POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione

POLO TERRITORIALE DI COMO
Master of Science in Computer Engineering

Ambient Assisted Living: indicatori quantitativi per monitorare la routine di una persona

Relatore: Prof.ssa Sara Comai

Correlatore: Ing. Fabio Veronese

Tesi di Laurea di: Tommaso Motta

Matricola: 816560

Anno Accademico 2014-2015

Sommario

Recenti analisi demografiche mostrano che la vita media della popolazione mondiale si stia allungando. Questo può comportare una progressiva estensione del bacino di utenza dei servizi socio sanitari, con ripercussioni su qualità delle prestazioni erogate e quantità di risorse economiche necessarie. Molti lavori di ricerca sono stati svolti al fine di alleggerire il servizio sanitario dalle crescenti richieste di intervento delle fasce più sensibili. Tali studi si sono indirizzati verso lo sviluppo di sistemi che, attraverso funzionalità tipiche degli Smart Environment, fossero in grado di operare come Ambient Assisted Living (AAL). Questi, impiegando tecnologie non invasive, consentono il monitoraggio delle attività quotidiane degli anziani nell'ambiente domestico e di supportare le loro necessità garantendo un prolungato stile di vita indipendente. Il Politecnico di Milano, tramite l'Assistive Technology Group (ATG), sta sviluppando BRIDGe, un sistema atto a questo compito.

La letteratura suggerisce alcune misure per valutare la routine di una persona che risiede in un AAL. Questo lavoro tesi si prefigge di fornire una risposta al problema sfruttando l'elaborazione in tempo reale dei dati provenienti dai dispositivi (sensori) installati nell'abitazione della persona assistita. Impiegando il linguaggio EPL ed Esper sono sviluppati indicatori quantitativi che, da flussi di dati reali (ARAS) o simulati (SHARON), costruiscono cinque stime utili per valutare lo stato di salute e di benessere di un soggetto. Sono individuati aspetti immediati come (a) le ore giornaliere di attività rilevate da un singolo sensore, e (b) i consumi giornalieri di servizi domestici. Successivamente aspetti più astratti come (c) il riconoscimento di un'attività complessa, (d) la valutazione dello stile di vita valutando entropia oraria e giornaliera relativa ad un singolo sensore nonché (e) quella relativa all'intera casa.

Così si estraggono tramite indicatori quantitativi informazioni riguardo diversi aspetti delle abitudini casalinghe di una persona che, applicate su dataset di test, si dimostrano adeguate per descrivere le condizioni di vita ordinarie ed evidenziare eventuali anomalie.

Abstract

A global process of population ageing emerges from the latest demographic studies. Among the consequences of this phenomenon there is the growth of health care services users, with effects on both the quality of the delivered services and the financial costs to be sustained. During last decade many studies aimed to improve the efficiency of services for fragile people (e.g. disabled or elderly). Those studies aimed at developing Smart Environments providing Ambient Assisted Living (AAL) functionalities. Such systems leverage unobtrusive technologies, enabling the monitoring of domestic daily activities of fragile people and support their needs, guaranteeing them an independent life and reducing the need of intervention.

This topic, thanks to Assistive Technology Group (ATG), is addressed by Politecnico di Milano with the research project named BRIDGe (Behaviour dRift compensation for autonomous and InDependent livinG).

Few projects in the literature suggest measures useful to evaluate the routine of a person trough devices adopted in AAL systems. This thesis aims at providing a solution, live processing of data incoming from sensors installed in an elderly person's house. Leveraging Esper and Event Processing Language (EPL) it was possible to develop quantitative indicators based on simulated as well as on real data streams. The extracted indices are five and they concern various information about routine of a person: from basic aspects like (a) daily hours of activity measured by a single sensor, and (b) daily consumption of domestic services (water, electricity, ...) to higher level aspects, e.g. (c) identifying the occurrences of a complex activity, (d) evaluating the activeness of a single sensor and (e) the lifestyle of the monitored person considering all the sensors.

This enables the extraction of information about different aspects of domestic habits and routines of elderly people, and applied to test datasets these indicators show to be effecting in describing the ordinary life conditions while enhancing behavior variations.

Ringraziamenti

Desidero innanzitutto ringraziare la Professoressa Sara Comai per avermi dato la possibilità di lavorare e contribuire, nel mio piccolo, ad un progetto che, per motivi famigliari, sento molto vicino. Apprezzerò sempre il suo grande interesse per lo svolgimento del lavoro, il suo contributo e la sua disponibilità nel leggere e correggere questa tesi.

Un ringraziamento sentito al mio correlatore Fabio Veronese il quale mi ha sempre supportato e, con infinita pazienza, sopportato in questi lunghi mesi di duro lavoro riuscendo sempre, con grande disponibilità, a risolvere tutti i problemi in cui riuscivo ad infilarmi.

Un ringraziamento speciale ai miei genitori perché, se sono arrivato fin qui, è grazie a voi che avete sempre creduto in me, mi siete sempre stati vicino e che mi avete permesso di diventare la persona che sono e di cui, spero, voi possiate essere orgogliosi.

Un grazie anche a mia sorella Camilla, compagna di avventure e disavventure letteralmente da una vita, e al resto della famiglia, vicina e lontana, ma che è ugualmente sempre con me.

Un ringraziamento a tutti i miei amici, vecchi e nuovi, il cui sostegno e la vicinanza sono stati fondamentali tanto nella vita quanto nello studio e sono stati sempre la migliore e più affidabile fonte di divertimento, ma anche valvola di sfogo per i momenti difficili.

Un pensiero, infine, anche agli amici che ho trovato in questo viaggio universitario, qualcuno l'ha vissuto interamente e qualcuno solo in parte, ma senza ciascuno di voi la mia esperienza al Politecnico, prima a Milano e poi a Como, sarebbe stata completamente diversa, e certamente più noiosa.

Indice

Indice delle figure

Indice delle tabelle

Capitolo 1

Introduzione

Nel secondo dopoguerra, al termine della stagione dei grandi conflitti, le nazioni dell'ovest hanno conosciuto incredibili slanci economici, che hanno portato a migliorare di gran lunga le condizioni economico sociali delle persone, e quindi anche la qualità della vita rispetto alla situazione precedente il conflitto bellico.

Per decenni si sono registrati continui incrementi del numero delle nascite e, grazie al ritrovato benessere ed ai continui miglioramenti della scienza e della tecnologia, si è manifestato un progressivo allungamento della vita media. Nel corso di questi ultimi anni, a seguito di evoluzioni delle dinamiche economico - sociali, se la crescita demografica è andata sempre più contenendosi, l'innalzamento della soglia di aspettativa di vita non si è mai arrestato o rallentato. Ad oggi infatti i dati relativi alla natalità parlano di una lenta progressiva riduzione delle nascite, come mostrato nella Figura 1.1 [**46**].

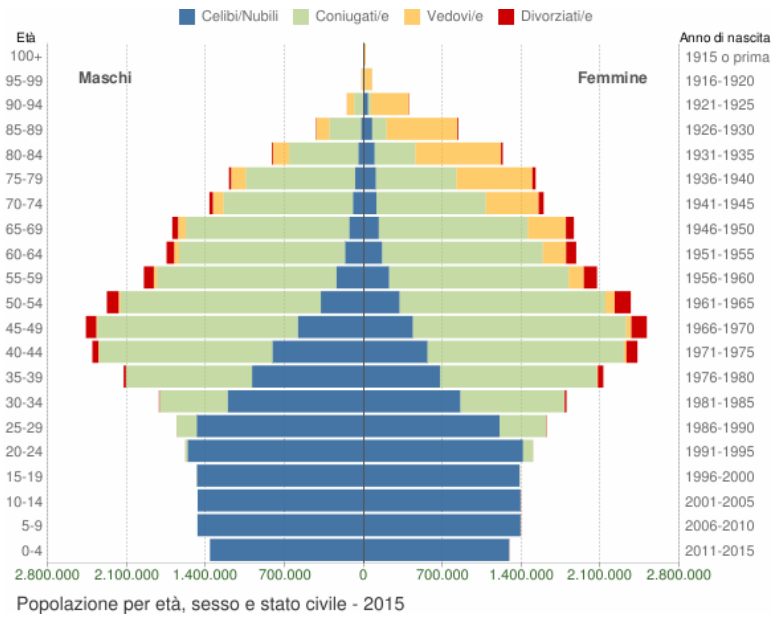

Figura 1.1: Piramide della popolazione italiana, ISTAT, 2015.

Il grafico in figura mostra chiaramente la situazione demografica italiana come specchio della descrizione precedente. In questa rappresentazione si comprende nitidamente come negli anni a venire i servizi in generale, tra cui anche quelli sanitari e quelli alla persona, saranno messi alla prova da un'utenza di gran lunga maggiore rispetto all'attuale sia a causa dell'allungamento dell'aspettativa di vita, sia per l'avanzamento verso età "fragili" di fasce numericamente più corpose a livello demografico. Un'analisi censitaria statunitense [**70**] stima una crescita pari al 101% fra gli anni 2000 e i 2030 delle persone over 65, con una media annuale del 2.3%, che porterebbe ad attestarsi ad oltre 25% della popolazione totale [**44**]. Uno studio svolto dalle Nazioni Unite [**64**] su scala globale afferma come entro il 2047 il numero di persone di età maggiore di 60 anni supererà quello dei nascituri, ed in particolare entro il 2030 la popolazione mondiale di centenari supererà il milione (dai 180'000 dell'anno 2000). Entrambi gli studi si soffermano a sottolineare come questa rapida crescita nelle fasce più avanzate di popolazione non sia seguita in egual misura da una crescita nelle possibilità di supporto medico o sanitario oppure di vigilanza da parte delle famiglie stesse. Negli ultimi vent'anni la consapevolezza di questo problema e il progresso tecnologico hanno permesso di approcciare degli studi che si muovessero nella direzione di *Smart Home* (domotica) e sistemi *Ambient Assisted Living* (AAL – Ambiente di vita assistito) con lo scopo di migliorare l'esperienza di vita nel proprio ambiente domestico garantendo prolungata indipendenza sociale. La sfida attuale quindi è quella di permettere di tenere traccia, tramite l'ausilio di dispositivi installati nelle abitazioni o indossabili, dello stato di salute di persone che vivono da sole in modo da poter riconoscere per tempo l'insorgenza di patologie, permettere assistenza domestica e rassicurare i familiari. All'interno di questa ricerca si inserisce il progetto BRIDGe (Behaviour dRift compensation for autonomous and inDependent livinG) [**58**], un sistema Ambient Assisted Living (AAL), sviluppato in collaborazione dall'Assistive Technology Group (ATG) [**8**] del Politecnico di Milano per l'aspetto tecnologico e CRAiS (Centro Risorse per le Autonomie e l'Inclusione Sociale [**27**]) per l'aspetto sociale.

1.1 Progetto BRIDGe

Behaviour dRift compensation for autonomous and inDependent livinG (BRIDGe) è un ambizioso progetto in corso di realizzazione da parte dell'ATG del Politecnico di Milano. Si occupa di sviluppare un sistema AAL che metta in comunicazione, attraverso una struttura di monitoraggio basata su tecnologie di sensori wireless, una persona che vive a casa da sola (detta IL, Independent Living) con l'ambiente sociale (famiglia, associazioni o figure professionali che se ne prendono cura). Il target di utenti al quale il sistema è indirizzato sono anziani, persone con deficit cognitivi o fisici e quelle affette da patologie che possono compromettere salute e stile di vita autonomo, ma che, pur necessitando di supervisione, vogliano mantenere la propria indipendenza. BRIDGe quindi ha proposto l'installazione nell'ambiente abitativo dell'utente un sistema basato su sensori wireless non invasivi ed economici che permettono il continuo controllo da remoto verificando quindi il corretto procedere quotidiano delle attività della persona, di valutarne le condizioni di salute e di intervenire ove venissero riscontrati comportamenti anomali oppure scostamenti sensibili dalla routine.

1.1.1 Architettura BRIDGe

Poiché il lavoro svolto è inserito all'interno di un progetto ampio, è opportuno prendere visione dell'infrastruttura complessiva che è stata sviluppata per il progetto BRIDGe e dove sia posizionato l'intervento del lavoro che verrà proposto. A questo scopo ATG ha fatto uso di due sottosistemi, uno *remoto* ed uno *locale*. La Figura 1.2 rappresenta in maniera semplice la struttura appena introdotta. A sinistra vi è la componente in *locale* costituita da apparecchi a basso livello di computazione ed energia tra cui dispositivi per l'Home Automation, sensori ed un piccolo server locale (in particolare questo progetto impiega un Raspberry PI). Nello specifico, questa parte di architettura ha lo scopo di connettere fra loro le componenti autonome della Smart Home per generare una serie di servizi di monitoraggio e controllo finalizzati a procedure di analisi a breve termine che definiscano funzionalità della Smart Home e migliorino la qualità di vita dell'utente. La componente in *remoto* (la parte sinistra nella Figura 1.2) invece è molto più potente a livello di risorse hardware. Essa ha infatti il compito di raccogliere informazioni provenienti dalla parte *locale* e svolgere analisi complesse su periodi di tempo maggiori

nonché a lungo termine: il maggior numero di informazioni disponibili sulla struttura *remota* consente la realizzazione di studi più ampi e maggiormente focalizzati sulle attività di routine e sulle abitudini dell'utente permettendo quindi la valutazione di aspetti relativi alla sua salute. Per ottenere queste funzionalità si rende necessaria l'installazione di un database, di un core (web server e intelligenza artificiale) per l'elaborazione e lo studio dei dati raccolti ed infine un'interfaccia per il controllo remoto e l'interazione da parte di persone esterne (familiari, assistenti, …).

Lo schema seguente mostra le componenti principali dell'architettura di BRIDGe.

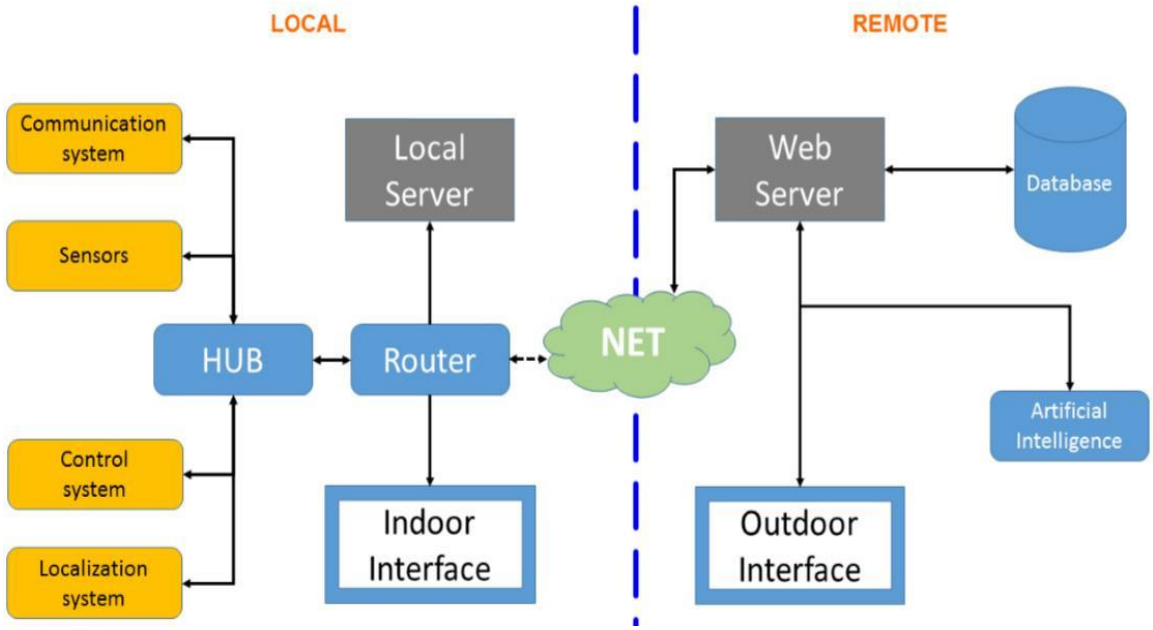

Figura 1.2: Architettura del progetto BRIDGe.

Il lavoro della tesi presente è incardinato nella componente del sottosistema *remoto*. L'oggetto sviluppato dalla tesi, infatti, è uno strumento della componente di Intelligenza Artificiale (IA): le attività rilevate dai sensori in *locale* sono inviati al sistema *remoto* dove sono analizzati in tempo reale per ricavare informazioni utili per valutare lo stile di vita della persona monitorata. A differenza della semplice domotica, in cui le informazioni sono estratte da analisi a breve termine che hanno lo scopo di migliorare l'esperienza casalinga della persona che vi risiede, in questo caso l'analisi può essere effettuata a medio e lungo termine, ed i risultati vengono utilizzati per garantire, a chi se ne prende cura, riguardo la salute del monitorato oppure avvisare l'insorgenza di criticità.

1.2 Formulazione del problema

La bibliografia è ricca di ricerche riguardo modelli volti a identificare e riconoscere le attività svolte da persone fragili all'interno di strutture abitative monitorate da sistemi AAL. Molti infatti sono gli studi reperiti che hanno portato alla definizione di modelli riguardo il comportamento degli utenti [**10**]. I loro risultati sono utili per effettuare le analisi di assistenza socio-sanitaria: lo studio in tempo reale del comportamento e delle sue variazioni può essere strumento valido per valutare lo stato di salute e benessere (nel breve e nel lungo periodo) della persona monitorata. Anche in questo caso la bibliografia è stata di supporto fornendo un'ampia collezione di informazioni che possono essere impiegate per questo genere di valutazioni.

La richiesta quindi è stata quella di svolgere un processo selettivo volto a scegliere le informazioni realmente utili da estrarre dai dati provenienti dai sensori installati in un ambiente intelligente sotto forma di indicatore quantitativo. La necessità successiva è stata quella di trovare le modalità implementative per questo scopo, metterle in pratica in un apposito ambiente di sviluppo ed infine verificare la validità e robustezza dei risultati ottenuti.

1.3 Contributo della tesi

Le sfide affrontate nello svolgimenti della presente tesi sono principalmente tre, corrispondenti ad altrettanti aspetti valutati.

In primo luogo è stata necessaria un'attenta valutazione riguardo l'attendibilità e la valenza degli indicatori proposti dalla bibliografia per poi procedere con la selezione di quelli che meglio potevano prestarsi come strumento di analisi a lungo o a breve periodo della condizione della persona monitorata. Per fare questo sono stati ricercati vari livelli di informazioni riguardo la routine della persona monitorata: misurare il tempo dedicato a svolgere attività semplici e complesse, stimare i consumi di servizi (luce, acqua o gas) e valutare l'imprevedibilità dello stile di vita.

La seconda sfida affrontata è stata quella relativa alla ricerca ed adozione di un dataset per poter simulare il monitoraggio in tempo reale di una struttura abitativa che (a) rispondesse in maniera completa alle tecnologie e alla struttura del caso di studio, (b) si avvicinasse maggiormente alla tipologia di utente di destinazione, ed infine (c) che

fornisse dati adeguati per valutare la bontà delle informazioni estratte. Fra i vari dataset rinvenuti pubblicamente da studi precedenti la preferenza è andata su Activity Recognition with Ambient Sensing (ARAS [**80**]), i cui dati presentano alcune peculiarità molto interessanti come la elevata copertura temporale, la tipologia di sensori adottati ed il numero di attività rilevate benché nell'abitazione in considerazione vi risiedessero due persone e non una sola come nel caso ideale. In fase di verifica e validazione dei risultati estratti dagli indicatori, il dataset usato è ottenuto da SHARON (Simulator of human Activities, Routines and needs: Analysis, Implementation and Validation [**68**]), un simulatore di attività umana sviluppato internamente al progetto BRIDGe.

La terza sfida è stata la strumentazione software da impiegare per lo sviluppo degli indicatori introdotti: quindi le modalità per simulare il flusso di dati in ingresso al sistema e l'elaborazione in tempo reale di questi (detta *Event Processing*). Allo scopo è stato adottato Esper [**38**] come soluzione software perché, integrabile in Java e con una sintassi simile a quella di SQL, permetta di riconoscere evenienze particolari nel flusso in ingresso.

Il lavoro oggetto di questa tesi permette di arricchire il materiale reperibile con un'unica soluzione che accorpi la ricerca teorica degli indicatori per monitorare la vita di una persona alla sua implementazione pratica applicandola su dati che ne validino i risultati.

1.4 Organizzazione della tesi

Dopo la parentesi introduttiva, la tesi nei prossimi capitoli sarà organizzata come segue.

Il Capitolo 2 illustra il cosiddetto Stato dell'Arte, un breve excursus volto ad esplicare i concetti di Smart Home e a descrivere quanto studiato finora in termini di dispositivi hardware (sensori) disponibili e riguardo i dataset reperibili. Infine verranno discusse le informazioni (estraibili dai dati) proposte dalla letteratura e ritenute interessanti per il progetto BRIDGe.

Il Capitolo 3 presenta in maniera approfondita il concetto di *Event Processing* e le modalità con cui svolgere l'analisi del flusso di dati in ingresso al sistema. Viene poi presentato Esper, la soluzione software scelta, con accenni riguardo la sua sintassi EPL e la sua architettura. Successivamente viene introdotto il Generatore di Eventi, una routine

Java che simula la ricezione di eventi nel sistema leggendo il dataset adottato. Infine viene illustrato brevemente SHARON, un sistema impiegato per generare un dataset sintetico per la validazione dei risultati ottenuti.

Il Capitolo 4 introduce la struttura dei dati adottati per la simulazione e successivamente presenta i primi due indicatori (ITASS e ISCS), quelli più semplici, nonché quelli che descrivono aspetti concreti della vita della persona monitorata. A questo scopo sono introdotte interrogazioni EPL e diagrammi di flusso per migliorare la comprensione.

Il Capitolo 5 introduce un secondo insieme di indicatori (ISVUS, IRAC e ISVIS), quelli più complessi ed articolati perché volti ad estrarre informazioni astratte.

Il Capitolo 6 verifica e valida i risultati estratti impiegando i dati simulati ottenuti con SHARON. A questo scopo si ricorre a grafici ed alla valutazione di precisione e recupero.

Il Capitolo 7 infine conclude la trattazione illustrando possibili lavori futuri o migliorie apportabili ai risultati ottenuti e agli strumenti impiegati.

Capitolo 2

Stato dell'arte

Uno degli aspetti fondamentali per procedere con l'illustrazione del lavoro sviluppato, consiste nell'effettuare un'analisi dettagliata della bibliografia di pubblicazioni, studi e tesi disponibili sul web riguardante il monitoraggio di un ambiente domestico. Benché il target del caso di studio sia quello riferito esplicitamente a persone anziane con stile di vita indipendente che vivono da sole, questo non ha portato ad escludere articoli che non trattassero propriamente questo argomento. Al fine preposto, il procedimento seguito per muoversi all'interno del materiale raccolto ha avuto l'iniziale necessità di chiarire e classificare i concetti fondamentali riguardo Smart Home (Sezione 2.1), le tipologie di sensori (Sezione 2.2) ed i dataset sviluppati in altre ricerche (Sezione 2.3). Successivamente (Sezione 2.4), utilizzando le esperienze e le ricerche testimoniate dai documenti letti e discussi, il lavoro si è indirizzato alla definizione delle informazioni più interessanti e significative da estrarre dai sensori tramite una delle soluzione software offerte dal mercato: Esper.

2.1 Smart Home

Per definire il concetto di Casa Intelligente (Smart Home) dobbiamo partire da quello di Ambienti Intelligenti (Smart o Intelligent Environment). Possiamo identificare questi ambienti come una perfetta integrazione e sinergie tra tecnologia (hardware), strutture (ambienti) e le quotidiane attività umane. In particolare esso sarà definito come un sistema in grado di acquisire ed applicare conoscenze riguardo un ambiente ed il suo abitante con lo scopo di migliorare l'esperienza ed il controllo del soggetto in quell'ambiente [**47**], [**26**]. Secondo Das et al. [**32**], [**33**], e Baeg et al. [**11**] uno Smart Environment è come un piccolo mondo in cui debbano sussistere alcune caratteristiche:

un livello di autonomia tale da ridurre l'interazione manuale umana, una capacità di adattamento al contesto e alle preferenze dell'utente ed infine la capacità di comunicare con gli umani in maniera naturale e quindi facilmente comprensibile. Le tipologie di Intelligent Environment che possono essere sviluppate sono diverse e fra queste troviamo Case Intelligenti [**65**], [**53**], [**62**], [**87**], Uffici Intelligenti [**52**], [**84**], [**49**], [**22**], Classi Intelligenti [**40**], [**42**] e Ospedali Intelligenti [**13**].

In questo modo risulta molto più chiaro quale sia il principio alla base di una Smart Home: un ambiente di tipo domestico in cui siano installate apparecchiature volte a sviluppare conoscenze per migliorare l'esperienza ed il controllo degli abitanti sulla base delle loro attività quotidiane e delle loro abitudini. Fra le capacità che una Casa Intelligente offre all'utente, oltre alle ordinarie singole attività domestiche (relative agli elettrodomestici e al loro funzionamento), vi sono quelle deputate ad integrare le apparecchiature domestiche con sistemi autonomi che interagiscano con gli abitanti migliorandone il comfort ed aiutandolo in evenienze particolari. Negli ultimi anni le ricerche hanno aperto una nuova direzione di ricerca, quella di sfruttare le strutture e le potenzialità degli ambienti intelligenti per destinarli ad un altro livello di assistenza, alla rilevazione di comportamenti anomali del soggetto monitorato, e quindi di trarre conclusioni riguardo il suo stato di salute. I dispositivi installati saranno dei sensori volti a monitorare l'ambiente e le attività dell'abitante in modo tale che quest'ultimo possa riceverne dei benefici. I benefici ovviamente dipendono da parametri come la tipologia del soggetto che viene monitorato, quindi in base all'età, gli interessi e i suoi bisogni. Tali benefici saranno il risultato dell'elaborazione in tempo reale dell'istante delle rilevazioni fornite dai sensori connessi in una rete che, attraverso un sistema di intelligenza autonoma, esegua in autonomia determinate azioni funzioni qualora vengano riscontrate condizioni specifiche. Esempi di queste funzioni possono essere degli avvisi di salute precaria o incerta, oppure di comportamenti che si allontanano da quelli abituali.

