Ro.Ma.S.

(Robot Management System)

RMS_R_002

(NotificationService – Requisiti)

Storia del documento

Data	Rilascio	Descrizione
2009.02.06	0.0.1	Bozza 1
2009.02.20	0.0.2	Bozza2

Scopo del documento

Stabilire i requisiti che deve avere il servizio NotificationService.

Requisiti

Premessa ai requisiti

Questo servizio astratto è utilizzato per far comunicare ed interagire tra loro i servizi del sistema attraverso l'utilizzo della NOTIFICATION_MAP_TABLE dove vengono correlate tra loro le azioni che compiono i servizi.

NOME CAMPO	DESCRZIONE
ID_SIGNATURE	Identificativo della notifica
SUBSCRIBER	Servizio che riceve la notifica
EVENT	ISTRUZIONE del NOTIFICATOR che fa scattare la notifica
NOTIFICATION	ISTRUZIONE notificata al SUBSCRIBER
NOTIFICATOR	Servizio che invia la notifica
ACTIVE	Indica se la notifica è attiva oppure no

Tabella 1

Ogni servizio non esegue puntualmente un'azione ma ne notifica la richiesta d'esecuzione.

A fronte di una richiesta di esecuzione il servizio ProcessManager esegue l'istruzione od il comando corrispondente all'azione richeista.

Si deduce che un servizio implementa il servizio ActionService per determinare l'istruzione da eseguire ed eredita dal servizio NotificationService per rilevare l'azione da intraprendere.

Il processo prevede che ogni servizio estenda le funzionalità del NotificationService implementando solo quelle che sono le caratteristiche proprietarie.

Con questo sistema di notifiche il SensorService può allertare in qualsiasi istante il ProcessManager di annullare tutti gli spostamenti in AVANTI perché il sensore anteriore ha rilevato un ostacolo.

I servizi devono implementare un meccanismo di feedback per interrogare la tabella sia all'andata che al ritorno del processo attuativo, questo viene realizzato esaminando l'esito dell'azione intrapresa.

Nel diagramma di Figura 1 che segue viene proposto l'interazione dei metodi NotyAction ed ExecuteAction attraverso il diagramma delle Attività.

