Sprint 0

Sommario

Introduzione

L'obiettivo principale dello sprint 0 è quello di realizzare un'analisi dei requisiti forniti dal committente, seguito da quello di definire un piano di lavoro che andrà a specificare il numero di sprint totali, con relativi obiettivi di ciascuno, necessari alla finalizzazione del sistema. In sostanza si cercherà di formalizzare i requisiti in modo da chiarire cosa sarà necessario sviluppare nel sistema, considerando i componenti già forniti dal committente e già posseduti per delineare quali parti saranno da sviluppare da zero e quali realizzabili attraverso il materiale fornito. A seguito dell'analisi dei requisiti e della loro suddivisione in base alle priorità del committente, verrà delineata una prima struttura di architettura del sistema descrivendo i macrocomponenti e le interazioni tra essi che risultano evidenti direttamente dai requisiti.

Requisiti

I requisiti richiesti dal committente sono consultabili al seguente link.

Di seguito vengono riportati tutti i requisiti in una lista in modo che possano essere consultati in maniera più comoda durante il processo di sviluppo.

1. Deve essere possibile registrare i prodotti presso un productservice. Il prodotto registrato dovrà essere memorizzato su un database.

2. Per la registrazione di un prodotto è necessario specificare il peso ed eventualmente un nome (il nome del prodotto non è specificato esplicitamente nei requisiti ma può essere utile). L'aggiunta di un prodotto deve comportare l'associazione di un *"unique product identifier"* PID (PID > 0).

3. Sono presenti 4 slot in cui poter depositare i container (disposti come da figura) (in uno slot ci può stare soltanto un container).

4. Lo slot 5 è perennemente occupato, perciò non utilizzabile dal sistema.

5. È necessario prevedere un sonar che, dopo che i prodotti sono stati registrati, rilevi la presenza di un container di un prodotto da caricare e inneschi il processo di caricamento. Nello specifico il sonar rileva un container se misura una distanza D tale che D < DFREE/2 per almeno 3s (DFREE distanza rilevata in caso di spazio libero davanti al sensore) (eventualmente il tempo di 3s può essere modificato).

6. Il cargoservice deve poter rilevare la richiesta di carico (mediante il sonar) di un container relativo a un prodotto già registrato sul product service (e ancora non caricato).

7. La richiesta di carico è rifiutata se tutti e 4 gli slot sono occupati oppure se il peso totale della stiva, con l'eventuale aggiunta del nuovo container, superi un parametro MaxLoad (MaxLoad>0).

8. In caso di richiesta accettata il cargoservice associa il PID del prodotto da caricare allo slot di destinazione e comanda al cargorobot di caricare il container sullo slot specificato.

9. Il cargorobot deve raggiungere l'IOPort, prelevare il container, raggiungere lo slot corrispondente, rilasciare il container e, infine, tornare alla HOME. Il cargoservice deve garantire che tutto il processo avvenga in maniera corretta

10. Il cargoservice non serve nuove richieste dal momento in cui viene associato il prodotto allo slot fino a quando il cargorobot non lo ha portato nello slot corrispondente.

11. In caso arrivi e venga accettata una richiesta tra il rilascio del container e il raggiungimento della HOME da parte del cargorobot, quest'ultimo non dovrà più tornare alla HOME ma andare a prendere il container all'IOPort.

12. È necessaria una web-gui aggiornata dinamicamente che mostri lo stato della stiva.

13. È necessario interrompere qualsiasi attività nel caso il sonar rilevi una distanza D > DFREE per almeno 3s. In tal caso deve essere acceso un led per segnalare lo stato di errore. Tutte le attività dovranno ricominciare solamente dal momento in cui venga rilevato D <= DFREE.

Classificazione dei Requisiti

Nella seguente tabella viene riportata una classificazione dei **requisiti funzionali** (specifiche che descrivono che cosa deve fare il sistema) in base alla loro priorità.
I requisiti con priorità più alta sono quelli relativi al core-business (cargoService, productService, cargoRobot), quelli a priorità media sono relativi al sonar e all'IOport, mentre quelli a bassa priorità sono riferiti alla GUI.