La tesi descriverà un sistema destinato al monitoraggio di una persona anziana sola a casa alla quale si possa garantire un discreto livello di autonomia, controllando al contempo il suo positivo stato di salute. I sensori impiegati sono introdotti nel paragrafo successivo.

2.2 Sensori per la raccolta di informazioni in una Smart Home

Come riportato da in numerose ricerche, tra cui [**10**], [**35**], [**2**], [**77**], uno dei requisiti per poter ottenere una consistente e varia quantità di dati per valutare lo stato di salute di un anziano monitorato che vive da solo è quello di utilizzare un'eterogeneità di sensori appropriati per il caso di studio.

Le ricerche riguardo le Case Intelligenti hanno mostrato come il mercato in questo senso offra un'ampia gamma di dispositivi in grado di restituire molteplici informazioni e risultati. La classificazione dei sensori presa in considerazione da Atallah et al. [**10**] li distingue in due aree di interesse:

a) *sensori ambientali*, installati nelle abitazioni per monitorare le attività domestiche degli inquilini.
b) *sensori indossabili*, in grado di monitorare gli aspetti fisici, biologici ed in alcuni casi pure psicologici delle attività svolte.

Tra i sensori ambientali possiamo citare:

- *Telecamere e microfoni*: sono fra i sensori più diffusi; si occupano di monitorare la vita dell'abitante e le sue attività catturando immagini e audio dalla casa.
- *Sensori di consumi (elettricità, acqua e gas)*: si occupano di tenere traccia della quantità consumate per le singole grandezze fisiche.
- *Sensori di pressione*: posizionati sotto sedie, divani o letti, si occupano di rilevare la posizione dell'abitante sul singolo elemento di arredamento quando, utilizzandolo, esercita una pressione su di essi.
- *PIR (Passive Infrared – sensore a infrarosso passivo)*: sono sensori che rilevano l'attività domestica in maniera alternativa alla telecamera. Questi infatti catturano i movimenti in un determinato ambiente i movimenti senza però utilizzare una camera che riprenda la scena. Sono spesso utilizzati per rilevare il cambiamento di luce al passaggio di un corpo.
- *Sensori Ambientali*: sono sensori che hanno lo scopo di rilevare caratteristiche come umidità, temperatura, fumo, (ecc…) per le stanze in cui sono installati.
- *Sensori di Distanza:* sono sensori volti a rilevare la presenza dell'abitante quando questo si avvicina ad essi. Le distanze di riferimento entro le quali sono rilevate attività dipendono dalle caratteristiche dei vari modelli.

- *Sensori di Contatto:* hanno la specificità di rilevare i movimenti di apertura e chiusura come per esempio antine o porte. Sono costituiti da due corpi separati ed installati sulle due antine o fra una porta e l'infisso tali che quando le due componenti entrano in contatto, il sensore restituisce un segnale di avvenuta chiusura.
- *Ricevitori infrarossi:* si occupano di rilevare quando due componenti entrano in comunicazione tra loro attraverso un raggio infrarosso; questo sensore è alla base del principio di funzionamento dell'interazione fra televisore e telecomando.

Per quanto riguarda invece i sensori che, indossati dal soggetto, ne monitorano l'attività, possiamo definirli appartenenti alla cosiddetta *Body Sensor Network* (BSN – *Rete di sensori del corpo*). Essi appartengono ad un'area di ricerca volta a risolvere sfide inerenti lo sviluppo di componenti hardware, connessioni sicure wireless e comunicazioni per il monitoraggio del corpo umano e delle sue attività. Fra i sensori sviluppati più diffusi possiamo prendere in considerazione quelli analizzati da Lowe et al. [**55**]:

- *Accelerometri:* sono sensori che, a seconda della configurazione, possono essere indossati in varie parti del corpo (alla vita, al polso, sul braccio, sulla gamba, all'orecchio, alla spalla…) [**9**]. Essi hanno il compito di misurare e/o rilevare la variazione di velocità, un'informazione utile per il riconoscimento delle attività eseguite dalla persona e dei suoi stati di salute.
- Rilevatori della *Temperatura corporea, Flusso respiratorio, ECG, Pulsiossimetro:* strumenti prevalentemente clinici volti a restituire informazioni riguardo questi parametri vitali.
- *Telefoni Cellulare (Smartphones):* numerose ricerche hanno utilizzato questi moderni strumenti di comunicazione sempre più potenti e inseparabili dall'uomo, per poterne sfruttare le funzionalità (camera, microfono, GPS, accelerometro, Bluetooth) per monitorare la persona che lo porta sempre con sé nelle attività quotidiane e nelle sue interazioni sia con le persone che con il mondo che lo circonda [**66**], [**36**], [**37**].
- *Giroscopio:* questo sensore misura la velocità angolare (movimento di rotazione) del soggetto che lo indossa è generalmente costituito da un elemento vibrante. Il calcolo di tale grandezza fa uso dell'effetto di Coriolis, data dalla formula

$F_c = -2m(\omega \times v)$, in cui ω è la velocità angolare, m la massa dell'oggetto vibrante e v la velocità lineare del movimento. La ricerca di questa informazione spesso è volta ad integrare quella dell'accelerometro per produrre un risultato più accurato di quello offerto da quest'ultimo sensore. Il giroscopio è presente in diversi modelli indossabili.

- *Magnetometro:* questo sensore indossabile ha il compito di rilevare attraverso l'impiego di campi magnetici interni alla stanza il corretto posizionamento della persona monitorata. Anche in questo caso, come in quello del giroscopio, le informazioni estratte andranno ad arricchire ulteriormente quelle restituite dal più semplice accelerometro. Infatti, mentre quest'ultimo sensore è in grado di restituire l'informazione riguardo l'attività "stare seduto", un magnetometro può restituire l'informazione che il soggetto sia seduto in direzione del televisore, e quindi probabilmente che lo stia guardando. Per determinare l'informazione si impiega il calcolo della forza di Lorenz riguardante le direzioni di un oggetto inserito in un campo magnetico.

La ricerca non si è fermata a questa distinzione; gli studiosi sono andati oltre e hanno svolto degli studi che permettessero di adottare contemporaneamente entrambe le tipologie di sensori e di ottenere dei risultati costruiti con ambo i contributi.

2.2.1 I sensori in BRIDGe

In termini di sensori da adottare per la realizzazione degli indicatori, la tesi ha seguito le decisioni prese dai creatori del progetto BRIDGe [**58**]. I due aspetti che gli autori del progetto hanno impiegato per la scelta sono risultati condivisi da molti altri autori e ricercatori in letteratura; questi due parametri sono:

1. il *costo* complessivamente contenuto (acquisto, installazione, manutenzione, ecc), affinché il prodotto possa essere accessibile al più ampio bacino di utilizzatori possibile.
2. la non *invasività*, cioè l'assenza di una qualsiasi forma di disagio, disturbo o di violazione della privacy dell'abitante nelle sue attività.

Pertanto sono stati esclusi sensori non rispondenti a queste due caratteristiche come telecamere e microfoni (benché estremamente diffusi) perché rappresentano un'eccessiva

intrusione nella vita dell'utente. E' stata scartata allo stesso modo l'intera famiglia di sensori da indossare perché troppo invasiva nel normale svolgimento delle attività del soggetto. La strada intrapresa è quella costituita da sensori da installare ed (o) incorporare nell'ambiente domestico. In particolare la scelta è ricaduta su una serie di sensori ambientali che riportassero solo due messaggi possibili '1' oppure '0' a seconda che questi sensori rilevassero o meno un'attività. In letteratura questi sensori vengono detti *binari* o di *cambiamento di* stato. Nella sezione seguente, riguardo al *dataset* impiegato, sarà più chiaro quali sensori verranno adottati all'interno di questo progetto.

2.3 Dataset di vita reale

Un elemento fondamentale per procedere con la corretta modellazione delle istruzioni volte ad estrarre informazioni dal flusso di dati provenienti dai sensori è l'*attività di raccolta dei dati*. Tale sfida in letteratura è stata affrontata con varie modalità. Sono diversi infatti gli studi che si sono occupati di collezionare, analizzare e pubblicare questi dati in forma di *dataset*.

La ricerca bibliografica che ho eseguito ha permesso di individuare cinque progetti che hanno sviluppato dei dataset offerti pubblicamente e potenzialmente adottabili dal mio progetto. Questi dataset prendono il nome del progetto all'interno del quale vengono sviluppati, essi sono ARAS (Sezione 2.3.1), CASAS (Sezione 2.3.2), Kasteren (Sezione 2.3.3), MIT (Sezione 2.3.4) e Domus (Sezione 2.3.5). A questi cinque si aggiunge un dataset costruito all'interno di BRIDGe tramite un sistema si simulazione autonoma di attività di una persona all'interno di un'abitazione; questo modello prende il nome di SHARON e verrà introdotto nel Capitolo 3. Nella tabella seguente sono riportati i dataset che verranno illustrati ed alcune delle loro caratteristiche principali.

2.3.1 Progetto ARAS

ARAS (Activity Recognition with Ambient Sensing – *Riconoscimento di Attività con Sensori Ambientali*) è un progetto di ricerca pubblicato da Tunca et al. [**4**] e descrive il monitoraggio di due ambienti abitativi (*HouseA* e *HouseB*) in cui vivono due persone. Il dataset relativo copre le cosiddette ADL (Activity of Daily Living – *Attività di Vita*

Giornaliera), che descrivono le attività svolte dagli abitanti della casa attraverso messaggi provenienti dai sensori. La copertura del dataset è di trenta giorni per ciascuna casa, per cui attraverso venti sensori binari sono riconosciute 27 attività diverse (dormire, preparare la colazione, fare colazione, guardare la tv, leggere, ...). Tali sensori sono inseriti in un protocollo di comunicazione ZigBee e restituiscono ogni istante un valore di stato '1' o '0' a seconda che sia rilevata o meno un'attività. Oltre ai messaggi di cambiamento di stato dei sensori sono restituite le annotazioni delle attività svolte da ambo gli inquilini, riportate manualmente tramite appositi strumenti collocati in vari punti della casa, in modo tale da non interrompere l'attività che viene svolta. Queste annotazioni sono la parte più ambigua e difficoltosa per il processo di raccolta di dati dai sensori perché soggetti a possibili errori o approssimazioni da parte dell'utente. Un esempio di approssimazione è rappresentato dalla rilevazione da parte dell'abitante della sola attività prevalente rispetto ad un determinato evento; questo comporta la perdita di informazioni riguardanti attività svolte in contemporanea (esempio: pranzare e guardare la televisione allo stesso tempo: l'utente annota solo pranzare). Il dataset prodotto è liberamente accessibile dal sito del progetto ARAS [**15**]. Di seguito vengono riportate le mappe delle strutture abitative messe a disposizione dal progetto, in particolare la Figura 2.1 illustra la soluzione abitativa denominata *HouseA* ed una legenda riguardo i sensori impiegati. In particolare in rosso sono riportati i sensori ove sono stati installati nella mappa ed il loro identificativo numerico. La Figura 2.2 illustra gli stessi aspetti ma per la seconda abitazione chiamata *HouseB*.

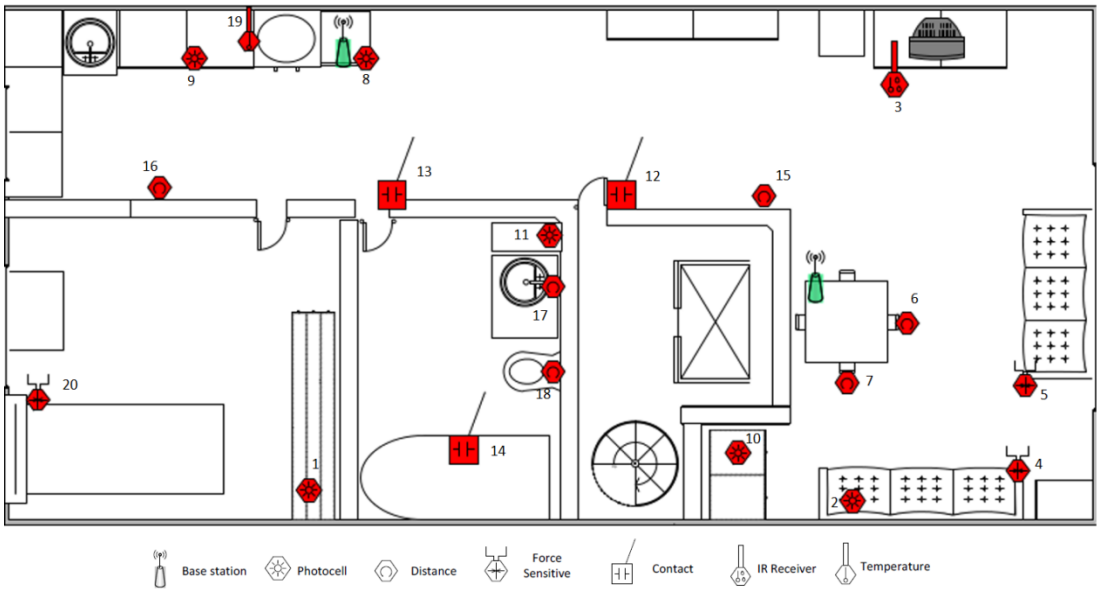

Figura 2.1: Mappa HouseA e legenda sensori per il progetto ARAS.

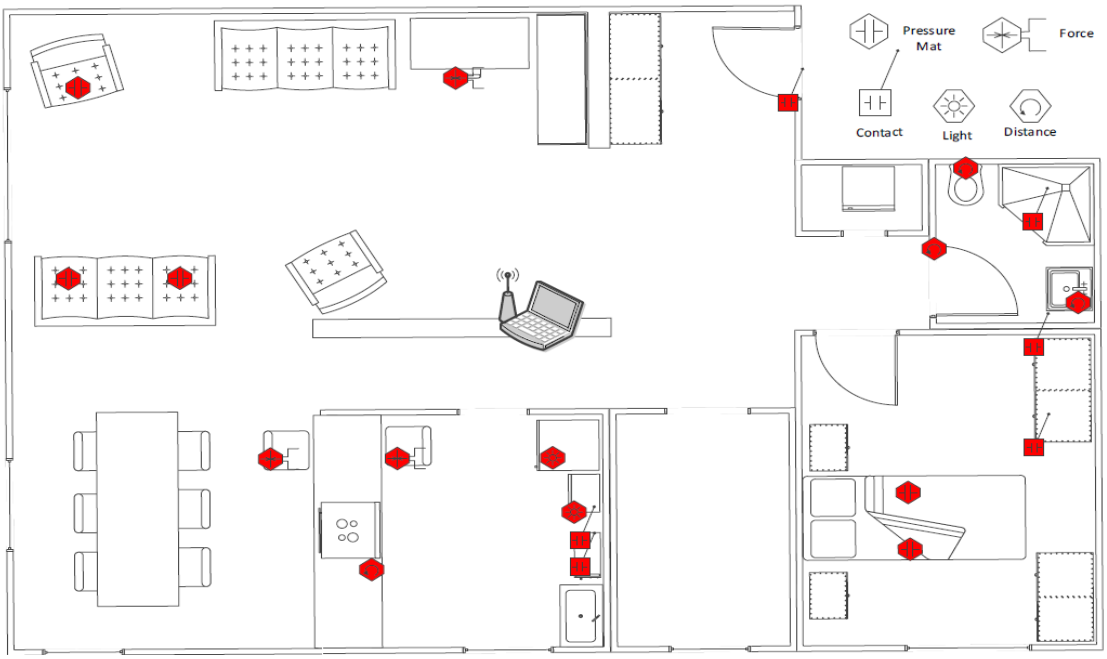

Figura 2.2: Mappa HouseB e legenda sensori per il progetto ARAS.

2.3.2 Progetto CASAS

CASAS (Center for Advanced Studies in Adaptive Systems – *Centro per Studi Avanzati in Sistemi Adattivi*) è un progetto di Diane J. Cook [**25**]. La ricerca è stata condotta con duplice finalità: riconoscere le attività e quindi il comfort per l'abitante in contesti

abitativi differenti, e minimizzare i costi di manutenzione. Per l'analisi sono state impiegate molte strutture abitative (riportate nella Tabella 2.1) al cui interno risiedono persone in condizioni differenti: chi da solo, chi in due, chi in tre e perfino due casi in cui era contemplata la presenza di un animale domestico. Alcuni dei casi visionati hanno evidenziato che alcune strutture sono state infatti replicate, ma con situazioni abitative, ossia inquilini, differenti. Le caratteristiche anagrafiche e morfologiche dei singoli abitanti sono risultate anch'esse diverse: dal giovane alla persona di mezza età, dall'anziano in salute a quello affetto da demenza; questa ampia casistica si è resa strumento di sfida per riconoscere tramite apparecchiature le attività da situazioni e ambientazioni molto diversi fra loro. Anche il numero di sensori impiegati per ogni struttura abitativa è stato soggetto a variazione, dovuto principalmente alla necessità di adattamento rispetto alla situazione da monitorare, così tale numero varia dalle 20 unità fino alle 86. Le attività il cui riconoscimento è stato studiato sono undici. Il periodo di analisi è stato di un mese per ogni situazione abitativa. Come nel caso precedente, il dataset è pubblicamente disponibile sul sito del progetto [**19**]. Nella Tabella 2.1 sono raccolte le informazioni di alcune delle collezioni di dati estratte all'interno del progetto CASAS, in particolare per ciascuno di essi viene riportato il nome della struttura abitativa, il numero dei residenti ed il numero di sensori installati.

Dataset	Bosch1	Bosch2	Bosch3	Cairo	Kyoto1	Kyoto2	Kyoto3	Kyoto4	Milan	Tulum1	Tulum2
Environment	Bosch1	Bosch2	Bosch3	Cairo	Kyoto	Kyoto	Kyoto	Kyoto	Milan	Tulum	Tulum
#Residents	1	1	1	2+pet	2	2	3	3	1+pet	2	2
#Sensors	32	32	32	27	51	71	86	72	32	20	20

Tabella 2.1: Riassunto delle caratteristiche di parte dei dataset del progetto CASAS.

2.3.3 Progetto Kasteren

Il dataset proposto da Kasteren et al. [**83**] è il risultato dell'analisi svolta su tre strutture abitative intelligenti differenti ma in ciascuna delle quali risiede una sola persona. La Tabella 2.2 riporta le caratteristiche che distingue i tre casi studiati.

	House A	House B	House C
Età	26	28	57
Sesso	Maschio	Maschio	Maschio
Tipo abitazione	Appartamento	Appartamento	Villa
# Camere	3	2	6
Durata	25 giorni	14 giorni	19 giorni
# Sensori	14	23	21
# Attività	10	13	13
Annotazioni	Bluetooth	Diario	Bluetooth

Tabella 2.2: Riassunto informazioni riguardo le tre strutture abitative analizzate.

Come riportato dalla pubblicazione, che spiega il processo di collezione di dati, le persone monitorate (tutti uomini di età compresa fra i 26 ed i 57 anni) si sono proposte volontariamente per la ricerca. Nella tabella precedente viene riportato inoltre il numero di sensori impiegato che varia fra le 14 e le 21 unità a seconda della tipologia di abitazione. I sensori impiegati sono di tipologie differenti: PIR, sensori di contatto, sensori di resistenza, interruttori a mercurio e sensori di galleggiamento; tutti però sono sensori binari. L'analisi ha collezionato dati per un totale di giorni variabili a seconda del caso di studio. Anche le attività riconosciute dipendono dalle differenti situazioni analizzate (la selezione di queste avviene attraverso l'indice Katz, classificazione usata per misurare i gradi di indipendenza nello svolgere ADL in un sistema Health Care [**45**]), di cui è stata richiesta annotazione all'utente attraverso l'impiego di un dispositivo Bluetooth per due casi su tre ed un diario per il restante. Anche in questo caso il dataset è di pubblico accesso sul sito del progetto [**82**]. Di seguito la Figura 2.3 illustra le tre strutture abitative impiegate dal progetto riportando il posizionamento dove i sensori sono stati installati.

Figura 2.3: Mappa delle strutture abitative impiegate dal progetto Kasteren.

2.3.4 Progetto MIT

MIT dataset è creato e utilizzato per la prima volta da Tapia et al. [**79**]nell'ambito di una ricerca finalizzata a testare una procedura di riconoscimento di attività (*Activity Recognition*) sviluppata ad hoc dagli autori. I casi studiati sono quelli di due donne, una anziana e una giovane, che vivono da sole in due appartamenti. I sensori binari installati nei due ambienti abitativi sono 77 per la prima e 84 per la seconda. Il periodo di analisi è stato di 14 giorni per ciascuna abitazione. Anche in questo caso è stato richiesto all'utente che annotasse le attività svolte e che queste si riflettessero sui cambiamenti di stato dei sensori. Per fare questo è stato impiegato un dispositivo che chiedesse all'inquilino ogni 15 minuti quale attività stesse svolgendo e da quanto tempo. Le opzioni di scelta per le

attività sono 35, a tale scopo è stata adattata la categorizzazione Szalai, impiegata a livello internazionale per studi svolti con questo approccio temporale [**76**]. I ricercatori hanno mostrato come questo approccio di annotazione sia incorso in diversi problemi. In primis errori da parte dell'abitante della casa nella selezione dell'attività corretta, poi quelli riguardanti la difficoltà di catturare attività di breve durata a causa della frequenza di richiesta delle annotazioni ed infine il problema della sovrapposizione fra attività. Per superare questi limiti e poter dare un aspetto ben più valido al dataset, sono state adottate delle scelte tra cui l'impiego di apparecchi che catturassero immagini per aiutare l'abitante a ricordare in un secondo momento le attività non annotate, oppure l'inserimento manuale di alcune attività svolte in contemporanea da parte dell'autore. Con queste modifiche il dataset è risultato completo per testare i metodi di riconoscimento di attività sviluppati. Anche questo dataset è disponibile al sito [**78**].

2.3.5 Progetto DOMUS

DOMUS è un progetto sviluppato da parte dell'equipe MultiCom [**43**] per l'omonima piattaforma di Home Intelligence. Per questo scopo è stato impiegato un appartamento all'interno del quale sono stati installati 79 sensori di varia natura, volti ad estrarre singolarmente o in collaborazione informazioni più o meno elaborate. Essi sono principalmente di due tipologie: (a) sensori binari, cioè che riportano stati '1' oppure '0' a seconda che rilevino o meno attività e (b) sensori volti a calcolare i consumi di grandezze fisiche come acqua, gas e luce. La ricerca si è svolta con lo scopo di testare il livello di percezione dell'ambiente intelligente da parte degli inquilini ed ha permesso ai ricercatori di sviluppare un dataset che raccogliesse le informazioni provenienti dai sensori.

Sono stati svolti due esperimenti. Nel primo sono state impiegate 20 persone esterne alla ricerca che a turno restassero per un'ora e mezza all'interno della casa e seguissero un percorso (prestabilito) attraverso le camere dell'appartamento e che ogni 5 minuti riempissero un questionario con le loro percezioni e sensazioni di comfort utilizzando una scala di valutazione da 0 a 10. Nel secondo test invece hanno partecipato in maniera volontaria quattro dei ricercatori del progetto (quindi "familiari" all'ambiente e allo scopo del lavoro), che si sono resi disponibili a trascorrere una notte all'interno dell'appartamento. Le indicazioni in questo caso erano quelle di comportarsi come se si trovassero in un albergo, e a cui dovessero poi lasciare un feedback. I dataset prodotti dal

monitoraggio delle persone all'interno dell'abitazione sono disponibili sul sito del progetto [**81**].

2.3.6 Confronto fra dataset

Fra i dataset disponibili il progetto BRIDGe ha deciso di muoversi verso ARAS in qualità di dataset più funzionale per lo scopo del progetto; questo nonostante le limitazioni riguardo la presenza di due inquilini (e non uno come nel caso ideale) e quelle dovute alle annotazioni. Delle due case offerte da tale dataset è stata scelta la prima (HouseA, Figura2.2) in qualità di ambiente abitativo più vicino al modello ideale. La scelta è stata motivata dalla copertura temporale di trenta giorni per l'analisi e la raccolta di dati dai sensori svolta dagli autori e restituita tramite il dataset che risulta la più ampia rispetto ai risultati offerti dalle altre ricerche che spesso hanno coperto un lasso di tempo inferiore. Altro fattore che ha influito sulla scelta è quella dei sensori impiegati nell'abitazione che rispecchiano le decisioni prese da BRIDGe. Per quanto riguarda i datasets scartati possiamo affermare che CASAS analizzava casi e situazioni che in certe evenienze si distaccavano fin troppo dal caso ideale, Kasteren e MIT invece, pur essendo più simili all'idea del progetto e alle condizioni abitative (una sola persona che risiede nell'abitazione), non fornivano un sufficiente numero di giorni analizzati per poter garantire la robustezza dei metodi sviluppati. DOMUS, in ultimo, oltre a fornire un'analisi tempistica troppo limitata, si distaccava anche concettualmente dallo scopo del lavoro presente.

Le informazioni inerenti le collezioni di dati discusse ed impiegate come discriminatori per la scelta effettuata da BRIDGe sono raccolte nella Tabella 2.3. In questa, per ciascun progetto, vengono riportati i dati relativi al numero di case che sono state impiegate per lo studio, se in esse vi fossero molteplici abitanti, la durata del periodo di analisi, il numero di sensori installati nell'abitazione, ed infine il numero di ADL che il singolo sistema è stato in grado di rilevare.

Nome Dataset	# Case	Più Abitanti	Durata	# Sensori	# Attività
ARAS [**4**]	2	Sì	30 giorni (ciascuna)	20	27
CASAS [**25**]	7	Sì/No	1 mese (ciascuna)	20-86	11
Kasteren [**83**]	3	No	14-25 giorni	14-23	10-16
MIT [**79**]	2	No	14 giorni (ciascuna)	77/84	35
Domus [**43**]	1	No	1:30/8 ore	79	ND

Tabella 2.3: Dataset disponibili.