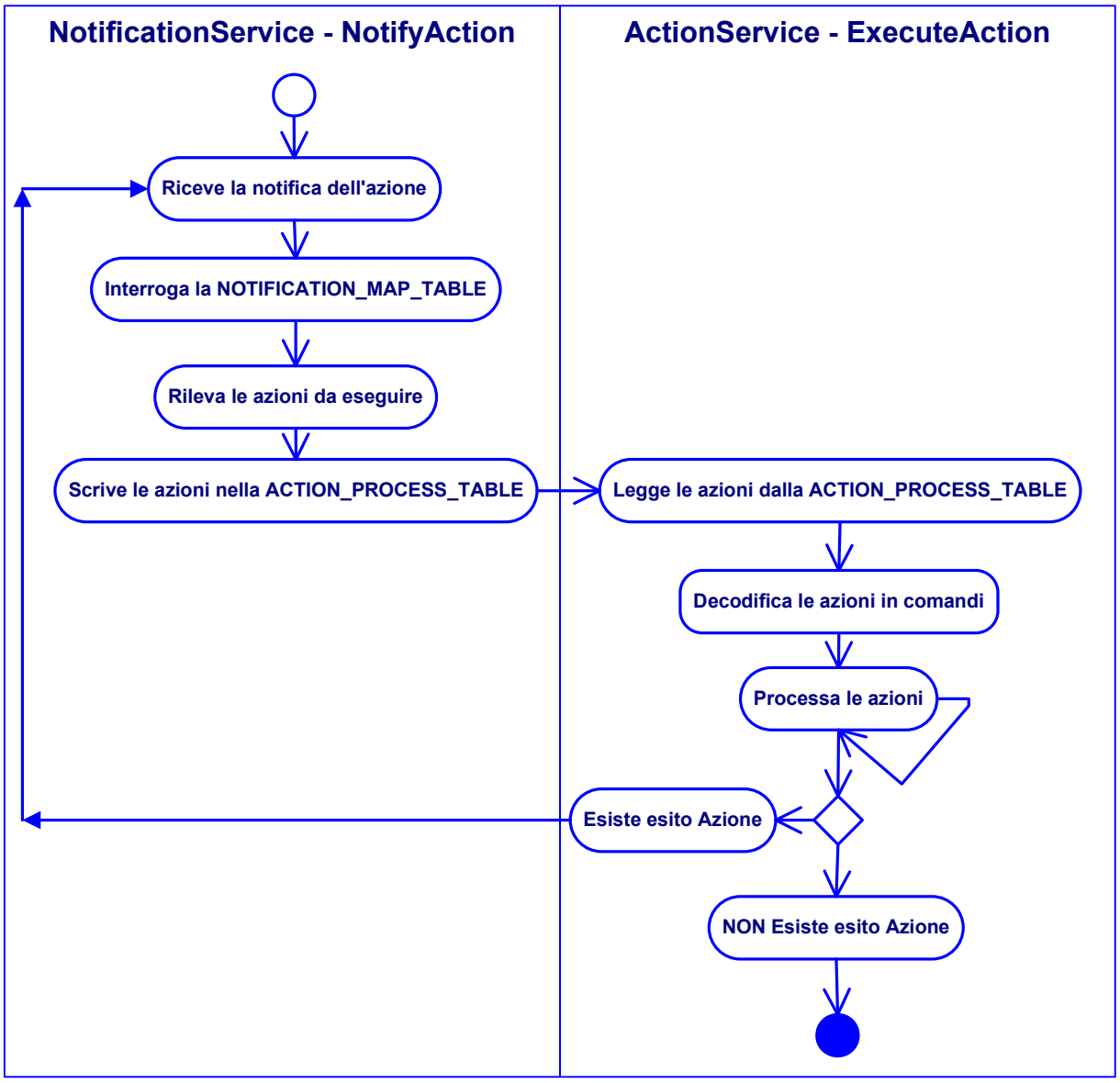

Figura 1

Esempio di file XML di sottoscrizioni:

```
<SpeakRecognitionService_Notifications>
        <Subscription>
                <Subscriber>SPEAK</Subscriber>
                <Event>BUONGIORNO</Event>
                <Notification>REPLY “BUONGIORNO”</Action>
        </Subscription>
</SpeakRecognitionService_Notifications>

<SensorService_Notifications>
        <Subscription>
                <Subscriber>MOVEMENT</Subscriber>
                <Event>SENSOR FRONT ALARM</Event>
                <Notification>FORWARD STOP</Action>
        </Subscription>
</SensorService_Notifications>
```

Esempio 1

Negli esempi precedenti abbiamo simulato la risposta che deve dare il sistema al saluto “BUONGIORNO” ed il fermo motori in senso avanti che deve azionare il sistema alla percezione di un ostacolo frontale.
Attraverso il servizio di notifica e sottoscrizione vengono comandate le azioni da processare dall’ActionService.

Nella Figura 2 di seguito viene illustrato il diagramma di flusso di un servizio che eredita dal NotificationService.

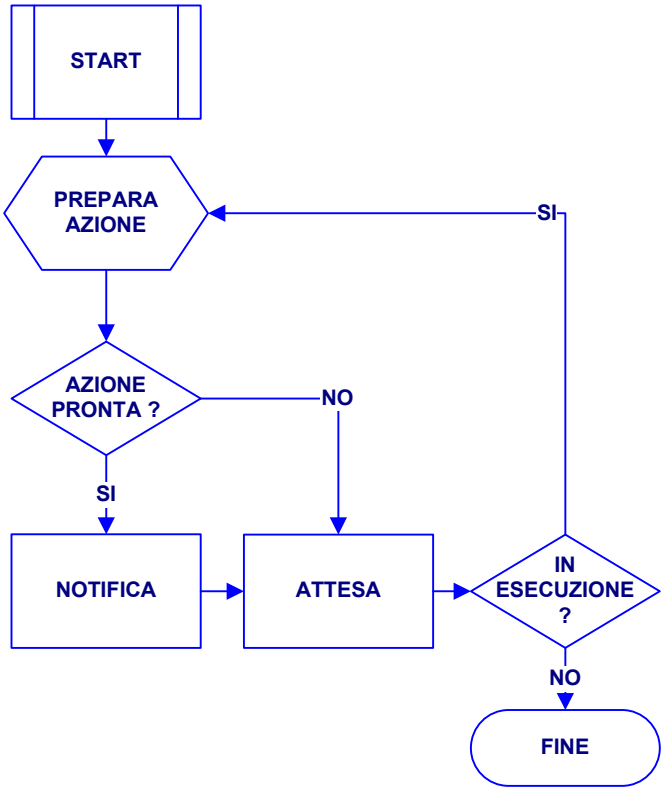

Figura 2

Ipotizziamo la classe che dovrà rappresentare il servizio NotificationService con tre metodi principali riportati nella Tabella 2 che segue.

Metodo	Tipo	Descizione
ExecuteWork	Final	Metodo eseguito dal thread principale
PrepareAction	Abstract	Metodo da implementare soltanto se il servizio deve agire in maniera autonoma. Es. Sensori, SpeakRecognition …
NotifyAction	Final	

Tabella 2

Nel diagramma che segue vengono integrate le funzionalità del NotificationService in un servizio reale evidenziando i metodi della classe.

Nell'esempio di Figura 3 viene schematizzata la struttura del SensorService per un sensore di ostacolo.