PRIORITÀ	REQUISITI	MOTIVAZIONE
Alta	1, 2, 6, 7, 8, 9, 10, 11	Sono requisiti relativi al core-business del programma, senza di essi il sistema non funzionerebbe
Media	5,13	Senza di essi il sistema non può consegnare i prodotti nello slot riservato corretto
Bassa	12	Il sistema funziona anche senza la presenza di questi requisiti, che rappresentano un servizio aggiuntivo (web-gui).

I requisiti mancanti in questa classificazione (3,4) sono i **requisiti non funzionali**, cioè quelli che descrivono come il sistema deve essere e non come deve funzionare.

Vocabolario

- **Compagnia** (*company*) [non rilevante]: compagnia marittima di carico.
- **Stiva** (*hold*) [oggetto]: stiva della nave. Caratterizzata da lunghezza, altezza e posizione degli ostacoli, deve essere rappresentata come una sorta di "mappa"
- **Cargorobot** (*cargorobot*) [attore]: si tratta di un Differential Drive Robot (DDR), un robot mobile che si muove usando due ruote motrici indipendenti, variando la velocità di ciascuna per andare dritto, girare o ruotare su sé stesso. Nel sistema che si vuole sviluppare, è l'entità responsabile del trasporto dei container agli slot assegnati.
- **Prodotti** (*products*) [oggetti]: Merci presenti all'interno dei container e trasportate dal cargorobot. Caratterizzate da PID, peso e nome.
- **Productservice** [servizio]: Servizio che registra il prodotto assegnando ad esso un ID univoco e tiene traccia del peso. Le informazioni relative ad ogni prodotto sono memorizzate in un database.
- **PID** (*product identifier*) [variabile/attributo (del prodotto)]: identificatore univoco di un prodotto.
- **Slot** [variabile/attributo (della stiva)]: Postazione dove vengono depositati i container. Rappresenta una sottoarea della stiva.
- **IOPort** [attore]: punto di accesso fisico/hardware di un container. Dovrà interagire con il cargoservice e rilevare situazioni di allarme
- **Container** [non rilevante/semplice contenitore del prodotto]: unità di carico di cargoservice, dotata di un peso
- **Cargoservice** [servizio]: servizio, omonimo del sistema, che si occupa dell'assegnazione degli slot e del carico dei container

- **MaxLoad** [variabile/costante]: peso massimo (espresso in Kg) che può essere trasportato dalla nave
- **HOME** [variabile/attributo (della stiva)]: locazione di partenza del cargorobot all'interno della mappa, al quale ritorna dopo aver finito di allocare un container nello slot designato
- **Web-gui** [servizio]**:** interfaccia grafica web che permette di visualizzare lo stato della stiva della nave
- **Sensore/sonar** (sensor/sonar) [attore]: sensore che rileva la presenza di un container quando misura una distanza inferiore a DFREE/2 per un tempo totale ragionevole (esempio 3 secondi)
- **Led** [attore]: lampadina che si accende nel caso in cui la distanza rilevata dal sonar sia superiore a DFREE o si siano verificati errori relativi al sonar
- **DFREE** [variabile/costante]: distanza massima che può essere rilevata dal sonar senza la cessazione delle attività del servizio.

Punti aperti

- La richiesta di caricamento di un container deve partire automaticamente quando viene messo un container davanti all'IOPort o serve l'azione di un'altra entità per innescarla (umano o macchina)?
- Quando il sonar rileva un container, come fa a capire di quale prodotto registrato è? Può essere registrato solo un prodotto non ancora caricato? Oppure prende il primo che è stato registrato? Oppure glielo indica un utente in qualche modo?
- Nel caso in cui venga preso il primo prodotto memorizzato, se esso non ci sta per il peso lo scarta e va a quello dopo o si blocca?
- Oppure dato che sonar e cargoservice sono distinti, si prevede invece un comportamento come descritto di seguito? Il sonar rileva il container, questo viene processato o meno dal cargoservice, e in ogni caso il sonar "segna" quel prodotto come visto e in caso di nuova rilevazione prenderà il prodotto successivo?
- Le richieste di carico di container durante il periodo in cui il cargoservice è inibito dal gestirle devono essere rifiutate? O accodate? E in caso accodate possono esserne accodate più di una? In caso di rifiuto, il sonar deve segnare eventualmente come "visti" quei prodotti o no (facendo riferimento al punto precedente)?
- Come vengono registrati i prodotti dagli utenti sul productservice? Mediante gui? Ovvero quale entità (macchina o umano) ha il compito di inserire i prodotti?