2.4 Informazioni estraibili dai dataset tramite indicatori

La tesi presente è finalizzata a continuare il percorso di ricerca di nuove tecnologie in grado di impiegare in tempo reale i dati restituiti dai sensori installati in una Smart Home per poter monitorare lo stato di salute della persona che vi risiede.

Per questo scopo sarà necessario sviluppare quindi degli indicatori quantitativi che prendano in input i dati provenienti dai dataset introdotti nel precedente capitolo e produrre dei risultati che diano delle informazioni utili per questa analisi. Così si è ricercato cosa offrisse la letteratura a riguardo e cosa avessero fatto altri studiosi per risolvere questa tipologia di problema.

La bibliografia ha restituito numerose informazioni impiegabili per questo studio, fra queste sono state scelte quelle che sono risultate maggiormente degne di nota e di rilevanza, nonché quelle che potessero soddisfare le necessità espresse dal progetto BRIDGe. Le informazioni scelte sono illustrate nelle sezioni successive di questo Capitolo secondo il presente ordine: indice di inattività JIM (Sezione 2.4.1), indici di benessere (Sezione 2.4.2), tassi per valutare le attività domestiche (Sezione 2.4.3), entropia (Sezione 2.4.4) ed infine cambiamenti comportamentali ed anomalie (Sezione 2.4.5).

2.4.1 Indice del livello di attività

Candàs et al. [**17**] hanno studiato le modalità con cui poter riconoscere deviazioni nel comportamento di un abitante. Per far questo hanno monitorato le attività casalinghe di diverse persone anziane attraverso l'impiego di accelerometri, simili ad orologi e quindi minimamente invasivi. Tramite un singolo accelerometro gli autori hanno monitorato l'attività fisica eseguita dal soggetto prendendo in considerazione una grandezza chiamata

JIM (*Jerk-based Inactivity Magnitude – Grandezza di inattività basata sullo strappo*), essa misura l'attività fisica tramite la formula:

$$(1) \quad JIM = \frac{1}{N}\sum_{i=1}^{N} |Jerk_x[i]| + |Jerk_y[i]| + |Jerk_z[i]|$$

$$(2) \quad Jerk_w(t) = \frac{da_w(t)}{dt} = \frac{d^2v_w(t)}{dt^2} = \frac{d^3r_w(t)}{dt^3}$$

Dove N sono i campioni di accelerazione calcolati ogni secondo; poiché JIM viene calcolato ogni minuto, N sarà 60, come i secondi che compongono un minuto ed in corrispondenza di ciascuno viene estratto un campione per l'analisi. Lo strappo (*jerk*) invece corrisponde al tasso con cui l'accelerazione di un corpo cambia, esso viene espresso nella formula (2) come la derivata dell'accelerazione rispetto al tempo calcolata per ciascuna delle tre dimensioni.

JIM viene quindi interpretata come la frequenza di cambiamento di accelerazione nello spazio della persona monitorata e rappresenta una misurazione del comportamento umano poiché ne calcola il livello di attività.

JIM (finalizzato a dare una misurazione del comportamento umano) è uno strumento utile perché può essere impiegato per vari scopi analitici; fra questi gli autori hanno proposto la creazione di modelli del comportamento umano. I modelli definiti derivano dal JIM calcolato, e quindi vengono generati automaticamente sulla base dei dati passati del livello di attività fisica da cui essi dipendono. Esempi proposti dall'autore sono: il comportamento di un singolo giorno (ottenuto come livello medio di attività), il calcolo della tendenza a lungo termine del livello di attività, ed il comportamento di riferimento calcolato per il singolo giorno. I risultati ottenuti sono successivamente impiegati dagli autori per costruire un algoritmo in grado di adattarsi all'utente e destinato ad analizzare il comportamento umano ed in particolare a rilevare la presenza in esso di anomalie, ossia periodi di tempo all'interno del singolo giorno in cui l'utente abbia un comportamento di gran lunga discostante rispetto a quello di riferimento calcolato dal modello. Con questo lavoro quindi gli autori hanno fornito una risposta utile per l'impiego di sensori (anche se in questo caso indossabili) per l'assistenza sanitaria all'esterno di una struttura medica basandosi esclusivamente sul monitoraggio autonomo del comportamento umano.

2.4.2 Indici di benessere basati sull'utilizzo di elettrodomestici

Suryadevara et al. [**74**] hanno effettuato una ricerca volta a sviluppare un *Predictive Ambient Intelligence* (*PAI – Intelligenza di predizione ambientale*) che utilizzasse le informazioni riguardo le ADL ottenute tramite una rete di sensori wireless (WSN). Questa rete è costituita da due tipologie di sensori: quelli destinati ad apparecchiature elettriche (che ne monitorino quindi l'utilizzo) e quelli per altri scopi (per esempio sensori di pressione per letti, divani, sedie); entrambe le tipologie comunicano utilizzando moduli XBee. Le informazioni che provengono da questi sensori sono impiegate in un sistema di intelligenza autonoma per andare ad effettuare previsioni riguardo il comportamento ed il riconoscimento dei possibili stati di salute della persona in osservazione. L'ambientazione ed il destinatario sono quelli di un anziano che vive in casa da solo.

La procedura che viene mostrata dagli autori per eseguire questo studio prevede un processo di iniziale identificazione delle attività svolte dall'utente per cercare di prevedere le attività successive e tentare poi di creare delle macro-attività che associno al loro interno più attività singole (per esempio fare colazione, pranzare…). La ricerca di attività (singole oppure complesse) risulta per gli autori fondamentale per determinare le condizioni di benessere dell'anziano inquilino, poiché viene ritenuto che tramite l'osservazione dell'utilizzo giornaliero degli elettrodomestici si possa definire lo stato di salute dell'osservato. Pertanto a questo scopo sono introdotte due funzioni atte a calcolare valori che siano indice della condizione dell'utente sulla base dei tempi giornalieri impiegati usando gli elettrodomestici.

$$(1) \quad \beta_1 = 1 - \frac{t}{T}$$

$$(2) \quad \beta_2 = 1 + (1 - \frac{T_a}{T_n})$$

La funzione (1) ricerca il tempo in cui non viene utilizzato un elettrodomestico; i parametri impiegati nella funzione sono:

- t è la durata di inattività attuale in cui non viene sollecitato alcun sensore,
- T è il tempo di massima durata di inattività durante la quale non viene utilizzato alcun apparecchio.

Gli autori, nell'analizzare i risultati restituiti da questa espressione, hanno definito delle soglie per cui poter affermare quale fosse lo stato di salute dell'anziano monitorato. Sui valori assunti da β_1 possiamo definire lo stato della persona monitorata:

$$\beta_1: \begin{cases} \beta_1 < 0.5 & \text{uso anormale dell'elettrodomestico, occorre un controllo} \\ 1 < \beta_1 \leq 0.5 & \text{uso anormale ma non critico dell'elettrodomestico} \\ \beta_1 = 1 & \text{uso perfettamente normale dell'elettrodomestico} \end{cases}$$

La formula (2) invece ricerca eventuale utilizzo eccessivo di uno specifico elettrodomestico; i parametri impiegati sono:

- T_a, la durata dell'utilizzo attuale,
- T_n, la durata dell'utilizzo massimo dell'elettrodomestico in considerazione.

In questo caso la valutazione sulla salute della persona che gli autori hanno effettuato tramite i valori assunti da β_2 è la seguente:

$$\beta_2: \begin{cases} \beta_2 < 0.8 & \text{uso anormale dell'elettrodomestico, occorre un controllo} \\ \beta_2 \geq 0.8 & \text{uso perfettamente normale dell'elettrodomestico} \end{cases}$$

In una pubblicazione successiva [**75**], per questo parametro, la soglia di 0.8 viene sostituita con una di 0.5.

A queste applicazioni vengono poi impiegate serie temporali con lo scopo di procedere all'aggiornamento dei parametri con i cambiamenti comportamentali e di prevedere le variazioni nell'impiego degli elettrodomestici. Con l'analisi introdotta gli autori costruiscono così un procedimento per la valutazione finale riguardo la regolarità od irregolarità di comportamento.

Le conclusioni che i ricercatori traggono sono inerenti il benessere della persona monitorata sulla base del controllo del loro interagire correttamente con gli elettrodomestici in una casa intelligente. Secondo gli autori infatti la valutazione dell'utilizzo corretto o incorretto dei singoli elettrodomestici inserita in un sistema di assistenza sanitaria può aiutare chi si occupa dell'utente (famiglia, tutori, associazioni, medici) nel diagnosticare l'insorgenza di patologie e quindi nell'intervenire per risolvere possibili anomalie riscontrate.

2.4.3 Indici sull'intensità di attività

Yange et al. [**86**] hanno svolto una ricerca finalizzata a monitorare le attività domestiche di una persona anziana che vive da sola allo scopo di estrarre delle informazioni quantitative per valutare lo stato di salute della persona monitorata. Lo studio è stato svolto presso l'abitazione di una donna anziana che vive da sola. Lì sono stati installati due tipi di sensori: (a) dei *passive infrared* (PIR) binari per tenere sotto controllo movimenti, umidità e temperatura, e (b) dei *trasformatori di corrente* (CT) per monitorare le attività delle apparecchiature elettriche presenti in casa. Per motivi di realismo nel monitoraggio e di non condizionamento della vita del soggetto interessato, quest'ultimo non è stato informato dell'installazione dei sensori, così che continuasse a svolgere le sue attività con normalità e secondo la sua routine.

Per analizzare quindi l'attività giornaliera, viene fatto uso di stime (definite come *activity features*) attraverso l'impiego di rapporti su informazioni estratte dalle attività. I dati impiegati sono quelli provenienti dai sensori e sono raccolti ogni 10 minuti, quindi per un singolo giorno saranno raccolti ed impiegati 144 campioni di attività. Le stime o features definite dagli autori sono quattro.

1) **Rapporto del tempo di attività (Activity time ratio):** calcola quanto di frequente la persona monitorata svolga un'attività nell'arco di una giornata. Per fare questo viene impiagata la formula:

$$R_{act-time} = \frac{T_{act}}{T_{act} + T_{ina}}$$

I parametri utilizzati sono:

- T_{act} è il periodo di attività cioè i campioni sui 144 della singola giornata in cui viene rilevata attività da parte dei sensori.
- T_{ina} è il periodo in cui non viene rilevata attività.

Il risultato è un valore compreso fra 0 e 1. Quando il $R_{act-time}$ cresce e si approssima a 1, questo indica un profilo di attività frequente.

2) **Tasso di attività (Activity ratio):** nel caso precedente non viene tenuta in considerazione l'intensità dell'attività svolta, questa stima, così come la successiva, avrà lo scopo di estendere la feature precedente con questa componente.

La formula quindi è:

$$R_{act} = \begin{cases} \frac{\sum_{i=1}^{k} c_i}{T_{act} \times 100} & if\ T_{act} > 0 \\ 0 & if\ T_{act} = 0 \end{cases}$$

In cui

- T_{act} è il periodo di attività come nel caso precedente.
- c_i è un parametro che viene restituito dai sensori che indica il livello dell'intensità di attività per il singolo sensore nell'i-esimo campione di 10 minuti,
- k infine è il numero di campioni presi in considerazione.

Se il risultato è alto significa che l'intensità di attività è maggiore nei campioni di tempo in cui è riscontrata attività, se è zero, allora significa che non c'è attività.

3) **Tasso giornaliero di attività (Daily activity rate):** calcola l'intensità delle attività svolte dall'utente monitorato nel singolo giorno.
   Per definire questo indice viene impiegato l'activity rate del punto precedente con $\boldsymbol{k = 144}$ che rappresenta il numero di campioni presenti in 24 ore

$$R_{act-day} = \frac{\sum_{i=1}^{k} c_i}{144 \times 100}$$

4) **Varianza delle attività giornaliere (Coefficient of variance of daily activities):** calcola una stima di quanto uniformemente il soggetto esegua attività in un giorno, impiegando la deviazione standard dell'intensità di attività $\boldsymbol{c_i}$

$$CV_{act} = \frac{Standard\ deviation\ of\ c_i}{R_{act-day} \times 100}$$

Con l'introduzione di questi strumenti gli autori hanno potuto studiare il profilo di attività a lungo termine per l'abitante ed il riconoscimento di comportamenti anomali. Questo perché con le stime ottenute possono essere costruite finestre temporali di analisi differenti e quindi si possono raggruppare informazioni così da poter determinare quale sia il comportamento medio e quale quello anomalo. Nel testo i risultati calcolati vengono mostrati sia in forma numerica che in forma grafica, visivamente più esplicita.

I risultati ottenuti da questa ricerca sono da ritenere anch'essi come uno strumento utile al fine della valutazione dello stato di salute della persona monitorata, infatti attraverso la ricerca delle stime è possibile sviluppare i profili dei comportamenti e delle attività abitudinarie della persona monitorata. In questo modo è stato possibile fornire uno strumento con un funzionamento analogo ai precedenti, ma con una logica differente, in grado di contribuire ad un sistema di cura a distanza. Quando infatti il comportamento si discosta dal suo andamento abitudinario, e quindi le feature restituiscono dei valori eccessivi che fuoriescono da quelli abituali, il sistema pensato restituisce delle informazioni che possono aiutare interventi in termini di assistenza sanitaria, e quindi vicino all'obbiettivo di BRIDGe.

2.4.4 Entropia dello stile di vita

La pubblicazione di Eagle et al. [**36**] ha l'obbiettivo di sviluppare un sistema in grado di percepire e predire in maniera accurata le azioni di un utente attraverso i dati forniti da un cellulare impiegato come sensore. Per riuscire nell'intento di calcolare la quantità di struttura prevedibile nella vita dell'individuo viene utilizzata l'entropia, una grandezza definita da Shannon nel 1938 per la teoria dell'informazione come la quantità di imprevedibilità in un segnale e descritta dalla seguente formula:

$$H(X) = -\sum_{i=0}^{n} p_i \log_2 p_i$$

In cui X è il messaggio portato dal segnale, n sono i messaggi possibili restituiti dalla sorgente e p_i è la probabilità che il messaggio ha di assumere il valore i-esimo.

Il valore di entropia restituito sarà compreso tra 0 e 1, ed in particolare quando il valore assunto è prossimo a 0, allora il segnale sarà prevedibile e di scarso interesse, mentre quando si avvicina a 1 sarà sempre più imprevedibile (corrispondente a probabilità 0.5) e quindi di ben maggiore interesse. L'andamento grafico dell'entropia è quello di una parabola. Questo è rappresentato dalla Figura 2.4.

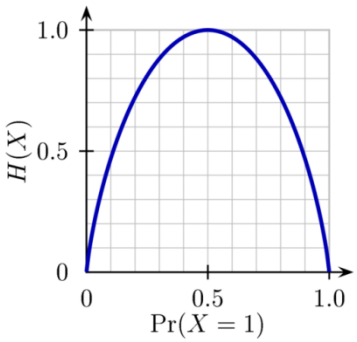

Figura 2.4: Andamento entropia.

I ricercatori hanno quindi utilizzato le informazioni riguardo il posizionamento GPS e le interazioni Bluetooth con altri dispositivi per rilevare la routine oraria giornaliera dell'individuo monitorato. In questo modo si è potuta valutare l'entropia del suo stile di vita e valutarne la prevedibilità o imprevedibilità. Quindi persone il cui stile di vita risulta entropico e perciò molto vario, saranno molto più imprevedibili rispetto a chi ha uno stile di vita a basso carattere entropico, e quindi con una forte presenza di pattern (strutture che si ripetono) nel tempo.

La ricerca quindi ha voluto mostrare come l'entropia relativa alle attività svolte o relativa alla vita del soggetto possa essere applicata come metodo per la valutazione della routine della persona, così come per il confronto di stili di vita. I risultati ottenuti permettono quindi di impiegare l'entropia come misura di valutazione ed autovalutazione della routine della persona monitorata ma anche come strumento per comparare il comportamento di diverse persone. L'applicazione alla quale gli autori aprono con questa ricerca è quella dell'assistenza sanitaria, in particolare, per fornire uno strumento che permetta di identificare variazioni impreviste delle abitudini motivate da possibili situazioni di fragilità di salute, ma anche di riconoscere l'insorgenza di patologie confrontando le abitudini di più persone, sane e malate. Gli autori inoltre ritengono che questi risultati siano utilizzabili anche con scopo predittivo di attività e comportamenti, perché costruiti sugli andamenti passati che mettano in luce i comportamenti abitudinari, ed i relativi pattern, per cui possono essere impiegati anche per predizioni inerenti lo stato di salute della persona monitorata.

2.4.5 Cambiamenti comportamentali ed anomalie

Nella tesi di Gaurav Jain [**47**] viene proposto un modello che si occupa di monitorare la salute di persone residenti in una Smart Home attraverso l'analisi del pattern dello stile di vita dell'abitante. Per questo scopo viene utilizzato un sistema di controllo della salute che raccoglie dati riguardo l'attività domestica utilizzando diverse tecnologie. Le informazioni sono raccolte e impiegate per costruire dapprima una base da utilizzare come riferimento per un'analisi dei pattern che verranno collezionati successivamente. L'analisi quindi si svolgerà ricercando nei pattern dei cambiamenti comportamentali (*drift*) nello stile di vita oppure delle anomalie (*outlier*) nel caso di condizioni di emergenza. Per quanto concerne la ricerca di drift l'algoritmo utilizza un approccio grafico e statistico basato sulla stima dell'autocorrelazione temporale fra finestre di valori per identificarne l'esistenza ed il tipo:

a) *"**ciclico**"* quando è causato da irregolarità o picchi che si ripresentano nel grafico dell'autocorrelazione,
b) *"**discendente**"* quando il trend dell'autocorrelazione cresce oppure decresce
c) *"**caotico**"* quando l'autocorrelazione presenta irregolarità o cambiamenti casuali ed improvvisi.

Per quanto concerne l'identificazione delle anomalie come segnali di avvertimento di situazioni di emergenza l'autore propone due metodi.

1) Il primo è un estensione del metodo utilizzato per identificare i drift: per ogni nuovo dato appartenente a una finestra in analisi si verifica se questo sia un'anomalia utilizzando una standardizzazione per il dato rispetto tale finestra. In questo modo se il valore ottenuto è maggiore di 3 l'elemento sarà considerato anomalo altrimenti no: si considerano anomalie e quindi condizioni di emergenza quelle situazioni in cui il pattern assume valori eccessivamente alti oppure bassi. Questa tecnica però non avviene in tempo reale, ma necessita di una preventiva archiviazione dei dati.
2) Il secondo metodo invece è sviluppato riconoscere questa tipologia di elementi in tempo reale, e quindi immediatamente. Viene definita anomalia un'azione che viene riscontrata non seguire un tracciato atteso e quindi che si allontani dal pattern o dalla routine prevista. Per trovare il pattern atteso viene utilizzato Active LeZi che applica un modello di Markov per predire ottimamente l'azione successiva. Per calcolare il livello di anomalia sono utilizzate due misure.

La prima modalità ipotizza che la misura dell'anomalia dipenda solo dalla probabilità dell'azione successiva; tale misura di anomalia è data dalla formula

$$n1(x) = \begin{cases} 1 & if\ \rho(x) * 100 < 1 \\ \dfrac{1}{\rho(x) * 100} & otherwise \end{cases}$$

dove x è la nuova azione osservata e $\rho(x)$ è la probabilità dell'azione x di essere l'azione successiva.

La seconda misura invece prende in considerazione l'azione corrente x e quella più probabile y, essa sarà descritta dalla formula

$$n2(x) = \begin{cases} 1 & if\ \rho(x) * 100 \leq \rho(y) \\ \dfrac{\rho(y)}{\rho(x) * 100} & otherwise \end{cases}$$

$\rho(y)$ sarà sempre maggiore di $\rho(x)$ e quindi $n2(x)$ potrà essere considerata più selettiva di n1(x). Tutte le anomalie trovate con n2(x) possono essere trovate con $n1(x)$, ma quelle trovate con $n2(x)$ avranno una probabilità maggiore di essere delle effettive anomalie. In questo modo trovando anomalie in tempo reale possono essere sviluppati poi degli allarmi.

In particolare, l'autore impiega i due aspetti estratti (*drift* e *outlier*) per sviluppare dei sistemi autonomi volti ad identificare cambiamenti nello stile di vita della persona con vari livelli di granularità. Tra i benefici che un sistema Healthcare può trarre da questa ricerca vi è l'analisi dei cambiamenti a lungo termine del comportamento dell'abitante della casa, e quindi supportare e dare benefici non solo all'utente, ma anche a chi ne ha responsabilità sociali e sanitarie. Inoltre l'autore nel suo lavoro mostra come questi aspetti analizzati possano tornare utili anche in situazioni di emergenza causate da cambiamenti totalmente inaspettati.

2.4.6 Indici in BRIDGe

Per quanto concerne BRIDGe, questa tesi estrae materialmente tramite indicatori quantitativi delle informazioni utili per lo studio e la definizione delle abitudini della

persona monitorata. A questo scopo è risultato necessario un processo decisionale volto a rispondere al meglio alla richiesta del progetto. Per fare questo è stata analizzata dapprima la struttura abitativa presa in considerazione, quindi la *HouseA* del progetto ARAS con le sue caratteristiche domotiche e le condizioni degli abitanti e, compatibilmente con questo, è stata poi verificata l'utilità o applicabilità degli indici o delle informazioni che la letteratura ha offerto.

L'obbiettivo che la tesi si prefigge è pertanto quello di restituire al progetto BRIDGe un'iniziale serie di informazioni per poter effettuare l'analisi della routine della persona monitorata così da permettere di adottarle per il funzionamento del sistema complessivo. Allo scopo di estrarre questo tipo di informazioni, si è deciso di costruire gli indicatori cominciando da aspetti più concreti, non presenti in letteratura, come la rilevazione dei consumi domestici e lo svolgimento di attività semplici. Successivamente il livello di astrazione è aumentato e si è ispirato ad alcuni aspetti rinvenuti in letteratura, ed in particolare alla valutazione della frequenza con cui viene effettuata un'attività complessa, e la valutazione dell'entropia - applicando differenti finestre temporali - relativa ad un singolo sensore e successivamente alla totalità dei sensori installati nell'abitazione. In questo modo viene offerto a BRIDGe un primo strumento di analisi comportamentale con una buona componente di eterogeneità di informazioni.

Capitolo 3

Materiali e metodi

Questo capitolo offre una panoramica sugli strumenti impiegati per svolgere l'analisi in tempo reale dei dati in ingresso al sistema provenienti dai sensori. Nella prima parte (Sezione 3.1) sarà affrontato il tema dell'elaborazione degli eventi (il cosiddetto *event processing*), analizzando come questo tema sia stato studiato nella letteratura e quali metodi possono essere adottati a questo scopo. La sezione si concluderà con la descrizione approfondita di Esper, la soluzione software adottata nella tesi presente. Successivamente saranno discusse le modalità con cui i dati forniti dai dataset possono essere impiegati per simulare un flusso proveniente dai sensori installati in un'abitazione (Sezione 3.2). Infine sarà trattato SHARON, un sistema sviluppato all'interno al progetto BRIDGe in grado di simulare il comportamento umano in un'abitazione (Sezione 3.3).

3.1 Event Processing

Nella teoria dell'informazione, l'informazione ha un ruolo fondamentale nel momento in cui si debbano prendere delle decisioni, in particolare se queste vengono prese in ambiti come la finanza, la difesa e la salute. L'informazione giunge all'interno di *flussi* sotto forma di messaggi o *eventi*, provenienti da numerose sorgenti che rendono notizia dello stato istantaneo del sistema. Analizzare questi eventi per riconoscere delle condizioni specifiche alle quali rispondere il prima possibile è lo scopo per cui è stata sviluppata una tecnologia detta *Event Processing*.

Il sistema di elaborazione gestisce quindi i flussi in ingresso da più sorgenti rilevando al suo interno singoli eventi oppure pattern (sequenze di eventi nel flusso). Nell'evenienza che le tecniche di elaborazione adottate siano in grado di combinare o correlare eventi semplici per dedurre informazioni più articolate, l'elaborazione prenderà

il nome di *Complex Event Processing* (CEP, elaborazione di eventi complessi, generati dall'aggregazione di eventi più semplici). La duplicità fra eventi semplici ed eventi complessi definisce due gradi di ricchezza informativa per cui in talune situazioni basterà trattare con i primi, mentre in altre saranno preferiti i secondi [**29**]. Sono molte le tecniche sviluppate per il CEP, ma l'architettura impiegata è nella maggior parte dei casi quella introdotta nella Figura 3.1.

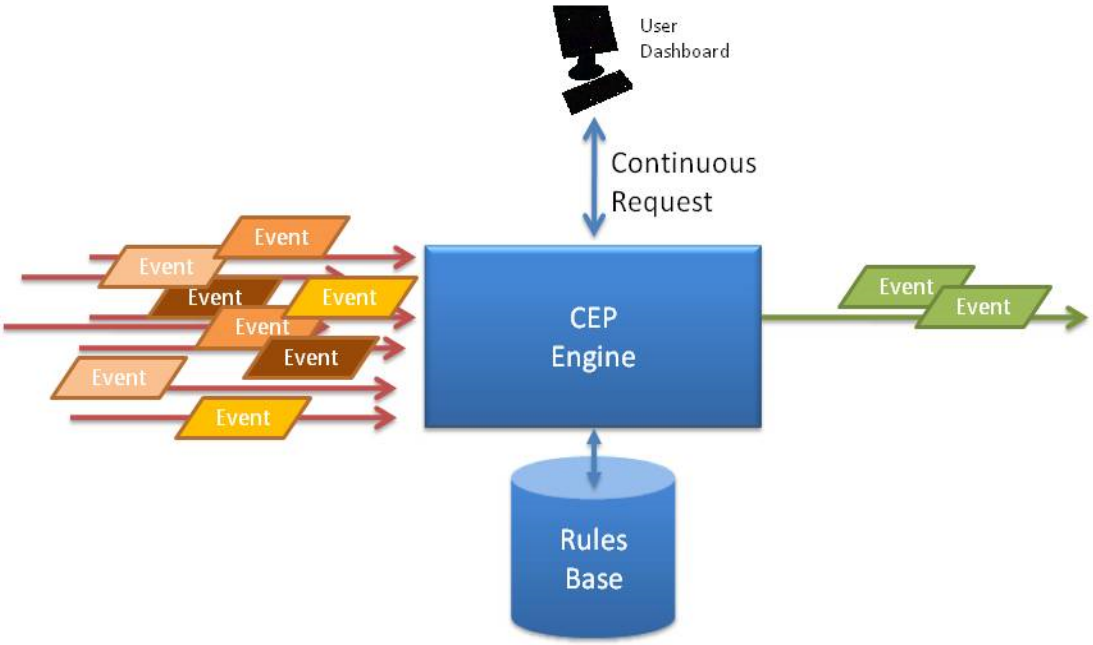

Figura 3.1: Architettura generale CEP.

