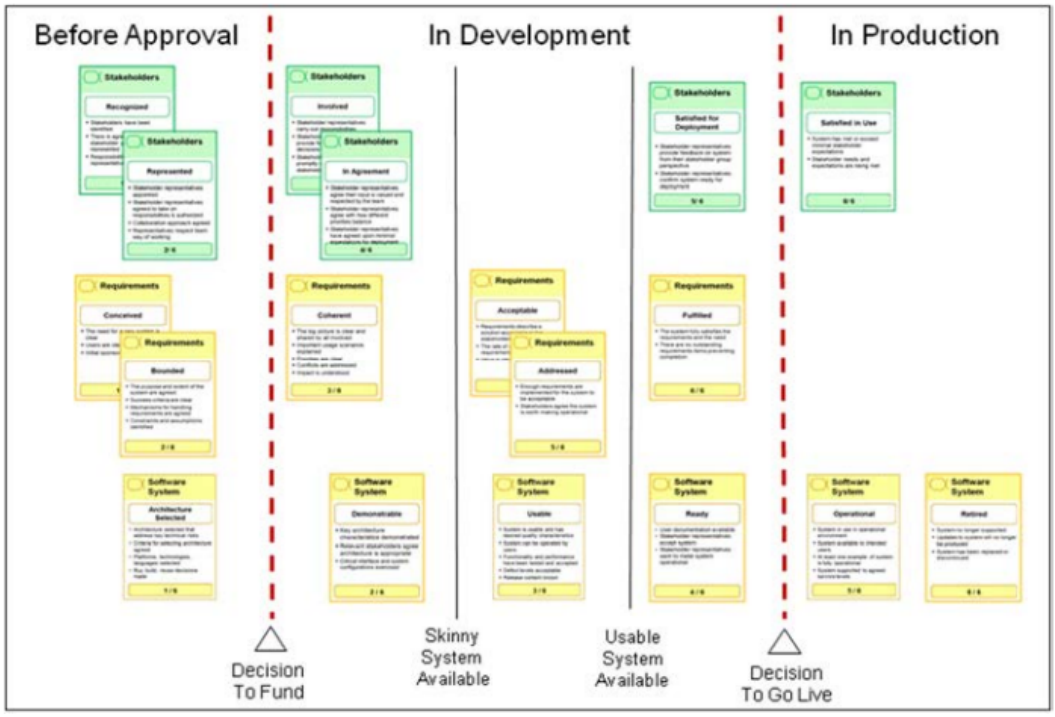

Figura 2.1: Alpha State Cards Game

Essence fornisce una base solida per la strutturazione e la conduzione di Alpha State Card Game, rendendo la valutazione dello stato del team più chiara e focalizzata. Le carte Alpha offrono una rappresentazione chiara degli elementi chiave da prendere in considerazione e le checklist associate aiutano ad avere una guida dettagliata utile all'analisi.

Inoltre, Essence favorisce la standardizzazione e la condivisione delle pratiche, permettendo al gruppo di adottare un linguaggio comune per discutere le prestazioni. Questo porta ad ottenere una migliore comprensione e collaborazione tra i membri del team, facilitando una valutazione più accurata e approfondita dello stato attuale e la definizione di azioni di miglioramento mirate.

2.2.1 Mad Sad Glad

La Fig.2.2 mostra un tabellone creato da un team a popolato con carte Essence.