Modello logico

Dall'analisi dei requisiti è emerso come sia necessario modellare alcune entità come **servizi** e/o **attori**, quindi come entità attive che possono comunicare tra loro attraverso **scambio di messaggi**. Inoltre, questi attori/servizi presumibilmente saranno situati su macchine diverse (ad esempio i servizi su un

server specifico, l'IOPort all'entrata della stiva e il cargorobot in un altro posto legato al robot), il che si traduce direttamente nella necessità di prevedere un **sistema distribuito** ed **eterogeneo** (per via del fatto che i vari componenti sono di natura diversa gli uni dagli altri)

Fatte queste considerazioni, la modellazione del sistema ci pone davanti a un **abstraction gap** che è necessario colmare. Ciò è dovuto al fatto che la maggior parte dei linguaggi di programmazione (che quindi permettono una sorta di "modellazione eseguibile" non hanno costrutti nativi che permettono di rappresentare un attore che interagisce sulla rete attraverso scambio di messaggi. A tal fine può essere utile utilizzare un linguaggio custom che permetta di modellare al meglio questi contesti, ad esempio un **DSL** come **qak.**

Scelta di QAK

Per realizzare una prima architettura logica riportata in seguito, si è scelto di utilizzare il metamodello QAK per due motivi.

Il primo è che esso permette la descrizione di modelli che sono automaticamente implementabili in software, ciò permette, quindi, di agevolare e velocizzare l'intero processo di sviluppo.

Il secondo deriva dal fatto che il metamodello è provvisto del concetto di attore, cioè un'entità con un proprio flusso di controllo, quindi attiva, che interagisce con le altre entità del sistema attraverso lo scambio di messaggi. L'utilizzo degli attori rende possibile ridurre l'abstraction gap tra gli strumenti di modellazione e ciò che bisogna modellare: se si utilizzasse il metamodello UML, si otterrebbe una modellazione efficace dal punto di vista concettuale e formale per un modello basato sugli oggetti, ma non direttamente eseguibile e non adatta a un modello basato su attori o in generale entità autonome.

La scelta del QAK è quindi legata alla possibilità di realizzare uno schema più formale dei requisiti in questo sprint, e di utilizzarlo come linguaggio di rapida prototipazione negli sprint successivi, al fine di poter fornire al committente una porzione di software funzionante al termine di ogni sprint. In questo caso si parla di attori nel contesto QAK quindi li consideriamo entità autonome che gestiscono stato e comportamento, comunicano e reagiscono a eventi in modo concorrente ma sicuro.

Entità

Nella tabella seguente sono riportate le principali entità del sistema, descritte nei requisiti, che possono essere considerate come autonome e interagenti mediante messaggi. Tali "macro-entità" sono indicative e servono a dare l'idea dei componenti che presumibilmente sarà necessario sviluppare negli sprint successivi, durante i quali verranno analizzate più approfonditamente.

Il sistema si dispone bene per essere sviluppato eventualmente su più nodi; quindi, queste entità potrebbero potenzialmente essere gestite anche su nodi separati definendo un'architettura adeguata, potenzialmente distribuita, ad esempio una basata su microservizi.

Attori	**Ruolo**	**Comportamento**
Cargoservice	Reattivo/Proattivo	Gestisce le richieste di carico di container Recupera i prodotti dal productservice e li associa agli slot dove dovranno essere caricati