Il generale funzionamento di un sistema di Complex Event Processing impiega un insieme di regole predefinite che vengono utilizzate da un motore (CEP engine) per analizzare gli eventi dei flussi in ingresso. Gli eventi infatti sono "filtrati" attraverso queste regole e, se vengono riscontrate le condizioni dettate da esse, vengono prodotti i risultati in termini di eventi richiesti [**41**]. A questo scopo il CEP Engine comunica ad un Esecutore di Decisioni (*Decision Executor*) il quale esegue quanto stabilito per la condizione trovata [**69**]**.**

3.1.2 Storia dei sistemi di Event Processing

Come introdotto nella sezione precedente, per effettuare l'elaborazione di eventi sarà necessario adoperare un motore che esegua l'analisi sul flusso di dati in ingresso al sistema. La letteratura presenta numerosi esempi di motori (CEP engine) sviluppati ed adottati con lo scopo di eseguire al meglio l'event processing.

L'event processing nasce come evoluzione del trattamento di dati complessi nei DBMS: l'elaborazione di eventi complessi infatti interpreta i dati che fluiscono nel sistema come notifiche di avvenimenti esterni che debbono essere combinati per comprendere ciò che sta accadendo. Varie famiglie di modelli sono nati in seguito all'introduzione dei DBMS: da quelli per il trattamento delle basi di dati, a quelli commerciali, fino ad arrivare a quelli inerenti i CEP, e quindi di interesse per la ricerca attuale. Di seguito vengono proposti brevemente i principali modelli che coprono cronologicamente la storia della ricerca in questo ambito.

Il primo modello introdotto, il *Sistema Tradizionale Publish-Subscribe Basato su Contenuto* [**39**], [**63**], è il sistema sul quale si sono sviluppate le basi dell'elaborazione di eventi complessi: analizza un sistema di comunicazione asincrona fra mittente (*publisher*) e destinatario (*subscriber*). Esso infatti prevede l'impiego di semplici regole (dette *subscription*) volte ad estrarre, sulla base del contenuto, singoli elementi di interesse da un flusso di informazioni che raccoglie messaggi provenienti da varie sorgenti. Le sorgenti sono i mittenti del modello mentre i destinatari delle informazioni estratte sono gli utenti. Diversi modelli con proprie sintassi sono stati sviluppati, SIENA [**18**], Gryphon [**73**], Xnet [**21**], JORAM [**12**], ma il potere espressivo si presenta in tutti i casi molto simile.

Rapide [**56**] [**57**] è il primo sistema costruito con l'idea attuale di elaborazione di dati: permettere un'analisi che prenda in considerazione la misurazione del tempo e le possibili relazioni fra gli eventi, tenendo traccia della storia di questi. Il sistema adotta un linguaggio che permette l'analisi dei pattern identificati nel flusso attraverso una serie di possibili operazioni (congiunzione, disgiunzione, negazione, sequenza, iterazione) sulla base di vincoli temporali. Rapide però non considera l'esistenza di un tempo assoluto, gli

eventi sono presi con un ordinamento causale fra di essi tenendo traccia della loro intera storia.

GEM [**59**] [**60**] è un linguaggio generalizzato per il controllo di sistemi distribuiti. Questo permette di notificare eventi e di filtrare informazioni dal flusso in ingresso attraverso le stesse operazioni di Rapide, tranne l'iterazione. Con queste operazioni è possibile costruire eventi compositi o complessi. Per il riconoscimento di eventi viene impiegato un procedimento ad albero. GEM inoltre adotta un modello temporale che permette la costruzione di finestre applicabili ai dati in analisi e di considerare gli eventi in base alle loro durate, e quindi di mantenere i riferimenti temporali degli istanti di inizio e di fine. Per ciascuna finestra però è fissato il numero di elementi che vi possono essere contenuti.

Amit [**1**] è uno strumento di controllo in tempo reale per implementare sistemi di reazione veloci ed affidabili. Esso è costituito da una componente detta *situation manager* che processa le notifiche provenienti da più sorgenti per identificare i pattern di interesse ed inoltrarli affinché siano analizzati. Questo attore è costruito con un linguaggio di riconoscimento espressivo ed al contempo flessibile che impiega operatori logici e temporali (tra cui quelli per valutazioni periodiche). Inoltre è introdotto un nuovo concetto di finestra per analisi (*lifespan*) vincolata fra due eventi (uno di inizio e uno di fine) e definibile dall'utente. Amit tuttavia si concentra sui pattern e non permette l'aggregazione di eventi.

Padres [**54**] ha lo scopo di riconoscere gli eventi di interesse non appena questi fanno il loro ingresso all'interno del sistema. A questo scopo impiega le funzionalità del publish-subscribe per mandare notifiche riguardo le informazioni attese, estratte attraverso regole espresse in un linguaggio basato su pattern (patter-based). Il presente sistema però arricchisce il tradizionale publish-subscribe introducendo operazioni logiche combinate con la possibilità di impiegare finestrature temporali fisse.

DistCED [**67**], invece, è un sistema costruito come estensione del tradizionale publish-subscribe basato sul contenuto. Esso applica le stesse operazioni logiche presenti in

Padres e aggiunge la possibilità per l'utente di specificare i tempi di validità in cui effettuare rilevazione di informazioni. A questo scopo vi è la possibilità di costruire finestre personalizzate che scorrano nel tempo. Il processo di elaborazione avviene compilando le regole all'interno di automi a stati finiti impiegati come rilevatori di informazioni.

Sase [**85**] è un sistema di monitoraggio per eseguire interrogazioni riguardo eventi complessi appartenenti a flussi generati da letture RFID (dall'inglese Radio-Frequency IDentification, identificazioni a radiofrequenza) in tempo reale. Esso impiega regole costruite con un linguaggio basato sui pattern costituito da tre clausole:

a) *"**event**"* specifica quali siano le informazioni da estrarre e le relazioni (espresse con operatori logici) fra di esse,
b) *"**where**"* definisce i vincoli sulle singole informazioni, e quindi le condizioni per estrarre quanto ricercato nella clausola *event*,
c) ***"within***" definisce il tempo di validità e quindi la possibilità di creare finestre ad hoc.

Con tale linguaggio quindi viene permesso il riconoscimento di sequenze di eventi, ma non è possibile aggregazioni. Le regole vengono inserite in un processo di analisi con struttura fissa (similmente ad una pipeline) che esegue in maniera sequenziale l'analisi delle informazioni in arrivo.

Esper è una soluzione sviluppata da EsperTech Inc. [**38**]. Esso è un software open source integrato in Java come libreria. Il suo scopo, come quello dei precedenti sistemi introdotti, è di prendere gli eventi che fluiscono nel sistema da varie sorgenti e di estrarre informazioni ricercate non appena sono verificate condizioni prestabilite. Per fare questo Esper è stato dotato di un linguaggio dichiarativo per specificare le regole che gli eventi devono rispettare perché il sistema esegua le azioni volute per l'evenienza. Questo linguaggio è chiamato EPL (Event Processing Language – *linguaggio di elaborazione di eventi*) ed include parte delle operazioni e della sintassi del ben più noto SQL (funzioni di aggregazione, unione, …), ma a queste vengono aggiunti dei nuovi costrutti, per la definizione e gestione di finestre di dati, di output e di pattern.

StreamCruncher [**48**] è un motore software proprietario sviluppato da Jayaprakashcon che esegue elaborazione di flussi di eventi, analizzando questi ultimi sia singolarmente che in relazione fra di essi.

CEDR [**14**] è un sistema in cui vengono impiegati aspetti temporali per analizzare il flusso di informazioni. Si sviluppa una memoria storica riguardo le informazioni ricevute che permette di notificare quando cambiano i risultati di una regola. In questo modo, attraverso un linguaggio basato su pattern si riesce a riconoscere l'occorrenza di eventi specifici. A differenza degli strumenti precedenti, in questo caso le finestre non sono fornite esplicitamente, ma i vincoli di validità temporale sono incorporati negli operatori. In questo sistema le istanze degli eventi hanno un ruolo fondamentale, in base ad esse, infatti, le regole definiscono non solo come gli eventi debbano essere filtrati, trattati e trasformati, ma anche quali politiche di selezione, consumo e validità debbano essere applicate.

Cayuga [**16**] è un sistema di controllo basato su un linguaggio dichiarativo detto CEL (Cayuga Event Language – *Linguaggio per eventi Cayuga*) simile per molti aspetti ai linguaggi impiegati per le basi di dati. Infatti ricorda SQL, dal quale prende diversi elementi di sintassi ed istruzioni (SELECT, FROM ed una PUBLISH per la produzione di output), nonché funzionalità come aggregazione, ridenominazione, unione, ma senza la possibilità di definire finestre. Analogamente al funzionamento in SQL, la clausola FROM viene impiegata per estrarre dal flusso di informazioni gli eventi all'interno di sequenze, questo rende il linguaggio non solo dichiarativo ma anche di rilevazione. L'assenza di finestre obbliga l'utente ad annidare le istruzioni per permettere la considerazione di eventi complessi. Cayuga è costruito esplicitamente per lavorare su larga scala, disponendo di una politica di selezione multipla di eventi.

NextCEP [**71**] è un sistema distribuito di elaborazione per eventi complessi. Esso adotta come Cayuga un linguaggio che riprende gran parte dei meccanismi e delle funzionalità di SQL insieme agli operatori per il riconoscimento di pattern. Il funzionamento però rispetto al caso precedente avviene in maniera distribuita, con una

forte attenzione riguardo l'ottimizzazione delle regole impiegate sulla base della frequenza con cui queste producono output.

Sase+ è stato introdotto dagli stessi sviluppatori al fine di arricchire l'espressività del linguaggio precedente aggiungendo le operazioni di aggregazione ed iterazione, ed incrementando la complessità dell'algoritmo di riconoscimento dei pattern. Viene inoltre introdotta la possibilità per l'utente di stabilire le politiche di selezione degli eventi validi (per esempio: solo il prossimo evento se viene soddisfatta una regola particolare, oppure solo il prossimo evento che soddisfa la regola, oppure ancora tutti gli eventi che soddisfano una regola, …).

Peex (Probabilistic Event EXtractor – Estrattore di eventi probabilistici) [**50**] è un sistema sviluppato per estrarre eventi complessi da dati prodotti da un sistea RFID. Viene adoperato un linguaggio basato sui pattern la cui sintassi prevede quattro clausole: FORALL definisce l'insieme di letture destinate alle regole, WHERE che specifica i vincoli sui pattern, CREATE EVENT e SET definiscono il tipo di eventi da generare e decidono il contenuto degli attributi. Sono previste congiunzioni, negazioni, sequenze ma non iterazioni. Il fine di questo modello è dare un supporto all'incertezza dei dati, ossia riconoscerne errori e ambiguità. Per fare questo, all'utente viene data la possibilità di assegnare un valore probabilistico alle informazioni ricercate, associare un parametro di confidenza alle occorrenze e dare valori di importanza per gli eventi composti. Il sistema concede il riutilizzo di dati passati al fine di mantenere aggiornati i valori da attribuire. In questo sistema sono previsti pure eventi parziali dovuti a errori di trasmissione o di riconoscimento di eventi.

PB-CED (Plan Based Complex Event Detection – *Riconoscimento di eventi complessi secondo un piano*) [**3**] è un sistema sviluppato con lo scopo di limitare la trasmissione di dati inutili. Impiega un semplice linguaggio di rilevazione con operatori di congiunzione, disgiunzione, negazione e sequenza ma senza iterazione e senza finestre esplicite, perché incorporate nei vincoli temporali degli operatori. Il funzionamento permette di evitare la propagazione degli eventi più frequenti compilando delle regole che filtrino ed estraggano informazioni esclusivamente da quelli la cui frequenza sia inferiore.

Raced [**30**] è un sistema di elaborazione di eventi basato su di un semplice linguaggio di riconoscimento che presenta funzionalità simili a quelle di Padres ma che impiega, per le richieste di informazioni, una struttura gerarchica ad albero per la rilevazione della frequenza degli eventi. Il sistema in questo modo permetterà di evitare, come in PB-CED, la trasmissione di eventi non necessari.

T-Rex [**28**] è un sistema sviluppato per bilanciare espressività ed efficienza. Esso impiega un linguaggio di riconoscimento chiamato TESLA [**31**] per esprimere le regole che permettono di definire eventi complessi da quelli semplici. Questo linguaggio è stato sviluppato con limitate funzionalità (selezione di eventi singoli, sequenze vincolate dal tempo, aggregazioni e negazioni) allo scopo di mantenerlo semplice, compatto ma altamente espressivo. Esso è dotato inoltre di un timer che permette di definire regole periodiche. Inoltre possono essere espresse gerarchie di eventi e politiche di consumo. T-Rex è implementato in C++ ed esegue l'elaborazione in maniera incrementale, quindi come un nuovo evento entra nel sistema.

RTEC (Event Calculus in Real Time – *Calcolo di eventi in tempo reale*) [**5**] è un motore sviluppato da Alevizos et al. per le applicazioni Big Data [**6**] [**7**], basato sul formalismo logico Event Calculus. Questo estende il potere espressivo dei sistemi logici [**51**] e permette di passare dalla tradizionale gestione delle basi di dati, a quella più ricercata di gestione di eventi che si manifestano nel tempo. Attraverso la costruzione di regole specifiche e di finestre, questo modello gestisce le relazioni che intercorrono fra gli eventi allo scopo di estrarre, da ciò che viene riconosciuto rilevante, i risultati ricercati.

ruleCore [**72**] è uno dei più recenti motori sviluppati, esso impiega per il riconoscimento di sequenze di eventi un algoritmo presentato sotto forma di albero [**20**], le cui foglie corrispondono agli eventi primitivi (quelli semplici) mentre i nodi agli operatori che vengono utilizzati per mettere in relazione questi eventi al fine di costruire quelli complessi. Il riconoscimento dei pattern e delle condizioni di interesse all'interno del flusso avviene quando la radice assume stato positivo (vero). Perché questo avvenga, i nodi devono prima essere a loro volta veri. Questo succede quando le foglie di tali nodi diventano vere rispettando il senso dell'operatore (se l'operatore è un "and" entrambe le

foglie devono essere vere, per un "or" ne basta una sola). Quando la radice dell'albero rappresentante la condizione è vera, il risultato della condizione analizzata viene salvato in un database impiegato per l'apposita conservazione delle informazioni.

3.1.3 Esper

Fra i numerosi sistemi sviluppati si è deciso di proporre al progetto BRIDGe una soluzione che adottasse Esper. La decisione è stata influenzata da due aspetti: innanzitutto lo scopo della tesi e poi le valutazioni attribuite al software da altri studiosi che lo hanno scelto per lavorare in situazioni analoghe.

La tesi presente si prefigge di identificare degli indicatori quantitativi per la caratterizzazione del comportamento abitudinario di una persona monitorata: questo obbiettivo rende Esper uno strumento adatto. Un indicatore, una volta formulato a livello teorico ed adeguatamente progettato, non è altro che un'interrogazione EPL: esso applica operazioni logiche, algebriche o quantitative su un insieme di dati selezionati ed aggiornati in tempo reale. In questo modo Esper ed EPL permettono di costruire un sistema che, in base agli eventi verificati in un flusso in ingresso, genera degli eventi complessi che rappresentino tali indicatori implementando le operazioni precedenti in maniera flessibile, senza significative limitazioni, né dal punto di vista modellistico/architetturale, né dal punto di vista implementativo (essendo basato su Java).

Per quanto concerne le esperienze di altri studiosi o ricercatori che hanno adottato questo strumento, Dekkers et al. [**34**] riconosce ad Esper un grado di maturità maggiore rispetto agli altri motori perché questo presenta un buon livello di integrazione fra l'elaborazione di flussi di eventi e quella degli eventi complessi. Inoltre gli autori si soffermano su una valutazione positiva sia del linguaggio EPL definito breve ed intuitivo sia delle performance che questo può raggiungere anche su dispositivi non di ultima generazione. Secondo Romero et al. [**69**] Esper viene scelto rispetto ad altri strumenti perché definito stabile, efficace e semplice da utilizzare per la gestione di eventi e per prendere decisioni. Foley [**41**] preferisce Esper per la sua estesa documentazione e la garanzia di supporto offerta dalla vasta comunità online.

3.1.3.1 La sintassi EPL

Come è stato affermato precedentemente Esper è una soluzione software integrata in Java che, adottata dalla tesi presente, permette di impiegare il linguaggio EPL allo scopo di operare in tempo reale sugli eventi in ingresso al sistema per costruire eventi complessi che rappresentino gli indicatori ricercati per descrivere la routine della persona monitorata. In questa sezione viene illustrata la sintassi del linguaggio EPL e come essa operi sugli eventi selezionati attraverso la costruzione di interrogazioni.

La struttura sintattica di EPL presenta una forte somiglianza con SQL, esso è costruito utilizzando una serie di clausole, molte delle quali con lo stesso funzionamento:

```
[insert into insert_into_def]
select select_list
from stream_def [as name] [, stream_def [as name]] [, …]
[where search_conditions]
[group by grouping_expression_list]
[having grouping_search_conditions]
[output output_specification]
[order by order_by_expression_list]
[limit num_rows]
```

Come per SQL, in EPL le uniche due clausole obbligatorie per una regola sono *"select"* e *"from"*, poiché estraggono un'informazione e ritornano il dato voluto. Le restanti clausole sono tutte facoltative, tra queste *"output"* e "*limit"* entrambe per la gestione dei risultati. La clausola *"output"* viene impiegata per il controllo della frequenza con cui sono restituiti i risultati; a questo scopo la condizione *output_specification* può impiegare la sintassi seguente:

```
output_specification: [all|first|last|snapshot] every
output_rate [seconds|events]
```

Così sono restituiti tutti, il primo oppure l'ultimo dei risultati calcolati per ogni secondo di analisi oppure per ogni evento. La clausola *"limit"* viene utilizzata per limitare con *num_rows* il numero di risultati prodotti.

Una differenza netta rispetto a SQL è la possibilità di introdurre nella clausola *"from"* una sintassi che permetta l'estrazione di un pattern (una singola sequenza di eventi) verificando che questo soddisfi condizioni di ricerca e di interazione fra di essi.

```
from pattern
[[every|every_distinct|until] stream_def [as name] (search
conditions)
[[and|or|not|followed_by] stream_def [as name] (search
conditions)[…]]
[where [timer:within|timer:withinmax](timer_rate)]]
```

Per individuare pattern si possono usare diversi operatori. *"Every"* è un'istruzione per attivare l'estrazione ogni volta che un pattern verifica una condizione specificata. Fra le condizioni vi sono anche gli operatori logici *and, or, not* e temporali *followed by* (->) impiegati per verificare relazioni fra pattern. Vi è infine la possibilità di controllare le sottoespressioni impiegando le condizioni temporali *timer:within* o *timer:withinmax*, interne alla clausola *"from"* in un'apposita *"where"* che vincoli temporalmente l'esecuzione dell'analisi.

3.1.3.2 Le finestre

Esper permette di applicare delle finestre sui flussi di dati per eseguire analisi sugli eventi. Questa soluzione può agire su come venga valutata l'interrogazione in questo modo:

- Con l'opzione *length(size)* viene fissato dall'utente il numero massimo di eventi che possono essere considerati contemporaneamente dal flusso per l'analisi.
- *time(time_period)* indica, invece, il tempo che la finestra mantiene gli eventi giunti al sistema. Questi verranno raccolti man mano che arrivano e quando sarà passato il tempo *time_period*, verranno rilasciati.

Oltre a queste due, sono impiegabili molte altre tipologie di finestre in grado di rispondere a diverse caratteristiche sia del flusso che degli eventi stessi. Questa ricchezza è uno degli aspetti che permette ad Esper di emergere fra le altre soluzione di event processing.

3.1.3.3 L'architettura

Utilizzando la sintassi presentata nelle sezioni precedenti si possono costruire delle interrogazioni che vengono collezionate all'interno del motore Esper e da questo impiegate per gestire ed analizzare in tempo reale i dati proventi dai flussi. Queste interrogazioni sono atemporali, ossia continue: esse vengono applicate ogni volta che un nuovo evento del flusso entra nel sistema. L'applicazione delle interrogazioni avviene in maniera "inversa" rispetto a quanto succede con le basi di dati. Se nelle basi di dati infatti una diversa interrogazione era applicata di volta in volta sui dati, in questo modello ad ogni istante in cui sopraggiungono nuovi dati, su di essi vengono effettuate le interrogazioni, in modo che siano filtrati per verificare se soddisfino le condizioni richieste dalle regole. Facendo un paragone: i flussi sostituiscono le tabelle e gli eventi le tuple. Alle interrogazioni può essere associato un oggetto Java (POJO, Plain Ordinary Java Object) appartenente ad una classe speciale detta "listener" che esegue, solo nel caso in cui sia verificata l'interrogazione, l'azione richiesta dall'evenienza (per esempio una notificazione, una scrittura su file, il calcolo di un'operazione, etc) [**41**]. L'architettura appena introdotta è illustrata nella Figura 3.2.

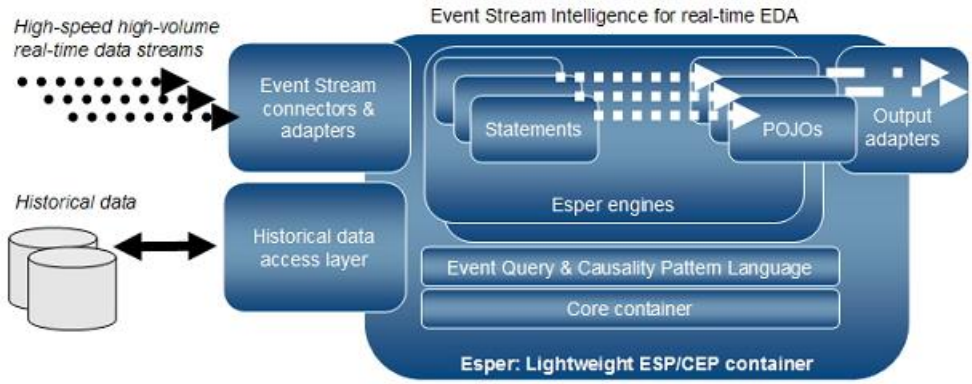

Figura 3.2: Architettura Esper.

3.2 Generatore di Eventi

L'impiego di uno strumento di simulazione permette al presente lavoro di tesi di sviluppare gli indicatori in un contesto simil-realistico che possa verificarne il corretto funzionamento ed essere facilmente adottato poi da un vero sistema di monitoraggio di una casa.

Per poter applicare con successo Esper ed i suoi costrutti EPL, è necessario avere in ingresso al sistema dei flussi di eventi originati dai sensori installati in un'abitazione intelligente. Disporre di questo genere di dati tuttavia richiede denaro, per l'acquisto delle apparecchiature, e tempo per la loro installazione e lo sviluppo dei sistemi competenti. Quindi una soluzione in un macroprogetto, quale BRIDGe, è quella di sviluppare diversi aspetti in parallelo per poi "assemblarli". In attesa che quindi BRIDGe disponga di una struttura abitativa propria dalla quale ricavare in tempo reale informazioni, una buona soluzione consiste nel costruire uno strumento di simulazione che, utilizzando i dataset introdotti nel Capitolo 2 Sezione 2.3, emuli il flusso in ingresso. Questo permette alla tesi presente di sviluppare un sistema di corretta analisi degli eventi e restituzione degli indicatori che, in un secondo momento, potrà essere adottato con dati provenienti all'istante da una vera struttura abitativa.

Per questo viene costruito un Generatore di Eventi: un programma Java che prende un singolo documento testuale alla volta dal dataset scelto, ne legge il contenuto riga per riga e lo invia immediatamente al sistema. In questo modo viene simulato il monitoraggio della struttura abitativa impiegata dal dataset, costruendo un flusso di eventi che riporta lo stato istantaneo dei sensori. Sotto forma di flusso quindi i dati potranno entrare nel processo di elaborazione ed essere effettivamente analizzati dal sistema di interrogazioni EPL per estrarre le informazioni cercate.

3.3 SHARON

SHARON (Simulator of Human Activities, Routines and Needs – *Simulatore di attività, abitudini e bisogni umani*) è un sistema di simulazione di vita in abitazione sviluppato da Proserpio et al. [**68**] all'interno del progetto BRIDGe. I contributi di questo sistema sono due:

a) introdurre un *simulatore* che riproducesse in maniera autonoma ed indipendente le abitudini umane,
b) produrre dei *dataset* sintetici, configurabili per scopi generici.

Per lo sviluppo di SHARON si è reso necessario impiegare un dataset per la validazione dei risultati, a tale scopo, anche in questo caso, è stato adottato il dataset ARAS poiché considerato il più simile alle necessità del progetto BRIDGe.