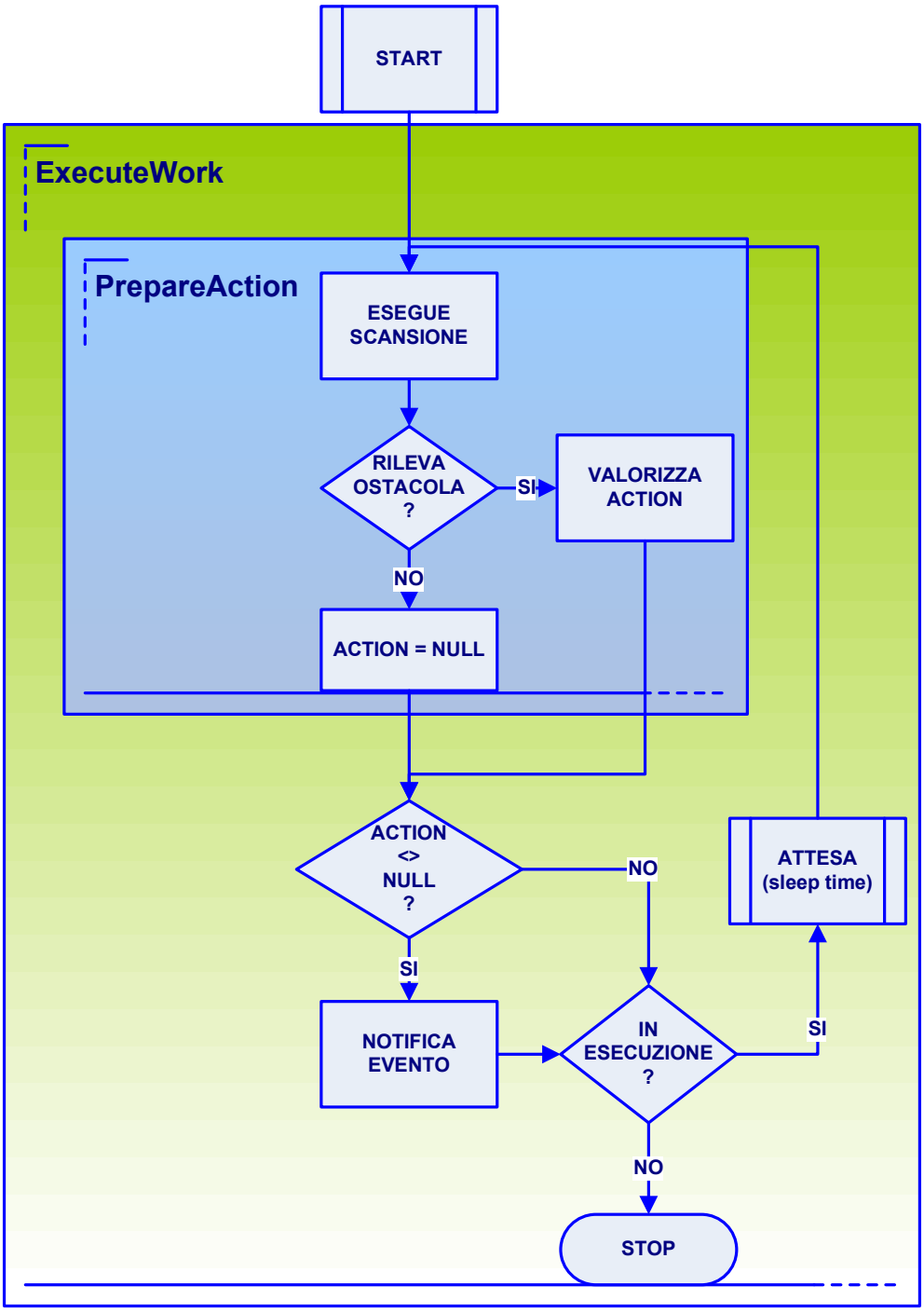

Figura 3

Tabella dei requisiti

Numero	Descrizione
1	Scrive le notifiche all'interno della NOTIFICATION_MAP_TABLE in fase di inizializzazione. Le notifiche sono in formato XML secondo la struttura dell'**Esempio 1**.
2	Espone il metodo astratto PrepareAction che deve essere implementato per preparare l'azione da eseguire. L'azione può essere stata comandata da remoto oppure estrapolata da una interrogazione di un sensore.
3	Implementa l'algoritmo per scrivere la notifica della action attraverso il metodo NotifycationAction.
4	Interroga la NOTIFICATION_MAP_TABLE quando viene richiamato il metodo NotifyAction e scrive nella tabella ACTION_PROCESS_TABLE le azioni da eseguire.
5	Crea l'azione alternativa se non è stato possibile eseguire l'azione richiesta (TYPE = 'ALTERNATIVE') e contrassegnando il campo OWNER con l'identificativo dell'azione ordinaria.
6	Se non è stato possibile eseguire l'azione alternativa esegue un'azione contraria. Le azioni contrarie sono indicate nella ACTION_TABLE come opposte (TYPE = 'OPPOSITE').
7	Cancella le azioni processate anche se non eseguite.
8	Ricrea le azioni impossibili da eseguire come azioni alternative.
9	Ricrea le azioni alternative impossibili da eseguire come azioni opposte.

Tabella 3