		Mediante il cargo robot gestisce il caricamento concreto dei container Inoltre, invia aggiornamenti all'interfaccia web sullo stato della stiva
Productservice	Reattivo/Proattivo	Permette di registrare i prodotti associando ad essi un PID univoco Permette di recuperare le informazioni relative a tali prodotti
IOPort	Proattivo	Rileva costantemente misurazioni sullo stato di occupazione dell'IOPort Deve essere in grado di rilevare la presenza di un container presso l'IO-port. Deve essere in grado di rilevare e comunicare un malfunzionamento e la sua risoluzione al fine di bloccare il sistema in tal caso
Cargorobot	Reattivo/Proattivo	Quando una richiesta di caricamento viene accettata deve muoversi dentro la stiva raggiungendo prima l'IOPort e successivamente lo slot corretto in cui caricare il container
Web-gui	Reattivo	Entità che riceve i dati dal cargoService e aggiorna la GUI, serve per monitorare lo stato della stiva da parte di operatori umani
Sonar	Proattivo	Attore che interagisce con il mondo esterno e manda informazioni relative alla presenza di un container e alla rilevazione di anomalie
Led	Reattivo	Attore che durante un malfunzionamento mostra tramite un led fisico il malfunzionamento al mondo reale

Messaggi

Nella seguente tabella viene riportato l'elenco dei messaggi che si possono dedurre dai requisiti. Essi sono scambiati fra entità e le loro caratteristiche verranno specificate e definite in maniera dettagliata negli sprint successivi. Questa è una descrizione parziale dei messaggi che saranno necessari per il corretto funzionamento del sistema. Questa tabella verrà raffinata attraverso le analisi del problema negli sprint successivi. Di seguito per "entità non specificata" si intenderà un'entità che al momento non è chiaramente deducibile dai requisiti in modo letterale. Questa in futuro potrà essere identificata come un attore completamente esterno al sistema oppure come un componente stesso del sistema, ma tale scelta richiederà una specifica analisi successiva

Messaggio	Tipo	Inviato da	Ricevuto da
RichiestaCarico	Request	Entità non specificata	Cargoservice
RichiestaCaricoAccettata	Reply (to Richiesta carico)	Cargoservice	Entità non specificata
RichiestaCaricoRifiutata	Reply (to Richiesta carico)	Cargoservice	Entità non specificata
RilevazioneAnomalia	Event	IOPort	Tutti i componenti che dovranno essere soggetti al blocco a seguito dell'allarme
RisoluzioneAnomalia	Event	IOPort	Tutti i componenti che dovranno essere soggetti al blocco a seguito dell'allarme
RegistrazioneProdotto	Request	Entità non specificata	Productservice
EsitoRegistrazioneProdotto	Reply (to Registrazione prodotto)	Productservice	Entità non specificata

Parametri del sistema

- **DFREE**: distanza massima (in cm) che il sonar può rilevare prima che il sistema consideri il sensore guasto.
- **MaxLoad**: peso massimo in kg che può essere caricato nella stiva.
- **SLOTS**: n° slots totali della stiva (design for change).

Componenti forniti dal committente

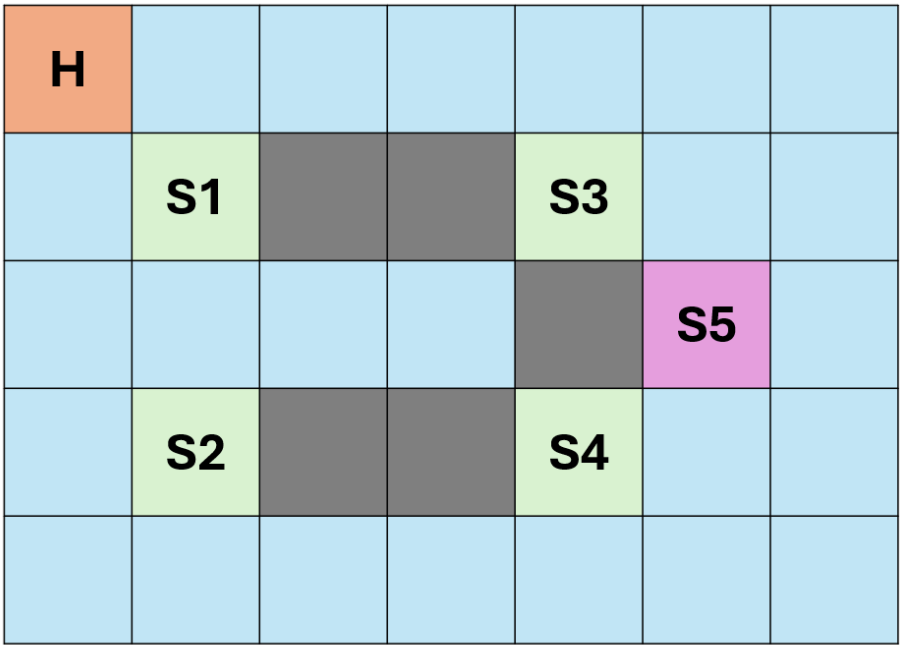
I seguenti componenti software sono forniti funzionanti dal committente, pertanto potranno essere utilizzati senza dover essere implementati