Il sistema è stato sviluppato per essere in grado di permettere a chi lo utilizza di configurare le ADL (attività di vita quotidiana) a propria discrezione. Nella trattazione l'autore esegue una riduzione delle attività (accorpandole) passando da 27 di ARAS a 17. Il processo decisionale con cui il sistema SHARON simula la successione di attività eseguite si basa sulla valutazione di tre parametri:

a) I *bisogni* rappresentano l'aspetto fisiologico e psicologico che muove una persona a scegliere un'attività piuttosto che un'altra.
b) La *dipendenza temporale* definisce i profili abitudinali dell'abitante nello svolgere specifiche attività.
c) L'*influenza probabilistica* definisce la capacità umana di scegliere cosa fare con un certo grado di incertezza, approssimazione e casualità.

In conclusione, sulla base della continua valutazione di questi tre parametri, SHARON decide in ogni istante quale sia l'attività da simulare. Il risultato è l'emulazione realistica dello svolgimento di ADL per una persona nella sua vita quotidiana all'interno della propria abitazione.

3.3.2 L'estensione

SHARON, con i suoi risultati, permette al progetto BRIDGe e ad altri utilizzatori di avere a disposizione la sequenza di attività di una persona nella sua quotidianità. Tuttavia questo risultato non è sufficiente per la tesi poiché si ha bisogno dei dati provenienti dai sensori (installati in un'abitazione intelligente) per poterli processare tramite costrutti EPL. A questo scopo è stata introdotta un'estensione di SHARON, basata sulla simulazione di un ambiente intelligente, in cui un agente segue dei pattern di movimento e attiva determinati sensori durante lo svolgimento delle ADL: in questo modo vengono prodotti i dati relativi allo stato dei sensori, necessari per la tesi.

Un ulteriore lavoro, sviluppato da Masciadri et al. [**61**] sempre all'interno del progetto BRIDGe ha introdotto un metodo basato su un modello stocastico per la riproduzione e l'estensione dei dati esistenti: validando i parametri di SHARON durante l'emulazione di

mesi di attività, gli autori sono riusciti anche a simulare un cambiamento comportamentale di entità nota. Questo sarà un valore aggiunto ai dati che permetterà di riconoscere tali variazioni nelle informazioni estratte con gli indicatori.

3.3.3 Dataset di dati simulati

SHARON, nella sua versione originale prima, ed in quella estesa poi, ha fornito il progetto BRIDGe, e la tesi presente, di due interessanti ed utili dataset. I dati prodotti infatti permetteranno di superare due limitazioni presenti invece in quelli ARAS: avere una durata di gran lunga maggiore (90 giorni, contro i 30) e includere una persona sola (invece di due). Impiegando un dataset sintetico prodotto con SHARON, dà la possibilità, infatti, di riconoscere i cambiamenti comportamentali inseriti nella simulazione dei dati tramite le informazioni estratte dagli indicatori sviluppati con Esper ha portato ad adottare questo dataset. Il progetto tuttavia, verrà anche testato utilizzando dati reali dal dataset ARAS.

I dati simulati pertanto prodotti impiegando SHARON descrivono lo stato dei 20 sensori dell'abitazione *HouseA* del progetto ARAS in due circostanze: quella in cui avvengono cambiamenti comportamentali e quella in cui questi non si verificano. In entrambi i casi le attività simulate sono 17.

In questo capitolo sono stati presentati gli strumenti che sono risultati necessari per lo sviluppo della tesi presente. Dapprima è stata introdotta la tecnologia dell'*Event Processing* che, analizzando un flusso di dati provenienti dai sensori installati in un ambiente intelligente, permette anche di definire degli *indicatori quantitativi*, misurazioni del comportamento della persona che risiede nell'abitazione considerata. Tra i numerosi metodi adottabili, per la tesi è stato scelto *Esper*, una soluzione software che attraverso l'impiego di un linguaggio, detto *EPL*, permette di progettare indicatori in termini di interrogazioni (simili a quelle di SQL). Successivamente è stato introdotto un elemento fondamentale per l'esecuzione di un sistema Esper quando, come in questo contesto, non è alimentato direttamente da un'abitazione intelligente: un *Generatore di Eventi* che legge i dataset adottati e simula la provenienza di tali dati dai sensori dell'abitazione da cui i dati sono estratti. Tra questi dataset, per la validazione del lavoro svolto, è stato adottato in primo luogo il dataset ARAS (Capitolo 2, Sezione 2.3.1), successivamente è stato

impiegato quello prodotto dal sistema *SHARON*, sviluppato nell'ambito del progetto BRIDGe, il quale produce dati sintetici che superano le limitazioni del primo dataset e permettono anche di rilevare i cambiamenti comportamentali.

Capitolo 4

Indicatori per la valutazione della routine di attività semplici

Nel Capitolo 2, in particolare nella Sezione 2.4 sono stati introdotti alcuni indici che possono essere estratti dai dati che fluiscono nel sistema tramite delle strutture dette *indicatori*. In questo capitolo ciascun indicatore sviluppato estrarrà una tipologia di informazione specifica. Per rispondere ai bisogni del progetto BRIDGe di dati per l'analisi delle abitudini e dello stile di vita della persona monitorata, la tesi ha selezionato un gruppo di informazioni che risultassero utili. In questo modo sono dapprima state identificate delle misurazioni che potessero rendere notizia dei consumi domestici (come per esempio acqua, gas o corrente elettrica) e attività semplici all'interno della routine dell'abitante. Successivamente sono state selezionate alcune delle informazioni presentate nella Sezione 2.4. Delle informazioni che la letteratura ha offerto, ne sono state scelte due:

- la valutazione dell'*entropia*, allo scopo di definire il carattere dell'abitudinarietà dello stile di vita della persona.
- La valutazione dell'*Activity Time Ratio*, per misurare quanto tempo della singola giornata viene trascorso svolgendo un'attività complessa.

Lavori futuri potranno cercare informazioni ulteriori ed applicare altri risultati che la letteratura ha mostrato e che qui non sono stati adottati.

4.1 Preparazione dei dati e introduzione agli indicatori

Il lavoro presente quindi, come affermato, si è rivolto principalmente a ricavare tipologie eterogenee di informazioni che potessero essere estratte tramite costrutti EPL offerti dal

software *Esper*. In questo modo si ha a disposizione un sistema per processare in tempo reale il flusso di dati provenienti dal *dataset* scelto (simulando, tramite l'impiego di un Generatore di Eventi, la provenienza continua dai sensori installati nell'abitazione), così che vengano restituite informazioni utili per valutare comportamenti abitudinari, e qualità di vita dall'abitante della casa.

Gli indicatori sono costruiti impiegando talvolta un corposo numero di interrogazioni (*query*) e potranno presentare struttura complessa, quindi, essi verranno introdotti uno ad uno illustrando le interrogazioni principali che li compongono, e ricorrendo all'occorrenza a diagrammi di flusso per semplificare la comprensione del funzionamento.

Come affermato nei capitoli precedenti, i dataset impiegati sono due:

- *ARAS* (Sezione 2.3.1),
- *Dati simulati* prodotti impiegando SHARON, quindi all'interno del progetto BRIDGe (Sezione 3.3.3).

Per entrambi i dataset la struttura abitativa presa in considerazione è quella offerta dal progetto ARAS ed in particolare è l'appartamento *HouseA* (Figura 2.2) caratterizzato da 20 sensori, ciascuno dei quali identificato dal numero riportato nella mappa corrispondente.

I sensori che vengono impiegati in questo dataset sono: 6 fotocellule (funzionamento analogo ai PIR), 6 sensori di distanza, 3 sensori resistori (alla pressione), 3 sensori di contatto, 1 di temperatura ed 1 ricevitore infrarosso. Il formato con cui ARAS e SHARON consegnano i dati destinati ad essere impiegati come input per un sistema di *Complex Event Processing* (come Esper) è attraverso una serie di file testuali denominati DAY_X.txt (dove X sono i numeri da 1 a 30 –per ARAS- oppure da 1 a 90 –per SHARON-, corrispondenti ai trenta o novanta giorni analizzati). All'interno, questi dati presentano una struttura matriciale costituita da 86400 righe e 22 colonne.

Le righe rappresentano i secondi che compongono la singola giornata, le prime venti colonne invece rappresentano i venti sensori installati nell'abitazione. Nel dataset ARAS le ultime due colonne rappresentano le annotazioni che i due inquilini hanno restituito al sistema corrispondenti alle attività svolte per ciascun istante; nel caso di SHARON l'inquilino dell'abitazione è uno solo, quindi vi sarà una sola colonna (la ventunesima) destinata alle sue annotazioni, la colonna ventiduesima invece ospiterà esclusivamente zeri. Tali annotazioni avvengono riportando tramite un applicativo apposito il numero

identificativo della specifica attività svolta, in ARAS sono riconosciute 27 ADL, in SHARON 17 oppure 14 (a seconda che si prenda in considerazione la versione standard oppure l'estensione). All'interno della matrice, per ogni secondo e per ogni sensore, viene restituito lo stato istantaneo di ciascun sensore installato nell'abitazione, questo potrà restituire due tipi di messaggio: '0', in cui il sensore è inattivo e non rileva attività, oppure '1', in cui il sensore è attivo e sta rilevando attività.

Di seguito in Figura 4.1 è riportato uno stralcio di contenuto di un file appartenente al dataset ARAS: ogni riga corrisponde a un secondo di analisi, ogni colonna ad un singolo sensore e le ultime due colonne, che sono state simbolicamente separate tramite una linea grigia, corrispondono alle annotazioni. La Figura 4.2 riporta invece un esempio di contenuto per un file appartenente al dataset generato impiegando SHARON. Come si evince facilmente, la struttura è identica, con la differenza che in questo caso per le annotazioni viene impiegata solo la ventunesima colonna perché il sistema simula la presenza di un solo inquilino nell'abitazione.

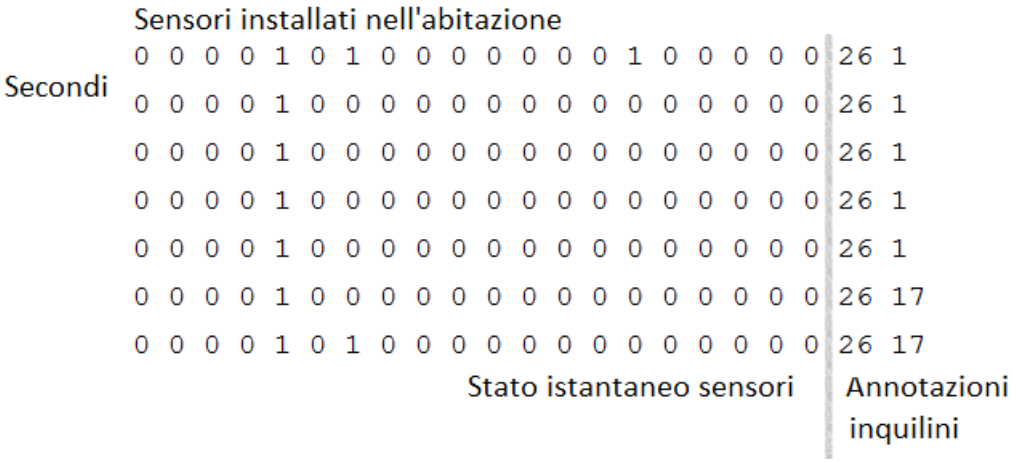

Figura 4.1: Esempio documento dataset ARAS.

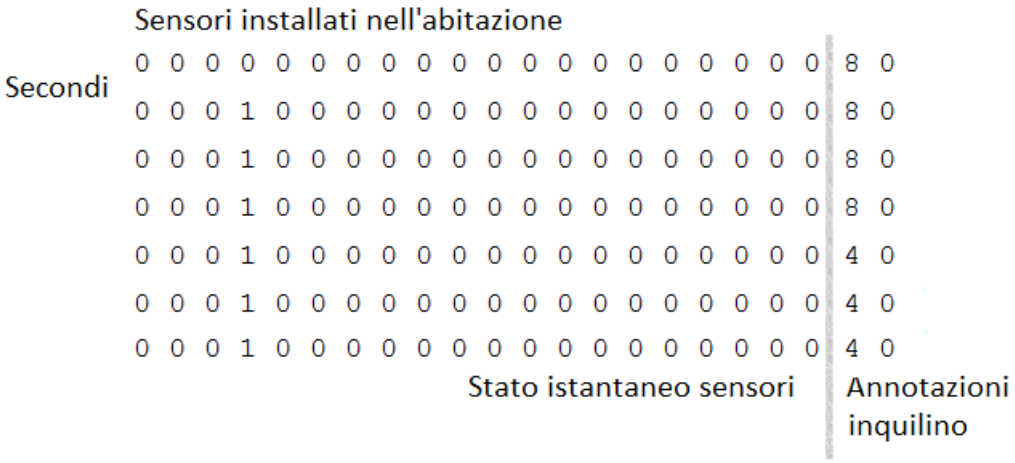

Figura 4.2: Esempio documento SHARON.

Sarà infatti leggendo una alla volta tutte le righe di ciascun file che il Generatore di Eventi introdotto precedentemente (Sezione 3.2) andrà a creare un flusso continuo di dati come se questi provenissero realmente dai sensori, così da permettere l'implementazione e la verifica del corretto funzionamento degli indicatori quantitativi presentati in questo e nel prossimo capitolo per estrarre informazioni in tempo reale

Gli indicatori sviluppati nel capitolo presente e nel successivo Capitolo 5 lavoreranno su questi flussi di dati allo scopo di trovare elementi ed informazioni eterogenee per la valutazione delle condizioni della persona monitorata. Pertanto saranno estratte informazioni con diversi gradi di astrazione e complessità che facciano luce riguardo le attività, i consumi casalinghi e le abitudini sia a livello del singolo sensore sia rispetto alla visione abitativa d'insieme. Gli indicatori appositamente costruiti a questo sono divisi in due capitoli, in questo vengono presentati i due più semplici, volti ad estrarre aspetti concreti sulla vita quotidiana della vita:

- **ITASS**: *Indicatore per il calcolo del Tempo di Attivazione di un Singolo Sensore* (Sezione 4.2).
- **ISCS**: *Indicatore per Stimare il Consumo di Servizi domestici* (Sezione 4.3).

Nel Capitolo 5 sono estratti aspetti più astratti e complessi:

- **ISVUS**: *Indicatore di Stile di Vita rispetto ad Un Sensore* (Sezione 5.1)
- **IRAC**: *Indicatore per Riconoscre un'Attività Complessa* (Sezione 5.2).

- **ISVIS**: *Indicatore di Stile di Vita rispetto a tuttI i Sensori* (Sezione 5.3).

4.2 ITASS: Indicatore del Tempo di Attivazione di un Singolo Sensore

Il primo indicatore sviluppato, ITASS (*Indicatore del Tempo di Attivazione di un Singolo Sensore*), ha un compito molto semplice: considerare un singolo sensore e valutare una generica attività legata ad esso. In particolare l'indicatore potrà estrarre il tempo totale un sensore scelto abbia rilevato attività nelle ultime 24 ore.

4.2.1 Lo scenario

All'interno della struttura abitativa offerta dal progetto ARAS sono installati 20 sensori che tramite un Generatore di Eventi restituiscono istantaneamente al sistema informazioni sul loro stato. Ciascuno di questi sensori riporta informazioni riguardo la rilevazione o meno in un istante specifico di una caratteristica per la quale è stato sviluppato. Una prima informazione utile per il sistema BRIDGe nella valutazione della routine della persona, è quella di valutare il tempo di interazione dell'abitante con un sensore specifico, ossia calcolare quanta attività sia stata rilevata. Per esempio può risultare utile conoscere per quanto tempo (in un mese, in un giorno, o in un'ora) la persona monitorata è stata a letto, ha utilizzato un fornello, oppure ancora si è intrattenuta al telefono, ossia quegli aspetti che permettano di dedurre informazioni riguardo alle sue abitudini e a lanciare allarmi nel caso in cui queste si deteriorino.

Per questa elaborazione è risultato opportuno adottare una finestra di tempo che permettesse di vincolare temporalmente l'estrazione dal flusso di dati delle informazioni riguardo lo stile di vita e le abitudini dell'utente. Per questo indicatore è stata adoperata una finestra temporale lunga ventiquattro ore all'interno della quale calcolare il tempo totale di attività rilevata.

4.2.2 Come avviene il calcolo

L'estrazione dell'informazione scelta avviene attraverso l'impiego di una serie di interrogazioni EPL concatenate fra loro, cioè in cui il risultato di una può permettere l'applicazione di una successiva. Esper costruisce le interrogazioni perché queste siano *continue*, cioè che vengano applicate istantaneamente ogni volta che i dati (detti *eventi* prodotti dal generatore di eventi) che descrivono lo stato dei sensori, arrivano in input al sistema. Il calcolo dell'attività rilevata nelle ultime ventiquattro ore da parte di un sensore è molto semplice. Innanzitutto viene scelto il sensore di riferimento da impiegare per il quale effettuare l'analisi. Dopo di che dal flusso di dati in ingresso al sistema vengono estratti ed isolati gli eventi di attivazione rilevati dal sistema, ossia sequenze intiniterrotte in cui il sensore scelto restituisce stato '1'. Per ciascun nuova attivazione individuata vengono estratte informazioni riguardo la sua durata (espressa in secondi), e gli istanti di inizio e di fine.

Una volta trovate queste informazioni viene preso come riferimento l'istante in cui termina tale evento appena ricevuto, ad esso viene sottratto il valore di 86400 secondi (tanti quanti compongono una singola giornata). Il risultato ottenuto da questa sottrazione corrisponderà all'istante di termine dell'attivazione ventiquattro ore prima. I due valori rappresenteranno quindi gli estremi di una finestra temporale lunga 24 ore all'interno della quale valutare il tempo totale in cui il sensore rileva attività. Il calcolo viene svolto sommando le durate di tutti gli eventi che appartengono interamente o parzialmente alla finestra appena definita: il risultato restituito corrisponderà al tempo totale, espresso in secondi, in cui l'utente ha svolto una particolare azione rilevata dal singolo sensore scelto. In questo modo per ogni evento di attivazione rilevato dal sistema verrà calcolata la corrispondente finestra, così che sia possibile definire con il suo scivolamento nel tempo come e se varia il comportamento della persona monitorata.

4.2.3 Esempio ITASS: tempo trascorso dormendo nelle ultime 24 ore

Per poter costruire l'indicatore ITASS e simularne il funzionamento bisognava scegliere uno dei sensori presenti nella configurazione ARAS. A questo scopo si è optato per sensore del letto, così che questo restituisse informazioni riguardo il tempo trascorso dormendo nelle ultime 24 ore. La scelta è stata effettuata perché il risultato potesse

mettere in luce una delle attività più semplici della vita quotidiana (dormire), ma che potesse al contempo delineare un aspetto fondamentale dell'abitudinarietà di una persona monitorata. La Figura 4.3 mostra il diagramma di flusso con cui i dati provenienti dal sensore scelto (quello del letto) sono filtrati da sei interrogazioni sviluppate e, dopo una serie di passaggi dentro flussi interni creati, producono risultati che vengono inseriti nel flusso *OutputDailySleepingStream*.

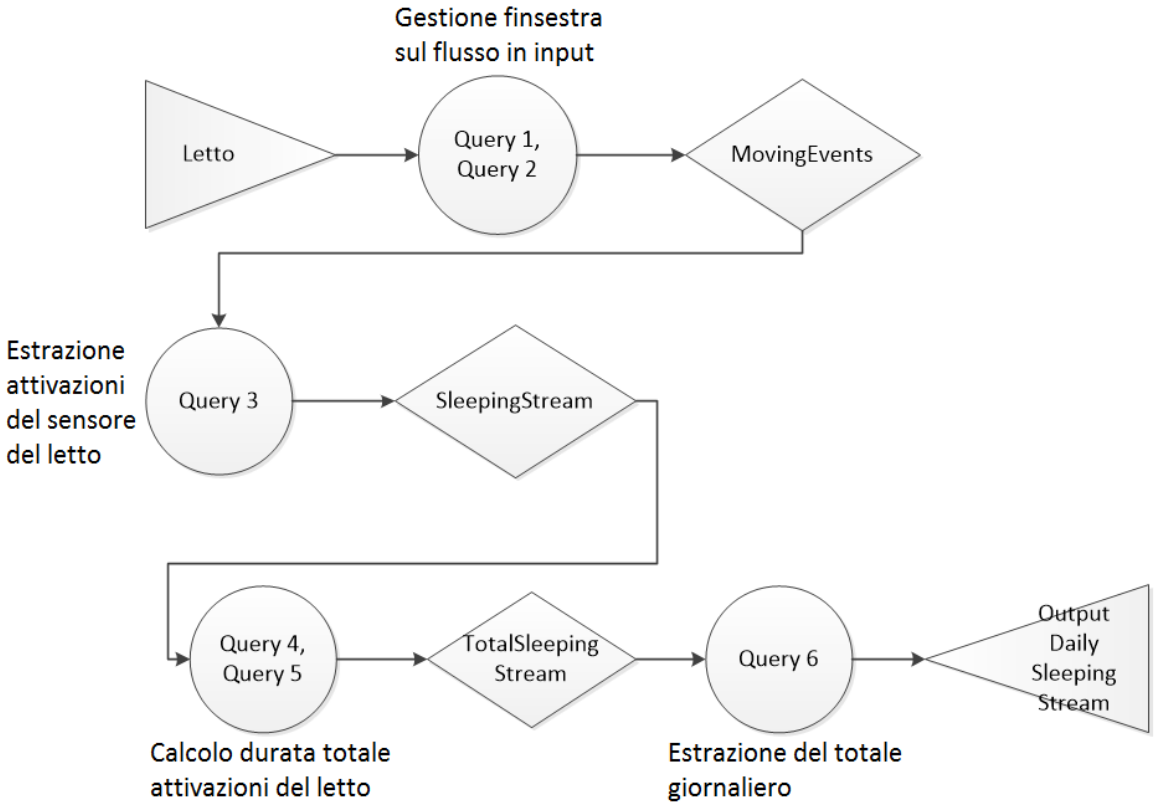

Figura 4.3: Diagramma di flusso per l'analisi del primo indicatori.

In questo diagramma vengono evidenziati tutti gli elementi adottati e costruiti per l'analisi: il *triangolo iniziale* rappresenta il sensore, quello del letto, i cui dati vengono analizzati dall'intero processo, i *cerchi* racchiudono una o più interrogazioni concatenate che hanno lo scopo di analizzare i dati più recenti (provenienti dall'elemento antecedente) e di restituirli in flussi costruiti internamente (rappresentati dai *rombi*) oppure in un flusso di output (detto *subscriber*, rappresentato dal *triangolo finale*) destinato a raccogliere le informazioni calcolate nella sequenza di passaggi. L'analisi delle operazioni svolte nel grafico sono introdotte nelle sezioni seguenti, in particolare la Sezione 4.2.3.1 descrive le

operazioni per identificazione ed estrazione dei segnali provenienti dal letto, che rappresentano le prime due righe del diagramma, mentre la Sezione 4.2.3.2 descrive la routine volta a calcolare il totale di attivazioni rilevate nelle ultime 24 ore e la restituzione del risultato opportuno nel flusso di output. Le interrogazioni sviluppate per ITASS sono raccolte nell'Appendice A, Sezione A1, nella quale esse sono elencate (con il numero di riferimento impiegato in questa sezione).

4.2.3.1 Estrazione attivazione del sensore scelto

In questa sezione viene presentato il procedimento di analisi rappresentato dalle prime due righe del diagramma di flusso della Figura 4.3.

Le interrogazioni Query 1 e Query 2 hanno il compito di estrarre dal flusso in ingresso relativo al sensore del letto sequenze di eventi che sono impiegate per popolare la finestra *MovingEvents*. Nello specifico, la Query 1 definisce la finestra *MovingEvents* sul flusso in ingresso *SensorStream* (quello che ospita i dati provenienti da tutti i sensori dell'abitazione). Questa finestra viene definita impiegando la sintassi *win:length(2)* a significare che essa possa essere popolata con due eventi alla volta singolaremnte estratti dal flusso per il quale è stata dichiarata. La Query 2 invece popola la finestra *MovingEvents* appena dichiarata inserendovi ogni nuovo evento riconosciuto all'interno del flusso *SensorStream* proveniente dal sensore del letto (*sensorID = 20*). Gli eventi riconosciuti dal flusso in ingresso ed inseriti nella finestra sono gli istanti in cui il sensore del letto cambia stato (ossia passa da '0' a '1' oppure viceversa). L'evento più recente viene inserito in *MovingEvents* al posto dell'evento più vecchio.

La finestra *MovingEvents* una volta popolata, viene analizzata dalla Query 3 ogni volta che un nuovo evento viene inserito. L'interrogazione verifica che i due eventi custoditi nella finestra siano un evento di attivazione seguito da uno di disattivazione. Se la condizione è verificata, la Query 3, dal confronto dei due eventi, estrae alcune informazioni necessarie per il singolo periodo di attivazione: istante di inizio, istante di fine, durata. Queste informazioni, necessarie per l'obbiettivo dell'indicatore di rilevare e sommare tutte le attivazioni del sensore nelle ultime 24 ore, viene inserita sotto forma di evento (con le proprietà estratte) in un flusso interno chiamato *SleepingStream.* Questo flusso raccoglie tutti i periodi di attivazione del sensore del letto man mano che vengono

rilevati dall'interrogazione Query 3 e sono impiegati nella sezione qui di seguito per il calcolo effettivo del totale di attivazione per le ultime 24 ore.

4.2.3.2 Calcolo tempo totale di attivazione del sensore nelle ultime 24 ore

In questa sezione viene presentata la procedura illustrata dall'ultima riga di operazioni del grafico della Figura 4.3 che permette di estrarre dal flusso interno *SleepingStream* i risultati da mandare al flusso in output finale.