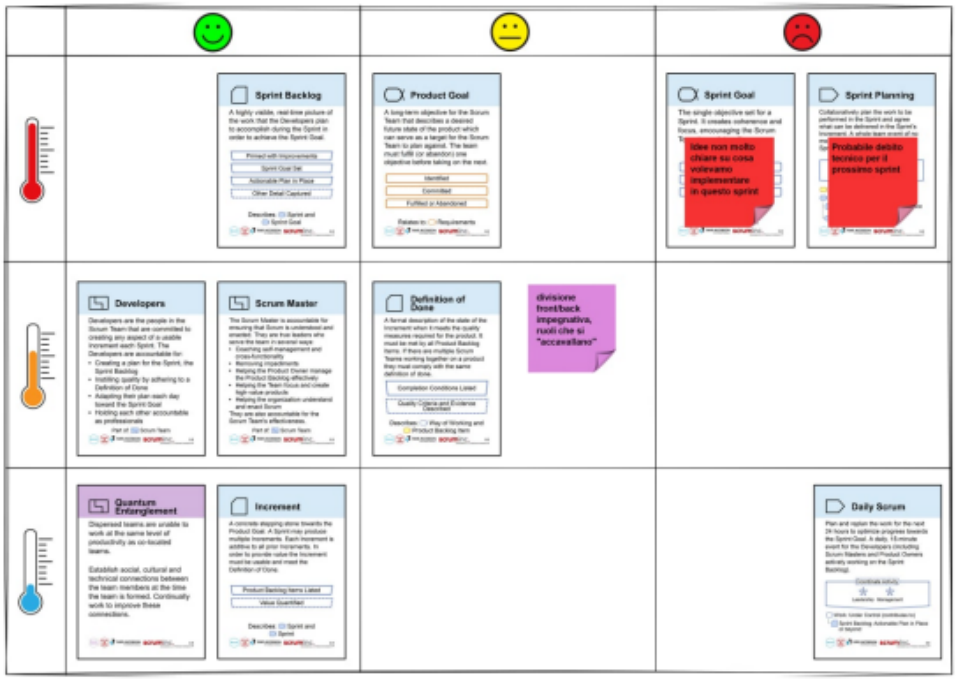

Figura 2.2: Tabellone di una retrospettiva con carte Essence

Le carte vengono scelte, posizionate, spostate, ordinate o raggruppate durante sessioni collaborative durante le quali il team prende decisioni o riflette sul proprio modo di cooperare usando le checklist mostrate dalle carte che vegono scelte dal team stesso.

Viene creato un tabellone bidimensionale in cui le colonne rappresentano quanto sia importante quel concetto per il gruppo, mentre le righe fanno riferimento a quanto bene la squadra sente di essersi comportata in quel determinato ambito. Il team, quindi, posiziona le carte sulla tavola in modo appropriato e le muove mentre discute e ne analizza il funzionamento [3].

Le carte in alto a destra sono quelle che richiedono maggiore attenzione in quanto considerate importanti e migliorabili (nel caso in cui ce ne fossero molte, si può assegnare una priorità, scegliendo su quali di esse focalizzarsi). Infine, i membri del gruppo approfondiscono la natura dei problemi e vengono proposte delle attività che possono essere usate nell'iterazione successiva.

Il contenuto delle carte riassume brevemente le informazioni più importanti contenute all'interno della guida di Scrum. Questo aiuta ad evitare eventuali discussioni o conflitti, dovuti ad interpretazioni personali di un concetto, da parte dei singoli membri del gruppo.

Organizzare in questo modo la retrospettiva permette ai team novizi di avere una

guida per il primo approccio a Scrum, aiutandoli a capire i diversi meccanismi che lo caratterizzano. Inoltre, spinge il team ad agire e a pianificare nel dettaglio i lavori che possono portare ad un miglioramento concreto.

2.3 Siti utili su Essence

- `http://www.software-engineering-essentialized.com`
- `https://www.essify.com`
- `https://www.ivarjacobson.com/publications/blog/better-scrum-through-essence`

Bibliografia

[1] I. Jacobson et al. *The essentials of modern software engineering.* ACM and Morgan & Claypool, 2019.

[2] I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, and S. Lidman. The essence of software engineering: the SEMAT kernel. *Commun. ACM*, 55(12):42–49, 2012.

[3] I. Jacobson, J. Sutherland, B. Kerr, and B. Buhnova. Better Scrum through Essence. *Software: Practice and Experience*, 52(6):1531–1540, 2022.

[4] Object Management Group. Essence – Kernel and Language for Software Engineering Methods version 1.2. Technical Report formal/18-10-02, OMG, 2018.

[5] J. Sutherland, I. Jacobson, and B. Kerr. Scrum essentials cards: experiences of Scrum teams improving with Essence. *Queue*, 18(3):83–106, 2020.