- **VirtualRobot:** simulatore del robot presente nella stiva. Permette di eseguire i movimenti del cargorobot reale in un ambiente di test virtuale
- **BasicRobot:** servizio che permette di interagire con **VirtualRobot** a un livello di astrazione più alto. Questo componente non mappa perfettamente i requisiti del cargorobot, ma offre alcune

funzionalità che potrebbero essere utili per alcuni dei requisiti. Nello specifico offre funzionalità di spostamento del robot ad alto livello tra cui:

- Avanzare e retrocedere di un passo (grande come la grandezza del robot)
- Ruotare a destra e a sinistra
- Andare da un punto all'altro della mappa calcolando autonomamente il percorso più breve
- Interrompere il movimento del robot

- BasicRobot come **modello della stiva**: il BasicRobot ha anche integrato un modello della stiva realizzato come mappa di celle della grandezza del robot (in "unità robotiche"). Questa mappa rappresenta già in modo specifico la stiva oggetto del problema. Dato che ciò corrisponde perfettamente ai requisiti descritti in termini di modellazione della mappa, è possibile utilizzare direttamente questo componente risparmiando il lavoro legato allo sviluppo del software per modellare la stiva. Di seguito viene riportata una raffigurazione della stiva come modellata dal BasicRobot.



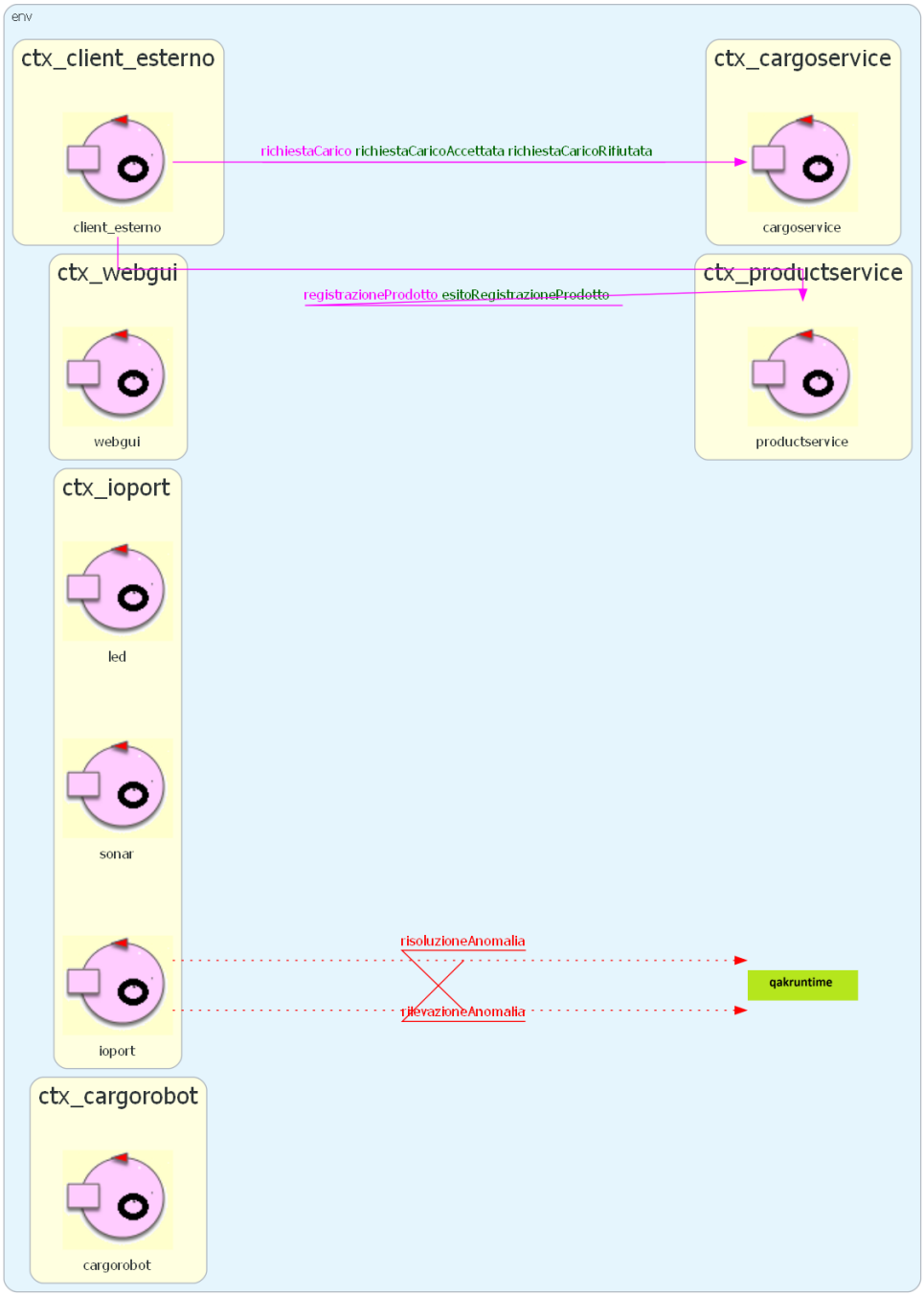
IOPort

- **Productservice:** il componente productservice è fornito come container docker e come codice attraverso il progetto nominato "cargoserviceqak" (si noti, nonostante l'errore del nome che tale progetto svolge esattamente le funzionalità richieste per pproductservice)
- Porzioni di codice per la gestione fisica del sonar e del led relativi all'IOPort

Inoltre, sono disponibili le seguenti componenti interne che possono essere utilizzate nel processo di sviluppo:

- Linguaggio qak con appositi plugin per eclipse