Innanzitutto si comincia con le due interrogazioni Query 4 e Query 5 che effettuano il calcolo del tempo trascorso dormendo all'interno della finestra corrispondente alle ultime 24 ore generata dall'evento di attivazione trovato più di recente (l'ultimo appena inserito nel flusso *SleepingStream*). Tale finestra verrà applicata al flusso *SleepingStream* prodotto nella sezione precedente. Per questo calcolo vengono considerate due possibili casistiche:

a) vi è la possibilità che l'istante di inizio della finestra in considerazione cada all'interno di un evento di attivazione trovato in precedenza, quindi per il calcolo del tempo totale di riposo necessita di estrarre la frazione di evento iniziale appartenente alla finestra e sommarla alla durata degli eventi interamente appartenenti a tale finestra (caso Query 5);

b) quando invece l'istante di inizio della finestra non "taglia" alcun evento, il tempo totale di riposo è restituito semplicemente da tutti e soli gli eventi interamente appartenenti alla finestra (caso Query 4).

Le figure seguenti rappresentano in maniera grafica i due casi appena menzionati. In particolare la Figura 4.4 ritrae un esempio del *caso a*: il grafico rappresenta in nero i pattern di attivazione, in blu la finestra di 24 ore (o 86400 secondi) costruita dall'istante di fine dell'attivazione più recente trovata dal sistema. Per quanto riguarda le attivazioni, in tratteggiato sono rappresentati gli eventi che non appartengono alla finestra e quindi non vengono considerati nel computo del totale di attività, con il tratto continuo invece sono rappresentati quelli la cui durata viene sommata per trovare il valore finale. La particolarità di questo caso è che il primo evento rilevato non sarà considerato

interamente nell'analisi, ne verrà infatti utilizzato solo il frammento interno alla finestra (la parte rappresentata con il tratto continuo). A svolgere questa operazione è deputata l'interrogazione Query 5, la quale controlla che sia soddisfatta la condizione del caso in studio e, per ogni nuovo evento inserito nel flusso *SleepingStream*, esegue la somma del tempo totale (espresso in secondi) in cui il sensore è in stato '1'. Il risultato calcolato viene inserito nel flusso *TotalSleepingStream*, volto a contenere tutti i totali calcolati. Gli eventi inseriti, oltre a presentare il risultato della somma, presenteranno l'istante temporale pari a quello in cui termina la finestra di 24 ore impiegata, in modo che questo identifichi in maniera univoca ciascuna somma calcolata.

Analoga è la Figura 4.5, questa però ritrae il *caso b* in cui non vi sia un evento iniziale "tagliato" dall'istate di inizio della finestra e nel computo saranno considerati solo gli eventi interamente interni alla finestra. In questo modo risulta più chiara la differenza fra i due casi. Per svolgere questa situazione viene adoperata l'interrogazione Query 4, la quale agisce come quella descritta per il caso precedente: somma le durate di tutte le attivazioni interne alla finestra ed inserisce il risultato (insieme all'istante in cui essa termina) in un nuovo evento dentro al flusso *TotalSleepingStream.*

Figura 4.4: Rappresentazione grafica di un esempio di *caso a*.

Figura 4.5: Rappresentazione grafica di un esempio di *caso b*.

Al termine di questa trattazione vi è l'ultima operazione: restituire ad un flusso destinato a raccogliere gli output i valori ricercati fra quelli inseriti nel flusso

TotalSleepingStream. Questa parte è quella più a discrezione dell'utente, infatti in base alle proprie esigenza di informazioni ciascuno può estrarre quelle che più gli interessano. Nel lavoro presente si è ricercato il valore calcolato al mezzogiorno di ciascun giorno di analisi, in modo che l'informazione coprisse senza interruzione la notte fino al mezzogiorno del giorno precedente. In questo modo la Query 6 estrae il valore di interesse di tempo totale di attivazione calcolata nelle ultime 24 ore ed inserisce tale risultato nel flusso di output *OutputDailtSleepingStream* al quale è legato un listener che riporta il risultato anche in un file testuale di destinazione impiegabile per le valutazioni successive sulle abitudini della persona monitorata.

4.2.3.3 Il risultato

In questo modo, utilizzando le sei interrogazioni appena introdotte, l'indicatore ITASS è in grado di calcolare correttamente il valore totale di attività rilevata nelle ultime 24 ore. Il calcolo è sempre aggiornato perché effettuato ogni volta che un nuovo evento di attivazione viene rilevato. Il risultato giornaliero estratto e restituito in un file testuale può essere impiegato dal progetto BRIDGe per una prima valutazione riguardo la vita dell'abitante della casa intelligente.

Per quanto concerne i risultati stampati a video, tramite listener associati alle Query 3, 4 e 5, la Figura 4.6 mostra la sequenza di attivazioni del sensore letto trovate ed inserite in *SleepingEvent*, ciascuna delle quali è seguita dal proprio ricalcolo del tempo totale di riposo salvato in *TotalSleepingEvent*.

```
Event received: SleepingEvent{duration=58.0, initActivation=87405.0, finActivation=87462.0}
Event received: TotalSleepingEvent{totalDuration=12754.0}
Event received: SleepingEvent{duration=1277.0, initActivation=87634.0, finActivation=88910.0}
Event received: TotalSleepingEvent{totalDuration=13405.0}
Event received: SleepingEvent{duration=133.0, initActivation=90666.0, finActivation=90798.0}
Event received: TotalSleepingEvent{totalDuration=12598.0}
```

Figura 4.6: Risultati di ITASS per il calcolo delle ore di sonno nelle ultime 24 ore.

Il risultato inserito nel file testuale, mostrato nella figura 4.7 invece è una sequenza di somme di attivazioni estratte, per il sensore del letto, una per ogni giorno di analisi a mezzogiorno.

```
OutputDailySleepingEvent { totalDuration= 35449.0 , dayAnalyzed= 1.0 }
OutputDailySleepingEvent { totalDuration= 35148.0 , dayAnalyzed= 2.0 }
OutputDailySleepingEvent { totalDuration= 27719.0 , dayAnalyzed= 3.0 }
OutputDailySleepingEvent { totalDuration= 28069.0 , dayAnalyzed= 4.0 }
OutputDailySleepingEvent { totalDuration= 34429.0 , dayAnalyzed= 5.0 }
OutputDailySleepingEvent { totalDuration= 35208.0 , dayAnalyzed= 6.0 }
OutputDailySleepingEvent { totalDuration= 32209.0 , dayAnalyzed= 7.0 }
OutputDailySleepingEvent { totalDuration= 33408.0 , dayAnalyzed= 8.0 }
OutputDailySleepingEvent { totalDuration= 33108.0 , dayAnalyzed= 9.0 }
```

Figura 4.7: Esempio di risultato testuale di ITASS.

Tuttavia il risultato che è stato ottenuto però non può essere immediatamente rimandato all'attività di dormire perché non è possibile identificare le attività effettivamente svolte dall'abitante mentre il singolo sensore rileva una pressione sul letto (per esempio: fare il cambio dell'armadio, preparare una valigia, rilassarsi senza dormire). Pertanto se si considera il risultato come espressione di questa informazione, bisogna essere consapevoli che talvolta (ma non sempre) questo sia approssimato.

Il lavoro che questo indicatore svolge può essere combinato con altri indicatori per ottenere informazioni più articolate. La sua semplicità ed essenzialità infatti permette di integrarlo facilmente all'interno di strutture più complesse. ITASS infatti può essere impiegato per la valutazione di informazioni diverse da quelle ottenute nell'esempio appena illustrato e con semplici modifiche può essere applicato allo scopo di estrarre, per esempio, informazioni riguardo altri aspetti della vita della persona monitorata.

4.3 ISCS: Indicatore per Stimare il Consumo di Servizi domestici

Il secondo indicatore sviluppato, ISCS (*Indicatore per Stimare il Consumo di Servizi domestici*), si occupa di cercare una modalità con cui poter stimare i consumi casalinghi. Una vita abitudinaria, come potrebbe essere quella di una persona anziana, si dovrebbe riflettere nei consumi di elettricità, gas ed acqua. Pertanto l'analisi di queste informazioni possiede le potenzialità di permettere il riconoscimento di situazioni di anomalia sia nel breve che nel lungo termine.

4.3.1 Lo scenario

Lo scenario nel quale si inserisce lo sviluppo di questo indicatore per il progetto BRIDGe rimane sempre quello del contesto abitativo utilizzato dal progetto ARAS. Rispetto all'analisi precedente in cui viene valutata l'attività svolta relativa ad un singolo sensore, per la rilevazione dei consumi domestici risulterà necessario impiegare dei gruppi di sensori. Se per esempio si vuole calcolare il consumo di elettricità, si prenderanno in considerazione i sensori installati atti alla rilevazione di attività da parte delle apparecchiature elettriche, per il consumo di gas invece saranno considerati i sensori relativi alle strutture che facciano uso di tale servizio.

Nella geografia di sensori offerta da ARAS però, non sono presenti sensori che permettano di tracciare l'utilizzo di questi servizi. Pertanto l'unica grandezza che la struttura ARAS permette di tracciare in maniera appropriata, riguarda il consumo domestico di acqua: esso infatti potrà essere espresso tramite l'impiego di alcuni sensori installati dove si trovano sanitari, e apparecchi per l'igiene. Nell'appartamento *HouseA* il consumo di acqua potrà essere stimato tramite l'impiego di tre sensori:

- *sensorID = 17*: il sensore di distanza applicato al lavandino del bagno che attiva il funzionamento dell'acqua quando rileva la presenza di una persona vicina al rubinetto. Esso restituisce stato '1' ogni volta che rileva una tale presenza.
- *sensorID = 18:* il sensore di distanza applicato alla tavoletta del wc che rileva attività (e quindi restituisce stato '1') ogni qualvolta una persona alza la tavoletta per usufruire del sanitario.
- *sensorID = 14*: il sensore di contatto posto sull'antina della doccia, questo restituisce stato '1' ogni volta che l'antina si chiude ed entra in contatto con una seconda componente del sensore installata sull'infisso.

Eventuali altri componenti atti a rilevare consumi casalinghi relativi all'utilizzo di acqua (come ad esempio il lavandino della cucina) non sono previsti nella presente configurazione dell'appartamento, ma l'analisi è aperta per un'estensione volta ad includere ulteriori contributi provenienti da altri sensori.

4.3.2 Come stimare il consumo del singolo elemento

Uno dei passaggi fondamentali per la stima del consumo di una quantità è assumere decisioni riguardo le modalità con cui i singoli sensori presi in considerazione permettano di calcolare il proprio contributo.

I tre elementi indicati nella sezione precedente permettono di definire due approcci validi per il calcolo dei consumi. In questo caso tali approcci sono impiegati per rilevare i consumi dell'acqua, ma sono ugualmente adottabili per altri servizi.

1) **Durata attività * Stima consumo**

   Una prima modalità per la valutazione del consumo di un sensore è quella di estrarre la durata (espressa in secondi) per singola attività rilevata dal flusso di dati in ingresso al sistema. In questo modo viene quantificata la durata temporale (espressa in secondi) del singolo evento di consumo da parte del sensore in considerazione. Ciascun valore ottenuto viene poi moltiplicato per la portata caratteristica del singolo elemento/sanitario. Così facendo, verrà restituita la quantità di risorsa consumata in una singola attività rilevata dal singolo sensore.

   Nel caso del rubinetto del lavandino del bagno (*sensorID = 17*) il consumo dell'acqua viene rilevato direttamente moltiplicando la durata della singola attività per una portata pari a 0.08 litri/secondo [**23**].

   Nel caso della doccia (*sensorID = 14*) il consumo di acqua viene calcolato moltiplicando la durata della singola attività rilevata per una portata pari a 0.133 litri/secondo [**23**].

2) **Singolo evento* Stima consumo**

   La seconda tecnica impiegabile è riferita a quegli elementi che generano un consumo che sia fisso, indipendentemente dalla durata dell'evento, oppure di carattere impulsivo. L'attivazione del sensore infatti, pur potendo estendersi per più istanti, genera un consumo prestabilito che avviene in un solo momento. Pertanto alla rilevazione di un'attività per il sensore in questione, il sistema riconosce che ad essa corrisponde un consumo fissato in precedenza. In questo modo viene restituita la quantità di risorsa consumata ogni volta che il sensore di riferimento riconosce un'attività di qualsiasi durata.

Questo approccio risulta utile per la valutazione, per esempio, del consumo di acqua effettuata dallo scarico del bagno (*sensorID = 18*): ogni qualvolta che viene riconosciuta un'attività di tale elemento, il sistema introduce un consumo di 9 litri di acqua, pari alla portata media di uno sciacquone [**24**].

In questo modo sono introdotti due principi applicabili in vari modelli ed in varie strutture per il calcolo del consumo da parte di un singolo sensore.

4.3.3 Come avviene il calcolo

Fino a questo momento è stato illustrato il contesto in cui può essere introdotto un indicatore atto a calcolare il consumo di una risorsa in un'intera abitazione e le modalità con cui impiegare i singoli sensori installati in una struttura abitativa intelligente per tracciare di tali consumi. Ora non resta che introdurre una proposta volta a quantificare concretamente il consumo globale di una particolare risorsa in un'intera struttura abitativa considerando tutti i sensori che permettono di dare una stima dell'utilizzo di tale servizio.

Il principio applicato è simile a quello introdotto per la costruzione di ITASS sviluppato nella Sezione 4.2.2. Dagli eventi creati dal Generatore di Eventi leggendo il dataset e passati al sistema tramite il flusso *SensorStream*, vengono individuate, attraverso interrogazioni EPL, le attivazioni dei sensori presi in considerazione per l'analisi. Per ciascun sensore viene calcolato il consumo secondo le modalità esplicitate nella sezione precedente.

Perché l'informazione estratta risulti impiegabile in un processo successivo di analisi è necessario individuare una finestra temporale per la valutazione del consumo. A questo scopo viene utilizzata, proprio come fatto per ITASS (l'indicatore precedente), una finestra temporale di 86400 secondi (ossia 24 ore) ridefinita ad ogni nuova attivazione trovata dal sistema e che calcoli ogni volta il valore del consumo totale della risorsa scelta per questo lasso di tempo; tale calcolo avverrà impiegando il contributo da parte di ciascuno dei sensori presi in considerazione.

Rispetto alla circostanza precedente però possono occorrere delle situazioni nuove dovute dall'impiego contemporaneo di più sensori, esse saranno illustrate a breve nell'esempio introdotto nella sezione successiva. Il risultato che il calcolo restituirà sarà

quindi la somma dei contributi di tutti i sensori impiegati nell'analisi nelle ultime 24 ore antecedenti l'evento più recente rilevato da uno qualsiasi dei sensori scelti.

4.3.4 Esempio IRAC: consumo totale di acqua nelle ultime 24 ore

L'esempio di calcolo di consumo che viene illustrato nel lavoro presente si riconduce all'unica situazione possibile dalla configurazione di sensori utilizzata dal progetto ARAS: la stima del consumo domestico di acqua impiegando i sensori di lavandino, doccia e scarico installati nel bagno dell'appartamento. Questa valutazione effettuata impiegando una finestra di 24 ore, permetterà di delineare i tratti abitudinari dei consumi nell'arco della singola giornata, e quindi successivamente di valutare eventuali deviazioni da interpretare come allarmi per lo stato di salute della persona monitorata. La Figura 4.8 illustra la struttura dell'analisi tramite un diagramma di flusso in cui viene mostrato il percorso cui sono soggetti i dati provenienti dai sensori allo scopo di estrarre la stima giornaliera del consumo di acqua domestico (di tutta la casa).

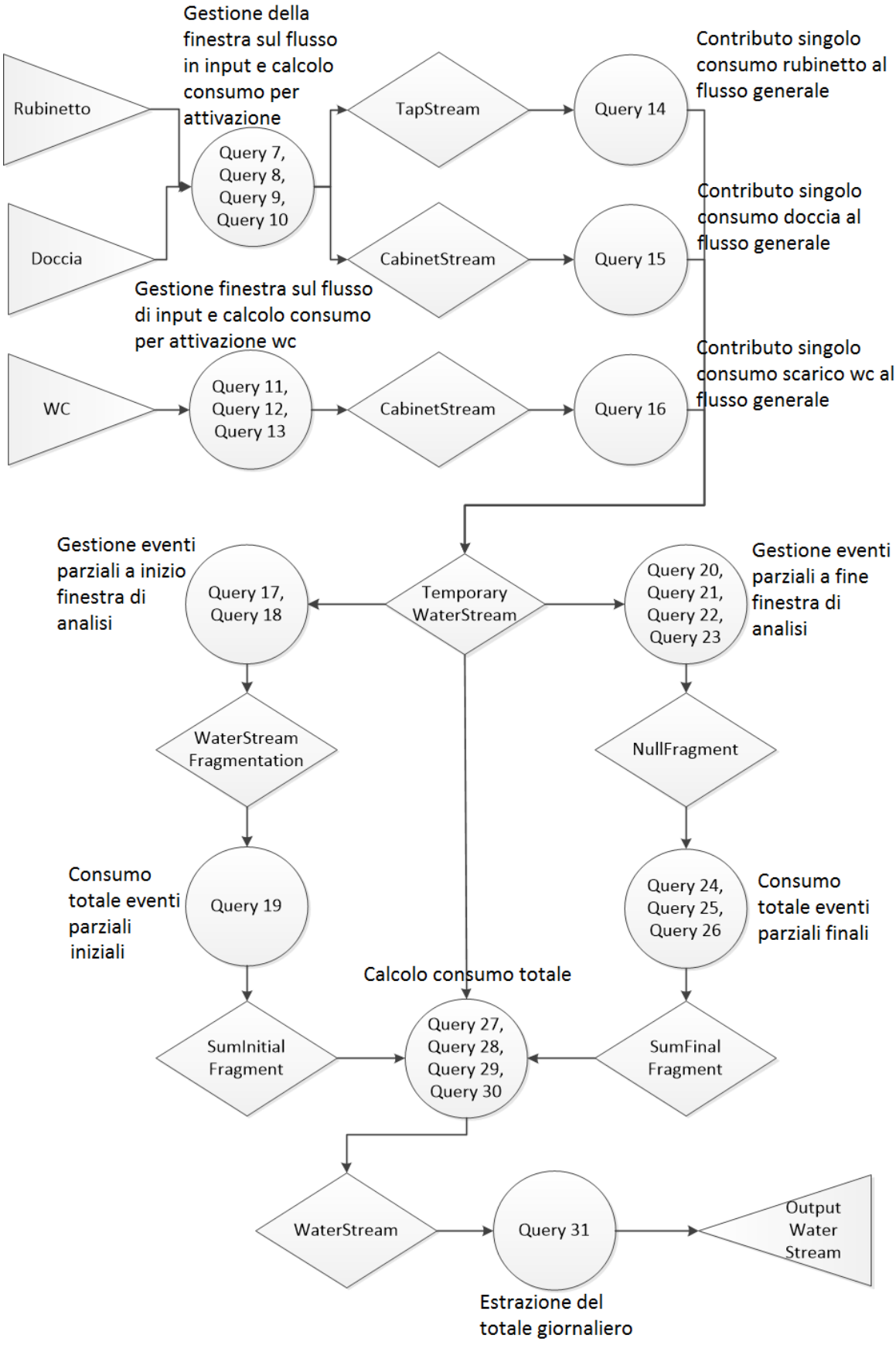

Figura 4.8: Diagramma di flusso di ISCS per il consumo di acqua in tutta casa.

Nel diagramma i triangoli iniziali rappresentano le sorgenti, quindi i tre sensori di rubinetto, doccia e wc considerati, i cerchi sono le interrogazioni che lavorano su di essi e li inseriscono in flussi o finestre interne rappresentati dai rombi, ed infine l'ultimo triangolo è il *subscriber* che riceve l'output. L'analisi si articola in vari passaggi, sottostrutture interne al grafico:

1. L'identificazione dei fenomeni di attivazione per i tre sensori impiegati per popolare i flussi *TapStream*, *CabinetStream* e *WcStream*. Come introdotto nella Sezione 4.2.3.1 i sensori di doccia e rubinetto si comporteranno in maniera analoga (analisi nella Sezione 4.3.4.1) mentre quello dello scarico avviene separatamente (nella Sezione 4.3.4.2).
2. Per ciascuno dei tre flussi *TapStream*, *CabinetStream* e *WcStream* (cioè per ciascuno dei tre sanitari) può essere calcolato il proprio consumo in un intervallo di tempo stabilito dall'utente (un giorno, una settimana…). Questo passaggio è facoltativo, e si presenta come un'applicazione dell'indicatore precedente, non è stato rappresentato nel diagramma (poiché sarebbe la Figura 4.3 ripetuta per ciascun sanitario). Questo passaggio è presentato nella Sezione 4.3.4.3.
3. Successivamente i contributi dei flussi *TapStream*, *CabinetStream* e *WcStream* vengono raccolti nel flusso *TemporaryWaterStream* che serve per calcolare la stima totale del consumo di acqua (Sezione 4.3.4.4).
4. Per effettuare il calcolo vero e proprio del totale di acqua consumata nell'abitazione in un arco temporale che per il lavoro presente è fissato a 24 ore (Sezione 4.3.4.6), vengono prima studiate tutte le possibili situazioni simili a quelle rappresentate nelle Figure 4.4 e 4.5 nelle Sezioni 4.3.4.5.1 e 4.3.4.5.2. Alla fine di questo passaggio il risultato vene inserito nel flusso di output che lo scrive, tramite un apposito listener, in un file testuale impiegabile per le analisi successive.

Le interrogazioni sviluppate per ISCS sono raccolte nell'Appendice A, Sezione A2 dove sono elencate.

4.3.4.1 Estrazione di attivazioni e consumi per sensori di rubinetto e doccia

L'inizio dell'analisi per i sensori del rubinetto e della doccia presenta una struttura molto simile a quella introdotta nella Sezione 4.2.3.1 per la valutazione dell'attività relativa al singolo sensore del letto. Allo stesso modo con cui in precedenza venivano rilevati dal flusso *SensorStream* tramite le Query 1, 2 e 3 le singole attivazioni del sensore del letto, ora vengono individuate quelle del sensore del rubinetto del lavandino e della doccia. Come prima infatti viene inizalmente creata una finestra (*MovingWaterEvents*) sul flusso *SensorStream* destinata ad ospitare coppie di eventi successivi tramite le interrogazioni Query 7 e Query 8. Poi, ogni volta che viene inserito nella finestra un nuovo evento, le Query 9 e 10 identificano se si è in presenza di un'attivazione seguita da una disattivazione, in caso positivo, vengono estratte le informazioni di durata, consumo ed estremi temporali dell'evento di attivazione e sono inserite come proprietà di un singolo evento in *TapStream* o *CabinetStream* a seconda del sensore considerato.

La motivazione che spinge ad impiegare un ragionamento analogo a quello precedente, è perché in ambo le situazione l'informazione necessaria per il calcolo del risultato finale è la durata del singolo evento di attivazione. Per il rubinetto e la doccia infatti il consumo sarà dato dal prodotto fra durata e stima del consumo come illustrato nel *caso a* della Sezione 4.3.4.

4.3.4.2 Estrazione e gestione attivazioni e consumi per sensore dello scarico

Per quanto concerne invece il calcolo del consumo dello scarico del bagno è stato illustrato nella Sezione 4.3.2 come l'approccio impiegato sia differente da quello utilizzato per i due sensori precedenti. Questa differenza è illustrata anche nella Figura 4.8 con il percorso a sé dell'analisi per questo sensore. Nello specifico, supponendo l'impiego di uno scarico con cassetta, ogni volta che il sensore rileva un'attività questa implica un utilizzo dello scarico, dopo di che sarà necessario un breve periodo di tempo atto a ricaricare la cassetta dell'acqua.

Per questo motivo, oltre al fatto che il sensore impiegato valuti la distanza e pertanto non sia un diretto rilevatore dell'utilizzo del wc, si è strutturata l'analisi in questo modo: dapprima con la Query 11 viene riconosciuto un evento di attivazione proveniente dal sensore posto sul wc (*sensorID = 18*), dopo di che si verifica che entro 10 secondi dall'istante di inizio non sopraggiunga nessun nuovo evento di attivazione tramite la

Query 13; se questa condizione è verificata viene considerato il contributo dello sciacquone, altrimenti si aspetta la prima attivazione che non sia seguita entro 10 secondi da un'altra. In questo modo si risolve il problema del ricaricamento della cassetta nel momento in cui il sensore rileva più attivazioni distinte ma estremamente ravvicinate nel tempo (con una distanza temporale inferiore ai 10 secondi necessari per riempire la cassetta).

I risultati validi a descrivere l'utilizzo del wc appena estratto, sono quindi inseriti all'interno del flusso di destinazione *WcStream.*

4.3.4.3 Calcolo del consumo per singolo apparecchio

Con gli strumenti che abbiamo individuato nelle due sezioni precedenti sono stati estratti dal flusso *SensorStream* le informazioni inerenti le stime di consumo di acqua per ogni possibile attivazione dei tre sensori presi in considerazione singolarmente: lavandino, doccia e scarico. Tali risultati sono stati inseriti in flussi differenti (*TapStream*, *CabinetStream* e *WcStream*) per ciascun sensore, di modo da mantenere isolati i valori estratti. L'approccio e l'idea adottati per stimare il consumo per singolo seonsore è molto simile a quella offerta dall'indicatore della Sezione 4.2: vengono infatti sommati tutti i contributi delle ultime 24 ore per singolo sensore ogni volta che questo rileva un nuovo evento di attivazione.

L'unica differenza, pertanto, rispetto a ITASS, consiste nell'impiego del consumo di acqua come misura utilizzata per l'analisi al posto della durata della singola attivazione del sensore. Di conseguenza si definisce il calcolo utilizzato in questo contesto come un esempio di applicazione dell'indicatore ITASS introdotto nella Sezione 4.2.

Come si è visto in precedenza, l'analisi dei consumi di rubinetto e doccia si comportano analogamente, pertanto per il calcolo del totale di acqua consumato, per ciascuno di essi si hanno due interrogazioni simili alle Query 4 e Query 5 le quali coprono i due casi che erano stati presentati nelle Figure 4.4 e 4.5. Per il rubinetto le Query 32 e 33 eseguono il conto ed inseriscono il risultato ottenuto nel flusso *TotalTapStream*, per la doccia invece le Query 34 e 35 che inseriscono i risultati in *TotalCabinetStream*.