- unibolibs: librerie varie, tra cui librerie di comunicazione

Architettura



cargorobotArch

Piano di test

Il piano di test si basa sul collaudo delle funzionalità base che sono state dedotte in maniera diretta dai requisiti del committente. Abbiamo infatti pensato ai seguenti test in questa prima fase:

- **TestRichiestaCaricoOK()**: si occupa di testare se la richiesta di carico è andata a buon fine.

- **TestRichiestaCaricoNoSlotLiberi():** si occupa di testare il fatto che la richiesta non può essere accettata a causa degli slot pieni.
- **TestRichiestaCaricoOltrePesoMax():** si occupa di testare il fatto che la richiesta non può essere accettata perché è stato raggiunto il peso massimo nella stiva.
- **TestRichiestaCaricoProdNonPres():** si occupa di testare il fatto che la richiesta non può essere accettata perché il prodotto non è presente.

Piano di lavoro

Oltre a questo sprint 0 iniziale, dove andiamo ad analizzare i requisiti e formalizzare i singoli termini del testo, abbiamo deciso di sviluppare gli sprint successivi in base alla priorità dei requisiti analizzati in precedenza:

- Sprint 1: Core-business, requisiti ad alta priorità necessari per il funzionamento del sistema.
- Sprint 2: IOport, requisiti con priorità media relativi alla corretta consegna dei prodotti.
- Sprint 3: Web GUI, requisiti a priorità bassa relativi allo sviluppo di interfaccia web.

SPRINT	INIZIO	FINE	ORE
Sprint 0	15/09/2025	19/09/2025	20
Sprint 1	22/09/2025	03/10/2025	40
Sprint 2	06/10/2025	10/10/2025	20
Sprint 3	13/10/2025	17/10/2025	20

Team di lavoro e attività specifiche

Brighi Valerio	0001189577
Guiducci Daniele	0001189894
Zoccadelli Lorenzo	0001191469

I diversi punti previsti negli sprint sono stati sviluppati in parallelo da tutti i membri del team, collaborando attivamente per ottenere un progetto coerente e integrato, evitando un approccio frammentato.