Per quanto concerne la ricerca del consumo totale per lo scarico del wc nelle ultime 24 ore, si è visto come l'approccio sia completamente diverso da quello degli altri due dispositivi presi in considerazione. Lo sciacquone viene infatti considerato come un

evento istantaneo che si verifica solamente quando l'attivazione trovata dal sensore non sia immediatamente seguita da un'alta, questo perché vi è la necessità di riempire completamente la vaschetta dell'acqua, parte fondamentale per un impianto di scarico. La definizione di questo carattere "impulsivo" del consumo di acqua rende molto più semplice la valutazione del consumato totale perché nel flusso *WcStream* sono contenuti i soli istanti in cui viene attribuito l'utilizzo dello scarico. In questo modo il consumo nelle 24 ore precedenti l'ultimo evento rilevato sarà dato dal contributo di tutti i soli istanti registrati interni alla finestra. L'interrogazione Query 36 svolgerà questo calcolo ed inserisce i risultati in *TotalWcStream*.

In questo modo è possibile avere un elemento in più per l'analisi delle abitudini dell'abitante della casa: valutare il consumo nelle ultime 24 ore per ciascun sensore preso singolarmente.

4.3.4.4 Gestione contributo contemporaneo di più sensori

Riprendendo il processo introdotto con la Figura 4.8, alla Sezione 4.3.4.2 eravamo giunti all'inserimento degli eventi di attivazione nei flussi *TapStream*, *CabinetStream* e *WcStream.* In questa sezione viene presentata la modalità per poter ottenere il consumo totale di acqua nella casa nelle ultime 24 ore (dato dai tre sensori allo stesso tempo): impiegare un flusso nel quale far confluire i contributi di tutti i sensori considerati. A questo scopo viene introdotto il flusso *TemporaryWaterStream*, in cui sono inseriti gli eventi relativi ai consumi calcolati per i tre sensori non appena questi vengono rilevati ed aggiunti negli appositi flussi di destinazione *TapStream*, *CabinetStream* e *WcStream*. Le interrogazioni 14, 15 e 16 hanno lo scopo di effettuare questa operazione per ciascuno dei tre flussi.

In questo modo il sistema raccoglie ogni nuovo evento di attivazione trovato per uno qualsiasi dei tre sensori in considerazione in un unico flusso *TemporaryWaterStream* che permette il calcolo totale del consumo di acqua impiegando il contributo di tutti i sensori considerati.

4.3.4.5 Gestione della finestra temporale

Il diagramma in Figura 4.8 mostra come successivamente al passaggio appena svolto vi sia la possibilità di seguire due percorsi intermedi prima giungere all'effettivo calcolo della somma del consumo totale nelle ultime 24 ore calcolate partendo dall'ultimo evento inserito nel flusso *TemporaryWaterStream*. Si noti che il percorso centrale viene escluso momentaneamente da questi ragionamenti.

Come per ITASS (Sezione 4.2) l'impiego di una finestra temporale può generare delle situazioni eccezionali (rappresentate nelle Figure 4.3 e 4.4) che non possono essere tralasciate nel momento in cui sia calcolata la stima totale dei consumi all'interno di una tale finestra. Poiché rispetto al caso precedente per ISCS vengono presi in considerazione contemporaneamente tre sensori, le situazioni che emergono sono in parte differenti rispetto a quelle trovate nella Sezione 4.2.3.2.

In particolare ci sono due situazioni da risolvere:

- La possibilità che vi siano eventi che iniziano prima dell'istante di inizio della finestra e terminano dopo il suo inizio (caso molto simile a quello riportato nella Figura 4.3 nella Sezione 4.2.3.2), che verrà studiato nella Sezione 4.3.4.5.1. Questa analisi è rappresentata nel diagramma di flusso dal percorso sinistro.
- La possibilità che nell'istante di riferimento in cui viene considerato l'estremo terminante della finestra vi sia per uno o più sensori la rilevazione di eventi di attivazione in corso e non ancora terminata. Il caso verrà affrontato nella Sezione 4.3.4.5.2 ed è rappresentato dal percorso destro nella Figura 4.8

4.3.4.5.1 Analisi dell'istante di inizio della finestra

Ogni volta che un nuovo evento di attivazione viene rilevato da uno dei tre sensori del bagno e, dopo averne estratto tutte le informazioni, viene inserito nel flusso globale *TemporaryWaterStream*, l'istante di termine di questo evento viene preso come riferimento per l'analisi volta a calcolare il consumo totale di acqua in casa nelle ultime 24 ore. A questo scopo viene costruita una finestra lunga 86400 secondi (24 ore) il cui estremo finale sarà l'istante finale preso come riferimento e l'istante iniziale sarà il valore di tale istante finale sottratto di 86400 secondi (ossia l'istante finale 24 ore prima). La

costruzione della finestra in questo modo potrà portare all'individuazione delle situazioni mostrate nelle figure seguenti.

Figura 4.9: Esempio di situazione in cui l'istante iniziale della finestra non genera situazioni eccezionali.

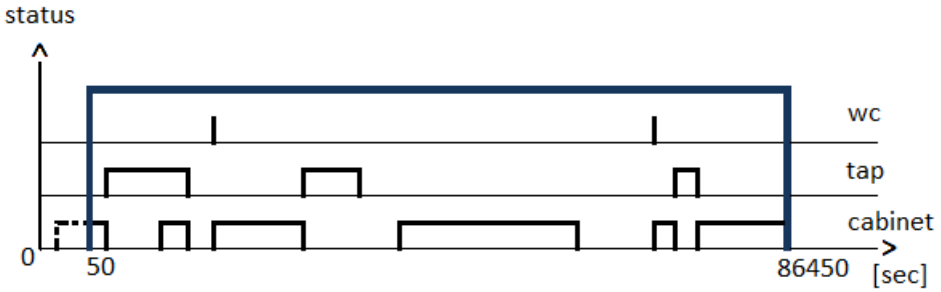

Figura 4.10: Esempio di situazione in cui l'istante iniziale della finestra taglia un evento.

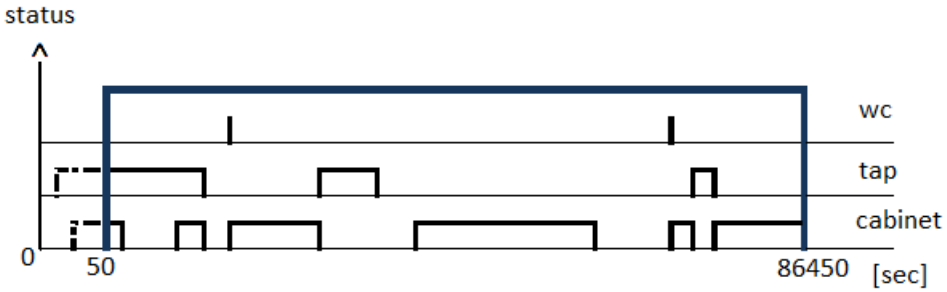

Figura 4.11: Esempio di situazione in cui l'istante iniziale della finestra taglia ben due eventi.

Nella Figura 4.9 l'istante iniziale della finestra (rappresentata in blu) non interseca alcun evento precedente all'ultimo osservato e pertanto l'analisi non incontrerà situazioni eccezionali. Queste si presentano invece nelle Figure 4.10 e 4.11 in cui l'istante iniziale interseca un evento per il sensore del rubinetto o della doccia (Figura 4.10) oppure addirittura per entrambi (Figura 4.11). In questi due casi è necessario sviluppare un

meccanismo che permetta al sistema di poter tenere traccia dei possibili "frammenti" di eventi (generati all'istante di inizio della finestra) appartenenti alla finestra. In questa analisi gli eventi rilevati dallo scarico non danno problemi poiché, essendo eventi di carattere istantaneo, o appartengono o non appartengono alla finestra, senza ulteriori possibilità.

Per gestire quindi i due casi delle Figure 4.10 e 4.11 sono adottate due interrogazioni Query 17 e Query 18, le quali vanno a verificare se esiste un evento di attivazione precedente di rubinetto (Query 17) e doccia (Query 18) i quali siano tagliati dall'istante di inizio della finestra di 24 ore costruita dall'evento più recente inserto in *TemporaryWaterStream.* Se pertanto esiste un evento che soddisfa la condizione, viene estratto il frammento interno alla finestra e viene inserito nel flusso *WaterStreamFragmentation*, destinato a raccogliere i frammenti trovati (per i quali oltre che al consumo corrispondente, viene estratto l'istante di fine della finestra, che permette di identificarli).

Il passaggio successivo a questa ricerca e raccolta di "frammenti" all'interno del flusso *WaterStreamFragmentation* è quello di sommare quelli trovati con lo scopo di impiegare il loro contributo poi per il calcolo completo del totale di acqua utilizzata nelle ultime 24 ore. La Query 19 esegue la somma degli eventuali frammenti trovati con le due interrogazioni precedenti. Il risultato viene inserito nel flusso *SumInitialFragments.*

Il flusso in output raccoglie quindi tutte le somme di frammenti per instante finale della finestra, quindi sarà facilmente accessibile nel momento in cui si debba calcolare il totale del consumo casalingo. Questo risultato permette di concludere l'analisi delle situazioni possibili dovute all'istante di inizio della finestra.

4.3.4.5.2 Analisi dell'istante di fine della finestra

Se per lo studio effettuato nella sezione precedente vi era qualche somiglianza con quanto già visto per il primo indicatore sviluppato, in questa sezione viene affrontata una situazione completamente nuova causata dall'impiego simultaneo di più sensori (questa analisi nel diagramma di Figura 4.8 è rappresentata dal braccio destro): quando viene considerato l'ultimo evento inserito nel flusso *TemporaryWaterStream* e viene costruita la finestra di 24 ore da impiegare per il calcolo del consumo totale, l'istante di fine di tale finestra potrebbe infatti coincidere con un istante in cui un altro sensore stia rilevando

un'attivazione non ancora conclusa. In questo modo per una corretta analisi del consumo di acqua per quella specifica finestra andranno estratti i frammenti di eventi appena citati.

I casi che possono verificarsi sono diversi e le Figure seguenti li rappresentano.

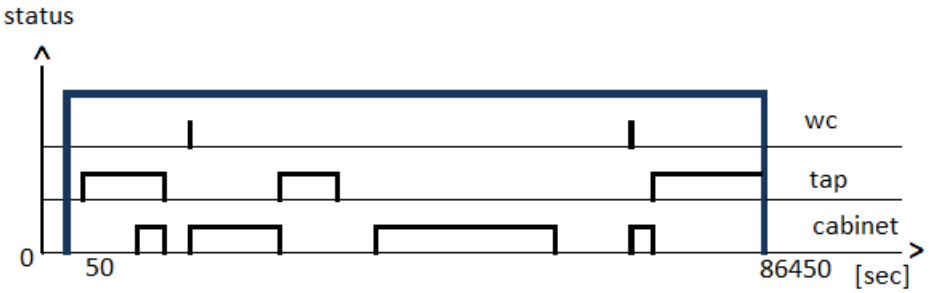

Figura 4.12: Esempio di situazione in cui l'istante finale delle finestra non genera alcuna situazione eccezionale.

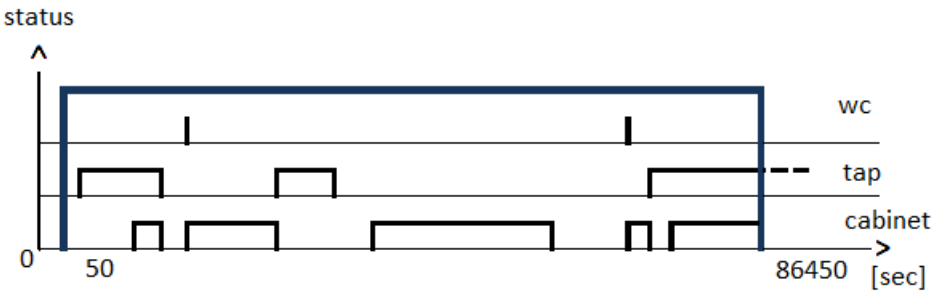

Figura 4.13: Esempio di situazione in cui l'istante finale della finestra taglia un evento non ancora monitorato del tutto.

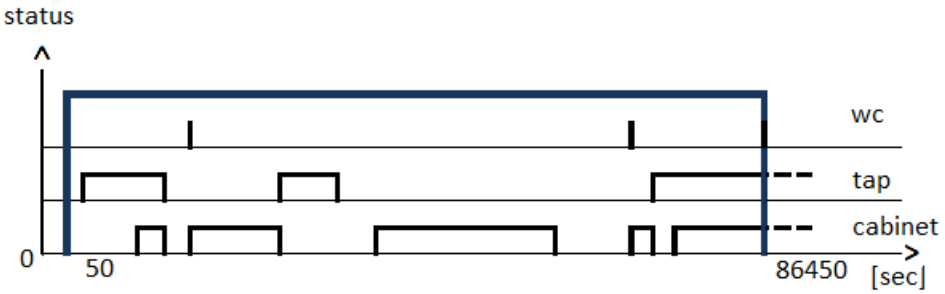

Figura 4.14: Esempio di situazione in cui l'istante finale della finestra taglia due eventi non ancora monitorati del tutto. Questo può capitare solo se la finestra viene generata da un evento del wc.

La Figura 4.12 rappresenta il caso semplice in cui nell'istante finale della finestra non vi sia alcun sensore che stia inviando stato '1'.

La Figura 4.13 mostra invece il caso in cui all'istante finale della finestra corrisponde almeno un evento di attivazione che stia venendo riconosciuto proveniente da un altro

sensore (simboleggiato dal tratteggio). Non è possibile infatti che lo stesso sensore il cui ultimo evento abbia permesso di costruire la finestra per l'analisi stia rilevando in quello stesso istante un'altra attività.

La Figura 4.14 mostra il caso estremo in cui la situazione appena descritta (rappresentata nella Figura 4.13) sia verificata per due sensori allo stesso tempo. Anche in questo caso occorrere riflettere che lo scarico, per il suo comportamento impulsivo, non darà adito a problemi di frammentazione perché i suoi eventi, come per il caso precedente, saranno o totalmente interni o totalmente esterni alla finestra.

Per trattare in maniera corretta queste situazioni viene proposto un percorso di analisi avviato dalle interrogazioni Query 20 e 21 le quali hanno lo scopo di estrarre dal flusso di eventi in ingresso gli eventi di disattivazione per il rubinetto (la 20) e per la doccia (la 21). La prima interrogazione inserirà i risultati nel flusso *NullTapStream* e la seconda in *NullCabinetStream* i quali raccolgono al loro interno tutti gli interi periodi in cui il sensore non rileva attivazioni.

Successivamente le interrogazioni Query 22 e 23 verificano se effettivamente è in corso la rilevazione di un'attivazione nell'istante in cui è stata costruita la finestra. La Query 22 per quanto riguarda il rubinetto e la 23 riguardo la doccia. Se quindi viene verificato questo aspetto, si estraggono i frammenti di attivazione monitorati fino all'istante in cui è stata costruita la finestra di analisi e sono inseriti rispettivamente nei flussi *NullTapFragment* e *NullCabinetFragment*.

Una volta ottenuti nei flussi *NullTapFragment* e *NullCabineFragment* i possibili frammenti presenti in coincidenza della fine della finestra, non resta che inserire nel flusso *SumFinalFragment* il contributo del/dei frammento/i iniziale/i per la finestra di 24 ore in analisi. In questa situazione sono possibili tre casi, ciascuno dei quali gestito da un'interrogazione apposita:

- L'interrogazione Query 24 gestisce il caso in cui vi sia solo un frammento proveniente dal sensore del rubinetto. L'interrogazione inserisce tale frammento nel flusso *SumFinalFragment* perché sia impiegato successivamente per il computo totale del consumo di acqua nell'abitazione.
- L'interrogazione Query 25 gestisce il caso in cui vi sia solo un frammento proveniente, questa volta, dal sensore della doccia.

- L'interrogazione Query 26 gestisce il caso in cui invece sono presenti un frammento per entrambi i sensori in analisi, quindi questi vengono sommati fra di loro ed inseriti nel flusso *SumFinalFragment.*

In questo modo la procedura ha terminato l'analisi delle possibili situazioni eccezionali agli estremi della finestra di 24 ore costruita con l'ultimo evento inserito in *TemporaryWaterStream,* pertanto non resta che effettuare il calcolo vero e proprio del consumo di acqua considerando gli eventi interni alla finestra ed i possibili contributi dovuti dai frammenti iniziali e finali.

4.3.4.6 Calcolo del consumo globale di acqua

Una volta che sono stati illustrati entrambi i percorsi alternativi (quelli di destra e sinistra al flusso *TemporaryWaterStream*) volti ad identificare e gestire possibili situazioni eccezionali riguardo gli eventi che appartengono solo in parte alla finestra impiegata per l'analisi, è possibile spostare l'attenzione su quello che è il reale scopo per cui l'indicatore ISCS è stato pensato: definire le interrogazioni atte a calcolare la vera e propria stima del consumo di acqua nell'abitazione intera (quindi con il contributo di tutti e tre i sensori) nelle ultime 24 ore.

Poiché l'analisi, rappresentata dal diagramma di flusso Figura 4.8, è eseguita ogni qual volta un nuovo evento di attivazione completo viene rilevato tramite il sistema di interrogazioni sviluppato, anche la somma del consumo nella finestra di 24 ore viene eseguita ogni volta. Questa analisi si svolge costruendo una finestra avente come estremo finale l'istante in cui termina l'ultimo evento (di uno qualsiasi dei tre sensori considerati) rilevato ed inserito nel flusso *TemporaryWaterStream,* e come evento iniziale il valore di tale istante sottratto di 86400 secondi, ossia quell'istante un giorno prima. In questo modo il sistema costruisce una finestra lunga 24 ore in cui il consumo degli eventi parzialmente o interamente interni debba essere sommato per restituire il totale consumato nelle ultime 24 ore. Questo processo fa in modo che per ogni nuovo evento trovato venga costruita una nuova finestra che quindi produce un risultato sempre aggiornato.

Poiché ciascuna finestra costituisce un'analisi a sé, con una casistica tutta propria di eventi che le appartengono (come abbiamo visto possono esserci tipologie diverse di appartenenze a tale finestra), le situazioni che il sistema può trovarsi ad affrontare sono quattro:

1. Con la finestra considerata non vi è alcun evento di attivazione che appartenga parzialmente né all'inizio né alla fine. Il consumo totale di acqua nell'abitazione per quella finestra sarà calcolato sommando tutti e soli i contributi degli eventi completamente interni alla finestra. Questa operazione viene eseguita dall'interrogazione Query 27, il risultato prodotto, viene inserito all'interno del flusso *WaterStream*. L'esempio è quello dei casi congiunti mostrati nelle Figure 4.9 e 4.12, e rappresentato dalla freccia che da *TemporaryWaterStream* tramite la Query 27 entra in *WaterStream.*
2. Con la finestra impiegata esiste almeno un evento che abbia istante di inizio antecedente all'istante di inizio della finestra, ma istante di fine successivo e nessun frammento finale. In questo caso il consumo totale di acqua sarà dato dalla somma dei possibili "frammenti iniziali" e tutti quelli interni alla finestra. Questa operazione viene eseguita dall'interrogazione Query 28 la quale, come per la situazione precedente, inserisce il risultato ottenuto nel flusso *WaterStream.* L'esempio è quello della Figura 4.10 o 4.11 con la 4.12. Nel diagramma 4.8 questo caso è quello che considera la freccia centrale (degli eventi interni) ed il percorso di sinistra (dei frammenti iniziali).
3. Con la finestra in considerazione esiste almeno un evento che abbia istante di inizio antecedente all'istante di chiusura della finestra e l'istante di fine non sia ancora pervenuto (quindi l'evento non è ancora stato rilevato del tutto). In questo caso il consumo totale sarà stimato come la somma di tutti i contributi interni più il/i "frammento/i finale/i". A questo scopo è deputata l'interrogazione Query 29 che, come nei casi precedenti, inserisce in *WaterStream* il risultato prodotto. L'esempio sarà rappresentato dalle Figure 4.9 e 4.13 o 4.14. Nel diagramma di flusso in Figura 4.8 questo caso è quello che considera sia la freccia centrale (degli eventi interni) ed il percorso di destra (dei frammenti finali).
4. Con la finestra attuale si verificano contemporaneamente i casi 2 e 3, quindi vi sia almeno un frammento di evento all'inizio della finestra ed almeno uno alla fine. Il consumo totale, in questo caso, è dato dalla somma dei contributi di tutti gli eventi interamente appartenenti alla finestra più i frammenti appena menzionati. L'interrogazione deputata a svolgere questo compito è la Query 30 che, come per

tutti i casi, inserisce il risultato in *WaterStream*. L'esempio sarà rappresentato dalle Figure 4.10 o 4.11 e 4.13 o 4.14. Graficamente, in relazione alla Figura 4.8, questa situazione prende in considerazione tutti e tre i percorsi che vanno da *TemporaryWaterStream* ed entrano, sommati tramite la Query 30¸ nel flusso *WaterStream*

Con quest'ultimo caso analizzato, l'indicatore è perfettamente in grado di stimare in maniera corretta il quantitativo di acqua consumata in un'abitazione nelle ultime 24 ore antecedenti all'evento di attivazione allena rilevato dal flusso in ingresso.

L'ultimo passaggio è quello illustrato nel diagramma in Figura 4.8 che prevede il passaggio dal flusso *WaterStream* al flusso di output *OutuptWaterStream.* Questo passaggio viene eseguito tramite un'interrogazione Query 31, la quale ha il compito di selezionare gli eventi inseriti all'interno di *WaterStream* che risultino di interesse per il processo di analisi in corso. Nel caso del lavoro attuale si è estratto il valore giornaliero di acqua consumata. A questo scopo quindi la Query 31 è stata costruita perché al termine di ogni giorni di analisi restituisse al flusso *OutputWaterStream* (e attraverso il suo listener ad un file testuale di uscita), in consumo totale per il giorno appena concluso.

4.3.4.7 Il risultato

Giunti al termine dell'analisi offerta dall'indicatore non resta che discutere i risultati ottenuti.

ISCS permette di raggiungere un nuovo obbiettivo per i servizi di assistenza domestica tramite una struttura domotica: fornire informazioni riguardo lo stile di vita di una persona monitorata tramite l'impiego di più sensori contemporaneamente. L'informazione che viene estratta in questo caso è relativa ai consumi domestici. Misurare infatti la quantità di energia elettrica oppure gas oppure acqua che viene utilizzata dall'abitante permette di fornire del materiale molto interessante ed utile per differenti approcci analitici alla valutazione della condizione di una persona sia nel breve che nel lungo periodo.

Nell'esempio presentato, l'informazione principale che viene estratta per ogni nuovo evento di attivazione trovato sarà il consumo di acqua per tutta la casa (quindi impiegando il contributo di un insieme di sensori selezionati). Per questo conto viene adottata una finestra di 24 ore, ma può essere scelta qualsiasi altra finestra, totalmente a

discrezione dello sviluppatore. Il risultato permetterà quindi elaborare molte possibili tipologie di informazioni in grado di tracciare il comportamento abitudinario e analizzare la salute della persona monitorata. Nel caso di questa tesi, è stato deciso di rilevare il valore giornaliero consumato.

Un'ulteriore informazione che viene estratta è il consumo per singolo sensore. L'introduzione dell'estrazione di questa informazione permette di esemplificare come l'indicatore presentato alla Sezione 4.2 possa essere adattato a contesti di analisi differenti, e al tempo stesso possa essere anche innestato all'intero di un sistema più ampio.

Attraverso opportuni listener è permesso stampare a video e su opportuni file testuali i risultati. Poiché entrambe le informazioni vengono calcolate ogni volta che un nuovo evento è riconosciuto dal sistema la struttura di messaggi che vengono mostrati è quello del riconoscimento di eventi di consumi da parte dei tre sensori dei sanitari, quindi di *TapStream*, *WcStream* e *CabinetStream* seguiti dapprima dal calcolo del consumo totale per sensore e poi di quello globale che consideri tutti i sensori.

La Figura 4.15 mostra un esempio di questa analisi. Innanzitutto viene riconosciuto un evento dello scarico e viene inserito nel flusso *WcEvent*, in seguito viene calcolato il totale dei contributi nelle ultime 24 ore antecedenti l'evento per il sensore dello scarico ed il risultato è inserito in *TotalWcEvent.* In seguito viene calcolato il totale di acqua consumata in tutta la casa per la stessa finestra che prevedrà anche l'estrazione di un frammento finale. Il risultato sarà inserito in *WaterEvent.* In seguito verranno poi trovati e studiati gli eventi successivi riconosciuti dal flusso.

```
Event received: WcEvent{sensorID=18, consumption=9.0, finActivation=140359.0}
Event received: TotalWcEvent{totalWcConsumption=189.0}
Event received: WaterEvent{totalWaterConsumption=634.6340000000312, timestamp=140359.0, paramForGrouping=0}
Event received: NullCabinetFragment{fragmentation=9.576, sensorID=14, timestamp=140359.0}
Event received: SumFinalFragments{sumFragments=9.576, instant=140359.0}
Event received: WaterEvent{totalWaterConsumption=644.2099999999957, timestamp=140359.0, paramForGrouping=0}
Event received: TapEvent{sensorID=17, duration=9.0, consumption=0.72, initActivation=140384.0, finActivation=140392.0}
Event received: TotalTapEvent{totalTapConsumption=55.599999999999992}
Event received: WaterEvent{totalWaterConsumption=635.354000000031, timestamp=140392.0, paramForGrouping=0}
```

Figura 4.15: Estratto del risultato stampato a video da parte dell'indicatore ISCS.

La Figura 4.16 mostra invece un esempio di risultati ritornati al file di output. Come si nota, per ciascun giorno analizzato viene restituito il totale di acqua utilizzata.

```
OutputWaterEvent { totalWaterConsumption= 103.779 , dayAnalyzed= 1.0 }
OutputWaterEvent { totalWaterConsumption= 83.459 , dayAnalyzed= 2.0 }
OutputWaterEvent { totalWaterConsumption= 110.97899999999998 , dayAnalyzed= 3.0 }
OutputWaterEvent { totalWaterConsumption= 104.779 , dayAnalyzed= 4.0 }
OutputWaterEvent { totalWaterConsumption= 83.16 , dayAnalyzed= 5.0 }
OutputWaterEvent { totalWaterConsumption= 28.579 , dayAnalyzed= 6.0 }
OutputWaterEvent { totalWaterConsumption= 51.81900000000001 , dayAnalyzed= 7.0 }
OutputWaterEvent { totalWaterConsumption= 84.459 , dayAnalyzed= 8.0 }
OutputWaterEvent { totalWaterConsumption= 85.899 , dayAnalyzed= 9.0 }
```

Figura 4.16: Esempio di contenuto di file restituito.

Capitolo 5

Indicatori per la valutazione della routine di attività complesse

In questo capitolo sono introdotti tre indicatori i quali presentano una complessità ed un'astrattezza maggiore rispetto agli indicatori presentati nel capitolo precedente. Le informazioni per descrivere la routine che qui vengono illustrate hanno lo scopo di descrivere il livello di prevedibilità di vita dell'abitante della casa rispetto ad un singolo sensore (ISVUS, Sezione 5.1) e rispetto alla casa intera (ISVIS, Sezione 5.3). Inoltre viene identificato dalle sole attivazioni di un insieme di sensori se la persona monitorata stia svolgendo un'azione complessa (IRAC, Sezione 5.2).

5.1 ISVUS: Indicatore di Stile di Vita rispetto ad Un sensore

L'indicatore ISVUS (*Indicatore di Stile di Vita rispetto ad Un Sensore*) descritto in questo capitolo calcola il valore dell'entropia per un singolo sensore. Rispetto agli indicatori precedenti questa informazione risulta più astratta, ma al contempo enfatizza la valutazione dell'abitudinarietà dello stile di vita della persona monitorata.

5.1.1 Lo scenario

La valutazione della prevedibilità dello stile di vita continua la ricerca di strumenti eterogenei per abilitare il progetto BRIDGe alla caratterizzazione della routinea della persona monitorata nell'abitazione ARAS. Poiché il lavoro ha bisogno che sia scelto un sensore per il quale svolgere il calcolo dell'entropia, tra quelli disponibili nella struttura abitativa è stato scelto quello del letto, perché già introdotto nella Sezione 4.2 del capitolo precedente. Pertanto la stessa procedura che in questa sezione viene proposta per il

sensore del letto può essere applicata a qualsiasi altro sensore. Per quanto concerne le caratteristiche dell'entropia, come riportato nella Sezione 2.4.4, essa è una misura che nella teoria dell'informazione quantifica il livello di novità di un messaggio ricevuto: nel caso attuale tale messaggio sarà lo stato (quindi '1' oppure '0') restituito in ogni istante dal sensore scelto e, tramite la sorpresa (o imprevedibilità) con cui viene ricevuto il contenuto del messaggio, si definisce il grado di prevedibilità delle attività dell'utente relative al sensore scelto.

5.1.2 Come avviene il calcolo

Per estrarre l'entropia è stato sviluppato un nucleo di interrogazioni EPL che processa il flusso di eventi in ingresso costruito dal generatore di eventi leggendo il dataset adottato. Poiché l'entropia è un'informazione il cui valore nel caso di un singolo sensore viene calcolato impiegando le probabilità che sia restituito stato '1' oppure '0', questa dipende dal periodo di tempo impiegato per l'analisi. A tale scopo si è scelto di usare due finestre temporali:

- una *finestra oraria*, che estrae il valore dell'entropia per il sensore scelto per ogni ora della giornata.
- Una *finestra giornaliera*, che calcola l'entropia per ogni giorno di analisi.

A livello pratico, per il calcolo dell'entropia oraria, ISVUS estrae gli eventi di attivazione relativi al sensore scelto dal flusso in ingresso al sistema e somma le durate dei singoli eventi quando questi appartengono alla stessa ora. Tale somma è impiegata nella formula generale dell'entropia. Nel momento in cui viene trovato il primo evento dell'ora successiva, viene restituito in un apposito file di destinazione il valore dell'entropia dell'ora precedente, calcolata utilizzando la somma totale delle durate per il sensore scelto. Nel caso di eventi che coprono ore adiacenti, questi vengono scomposti nei frammenti appartenenti correttamente alle singole ore. I risultati dell'entropia così calcolati vengono raccolti in file testuali giornalieri: per ogni giorno verrà creato un nuovo file per raccogliere 24 entropie orarie, una per una singola ora.

L'indicatore per il calcolo dell'entropia giornaliera funziona allo stesso modo, ma i riferimenti temporali per valutare questa grandezza non sono più le ore ma bensì i giorni.

I risultati prodotti vengono salvati in un file testuale destinato a raccogliere tutte le entropie giornaliere prodotte. In entrambi i casi l'entropia viene calcolata impiegando la formula introdotta nel Capitolo 2, Sezione 2.4.4 :

$$H(X) = -\sum_{i=0}^{n} p_i \log_2 p_i$$

In cui i assume i soli valori '0' e '1' pari ai due possibili stati del sensore e p_i è la probabilità che si verifichi uno stato nell'intervallo di tempo verificato.

5.1.3 Esempio ISVUS: entropia oraria e giornaliera per il sensore del letto

In questa sezione viene proposto un esempio di struttura di interrogazioni EPL per processare i dati in ingresso tramite il flusso *SensorStream*, così da identificare e raccogliere gli eventi di attivazione per il sensore del letto e di calcolare l'entropia oraria e giornaliera della persona monitorata. La scelta del sensore, fra i 20 a disposizione nella configurazione ARAS è ricaduta su quella del letto poiché già usato in precedenza.

La Figura 5.1 rappresenta il flusso di operazioni che le interrogazioni sviluppate svolgono per ricercare correttamente i risultati richiesti. La notazione nel diagramma è la stessa di quella adottata nel capitolo precedente: il *triangolo iniziale* rappresenta il flusso impiegato per l'analisi (quindi quello del letto), le interrogazioni sono rappresentate nei *cerchi*, i flussi e le finestre costruite internamente al sistema sono invece rappresentate dai *rombi* ed infine i *triangoli finali* rappresentano gli output, i flussi ed i file di destinazione in cui vengono inseriti i risultati ricercati. Nelle sezioni a seguire viene introdotto il procedere dell'analisi con riferimenti alle interrogazioni che sono elencate nell'Appendice B Sezione B1.

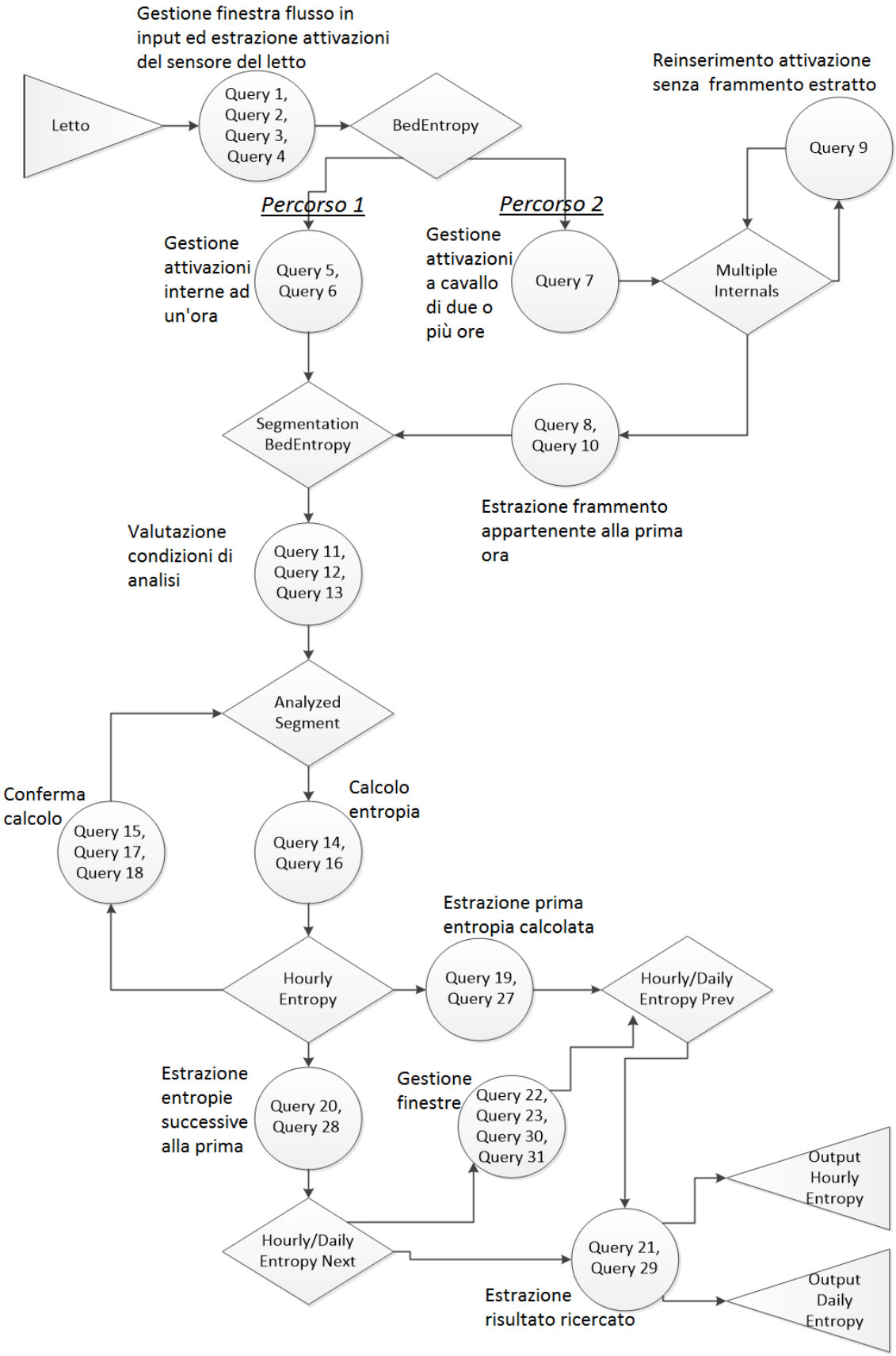

Figura 5.1: Diagramma di flusso per il calcolo di entropia oraria e giornaliera.

5.1.3.1 Estrazione attivazioni del sensore

L'inizio dell'analisi di questo indicatore, come per i precedenti, prevede una serie di interrogazioni volte ad estrarre dal flussi in ingresso *SensorStream* i soli eventi di attivazione per il sensore, scelto, del letto.

La prassi impiega due interrogazioni, la Query 1 e la Query 2, che hanno lo scopo di costruire una finestra (*BedMovingEvents*) sul flusso in ingrasso *SensorStream* per ospitare due eventi (attivazione e disattivazione) consecutivi rilevati dal sensore scelto. Quando la finestra è piena, ad ogni coppia di eventi, le interrogazioni Query 3 e Query 4 verificano di essere in presenza, per il sensore, di un evento di attivazione seguito da uno di disattivazione. In questo modo, le interrogazioni estraggono le informazioni complete (durata, istante di inizio, di fine ed ora di appartenenza di tali istanti) per ciascun periodo di attivazione del sensore e le inseriscono (come un singolo evento) nel flusso *BedEntropy*, destinato a raccogliere tutte le attivazioni del sensore scelto. In questo modo viene descritto il passaggio di estrazione dal *SensorStream* al flusso *BedEntropy* rappresentato all'inizio della Figura 5.1

Le interrogazioni Query 3 e Query 4 eseguono lo stesso compito, ma in presenza di situazioni differenti, rappresentate nella Figura 5.2: (a) l'attivazione che può essere estratta dalla finestra appartiene interamente ad un unico giorno e (b) l'attivazione si svolge a cavallo di due o più giorni, quindi con istante di inizio interno ad un giorno e istante di fine appartenente ad un altro. Il riconoscimento di queste due tipologie di eventi risulterà necessario al fine di calcolare correttamente l'entropia per la singola fascia oraria e per il singolo giorno.

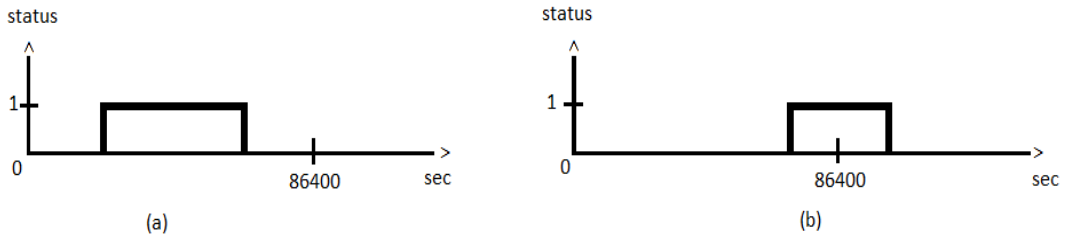

Figura 5.2: Esempio di eventi riconosciuti: (a) attivazione interamente appartenente ad un singolo giorno, (b) attivazione che si sviluppa "a cavallo" di due giorni.

Per distinguere i due casi, vengono impostati dalle due interrogazioni dei parametri atti a permettere all'analisi di sapere da quale situazione provenga l'evento in considerazione.

5.1.3.2 Procedura per calcolare l'entropia oraria

Nella sezione presente sono introdotte le procedure per lavorare con tutti gli eventi di attivazione che, di volta in volta, vengono trovati dalle interrogazioni della Sezione 5.1.3.1. Operare in maniera ordinata in questo processo risulta fondamentale per produrre i risultati corretti. A questo scopo il diagramma di flusso (in Figura 5.1) presenta un articolato sviluppo. Esso prevede due percorsi possibili che partono dal flusso *BedEntropy*:

I. **Percorso 1** in cui l'evento trovato nella routine di rilevazione di attivazioni (Sezione 5.1.3.1) inizia e finisce all'interno di una stessa ora. La Figura 5.3 mostra un esempio di questa situazione.

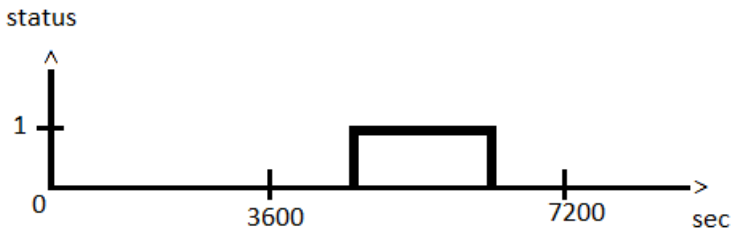

Figura 5.3: Esempio di evento interamente appartenente ad una singola ora.

II. **Percorso 2** in cui l'evento trovato inizia in un'ora e finisce in un'altra (gli istanti di inizio e di fine appartengono quindi a due ore differenti). La Figura 5.4 mostra due esempi di questa situazione: il caso (a) in cui l'evento inizia e finisce in due ore adiacenti, il caso (b) in cui l'evento inizia e finisce in due ore non adiacenti.

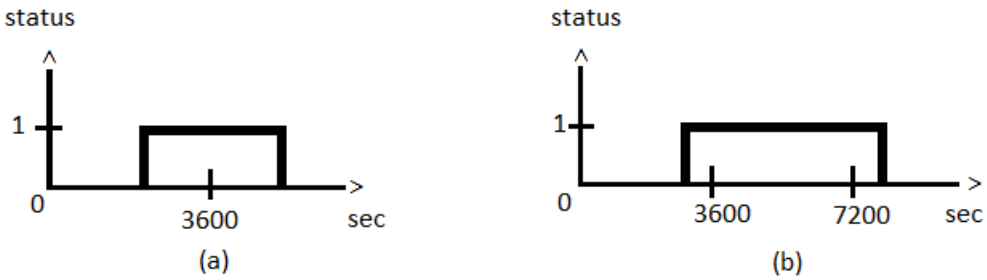

Figura 5.4: Due esempi distinti di situazione in cui l'attivazione ha istanti di inizio e fine in due ore diverse.

5.1.3.2.1 Svolgimento Percorso 1

Il Percorso 1 viene seguito ogni qualvolta nella Sezione 5.1.3.1 viene estratto ed inserito nel flusso *BedEntropy* un evento i cui istanti di inizio e di fine appartengono ad una stessa ora (come mostrato nella Figura 5.3). In questo processo l'evento in analisi (l'ultimo aggiunto al flusso *BedEntropy*) viene dapprima inserito (tramite l'interrogazione Query 5) nel flusso *SegmentationBedEntropy*, nel quale sono raccolte tutte le attivazioni riconosciute e per le quali debba essere calcolata l'entropia. L'evento appena inserito è confrontato con l'ultimo per il quale è stata calcolata l'entropia (l'ultimo evento inserito in *HourlyEntropy*). Il confronto quindi fra gli ultimi eventi inseriti in *SegmentationBedEntropy* e *HourlyEntropy* può seguire la casistica di situazioni seguente:

i. Se si è in presenza del primo evento di attivazione assoluto per il sensore, non esiste un termine di confronto riguardo un'entropia precedentemente calcolata. L'evento di *SegmentationBedEntropy* quindi è subito inserito nel flusso *AnalyzedSegment* tramite l'interrogazione Query 11. Tale flusso prevede per ogni evento, oltre alle proprietà già in *SegmentationBedEntropy*, due parametri *approved* e *inserted*. Il primo è impiegato per valutare se per l'evento possa essere calcolata immediatamente l'entropia (valore '1') oppure no (valore '0'). Il secondo, invece, viene impiegato successivamente per indicare che l'entropia è stata calcolata (valore '1'). In questo caso sono assegnati '1' per *approved*, poiché è il primo evento e può subito essere calcolata l'entropia, e '0' per *inserted.*

ii. Se l'evento di *SegmentationBedEntropy* avviene nella stessa ora oppure in quella immediatamente successiva all'ultimo evento di *HourlyEntropy* per il quale è stata calcolata l'entropia, l'interrogazione Query 12 inserisce l'evento di *SegmentationBedEntropy* in *AnalyzedSegment* con '1' per *approved*, perché si può subito calcolare l'entropia, e '0' per *inserted*. La Figura 5.5 mostra i due casi coperti da questa procedura.

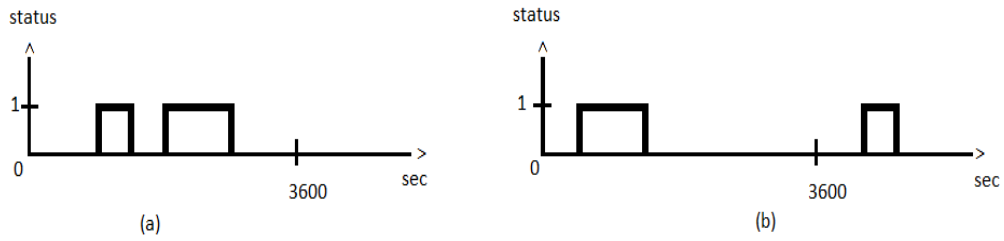

Figura 5.5: Esempio dei due situazioni coperte da questo caso: (a) quando l'ultimo evento è nella stessa ore del suo precedente, (b) quando è nell'ora immediatamente successiva.

iii. Se l'evento di *SegmentationBedEntorpy* avviene in un'ora non immediatamente successiva all'ultimo evento per il quale è stata calcolata l'entropia, ossia intercorre fra i due eventi almeno un'ora intera in cui non viene rilevata nessuna attivazione per il sensore impiegato, è necessario che il sistema restituisca tale vuoto di informazioni prima di poter calcolare per l'evento la sua entropia. In questo caso l'interrogazione Query 13 inserisce in *AnalyzedSegment* un evento con tutte le stesse proprietà di quello in considerazione da *SegmentationBedEntropy*, ma con '0' per *approved* (volto a significare che non vi è la possibilità di calcolare immediatamente l'entropia, ma solo dopo aver colmato l'assenza informativa) e '0' per *inserted*. La Figura 5.6 mostra un esempio di situazione in cui intercorre almeno un'ora interamente priva di attivazioni rilevate fra i due eventi più recenti.

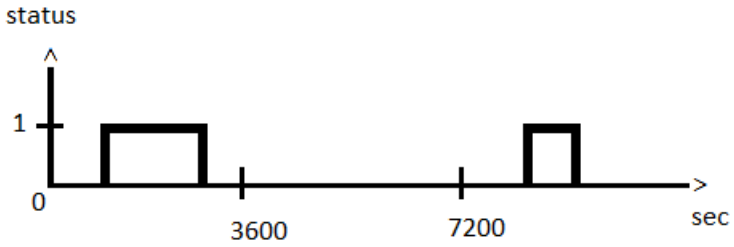

Figura 5.6: Esempio di eventi con un'intera ora senza attivazioni rilevate fra loro.

In questo modo viene realizzato quello che nella Figura 5.1 è rappresentato come il passaggio che avvia il Percorso 1 partendo dal flusso *BedEntropy* e che tramite *SegmentationBedEntropy*, giunge fino al flusso *AnalyzedSegment*.

5.1.3.2.2 Svolgimento Percorso 2

Il Percorso 2 (mostrato nella Figura 5.1) viene seguito invece ogni qualvolta nella Sezione 5.1.3.1 venga estratto ed inserito nel flusso *BedEntropy* un evento i cui istanti di inizio e di fine non appartengano ad una stessa ora (l'evento inizia in un'ora e finisce in un'altra, come mostrato dalla Figura 5.4). L'analisi richiede che l'evento sia scomposto in parti corrispondenti alle singole ore durante le quali tale attivazione è rilevata, come rappresentato nella Figura 5.7: (a) mostra il caso semplice in cui l'evento venga scomposto in due frammenti in corrispondenza della fine dell'ora, tali frammenti saranno analizzati singolarmente poiché appartenenti a ore diverse; (b) mostra il caso più complesso in cui la parte compresa fra 3600 e 7200 verrà isolata ed inserita nel flusso per un'analisi ad hoc.

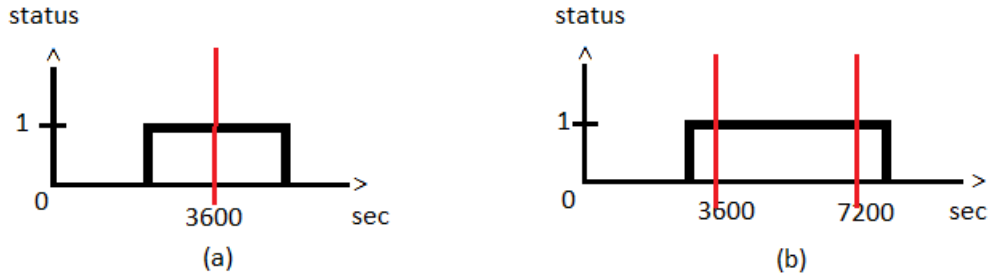

Figura 5.7: Esempio di frammentazione di attivazioni.

La procedura di questo percorso analitico che prevede la frammentazione degli eventi non appena sono inseriti nel flusso *BedEntropy* si articola in cinque interrogazioni.

i. Se l'evento più recente inserito nel flusso *BedEntropy* presenta istante iniziale appartenente ad un'ora differente dall'ora a cui appartiene l'istante finale, viene inserito nel flusso *SegmentationBedEntropy* solo il frammento che risiede nell'ora in cui l'evento di attivazione inizia. Questo frammento sarà quindi processato secondo l'analisi del *Percorso 1*. Nella Figura 5.7 il frammento considerato è, in ambo i casi, quello prima dell'istante 3600. La Query 6 svolge questo compito.

ii. L'evento più recente inserito nel flusso *BedEntropy*, per il quale è stato appena estratto il frammento iniziale nello *passaggio i*, viene inserito nel flusso *MultipleInternals*. L'evento che però è aggiunto in questo flusso è privato sia del frammento iniziale (che copre l'ora in cui inizia l'evento, già trattato nel passaggio

precedente) che quello finale (che copre l'ora in cui l'evento finisce). Questo passaggio è impiegato solo dagli eventi i cui istanti di inizio e di fine non risiedono in ore adiacenti, ma bensì coprono almeno un'intera ora (tutti i casi simili a quello rappresentato nella Figura 5.7 (b) dove il frammento estratto è quello compreso fra 3600 e 7200 secondi). Per svolgere questo compito è stata sviluppata la Query 7, rappresentata nella figura 5.1 dal cerchio che collega *BedEntropy* con il flusso *MultipleInternals.*

iii. Nel caso in cui sia stato inserito un evento in *MultipleInternals*, da questo flusso viene estratto il frammento corrispondente alla prima ora ed è inserito nel flusso *SegmentationBedEntropy* quando l'analisi del frammento precedente estratto nel passaggio *i* sia terminata. In questo modo tale frammento prelevato verrà indirizzato all'analisi del *Percorso 1* finalizzato a calcolarne l'entropia. La Query 8 esegue questo passaggio da *MultipleInternals* a *SegmentationBedEntropy*.

iv. L'eventuale parte di evento che non è stata prelevata nel *passaggio iii*, viene inserita in *MultipleInternals* perché possa essere analizzata ripetendo il passaggio precedente. Il processo che itera fra i *passaggi iii* e *iv* permette infatti di analizzare e calcolare le entropie di tutte le fasce orarie in cui il sensore scelto risulta interamente attivo. Procedendo una fascia oraria (un frammento orario) alla volta, viene esaurita l'analisi interna dell'evento spogliato dei suoi frammenti iniziali e finali al passaggio *ii*. La Query 9 permette di svolgere questo passaggio, qualora vi fosse la condizione. Nel diagramma (Figura 5.1) il passaggio è rappresentato dalla freccia che rientra in *MultipleInternals.*

v. Al termine del processo iterativo fra i passaggi *iii* e *iv*, avviene, tramite la Query 10, l'analisi del frammento finale dell'evento: quello che copre l'ora in cui termina l'evento (il secondo frammento nella Figura 4.18 (a), il terzo nella (b)). Esso viene inserito nel flusso *SegmentationBedEntropy* perché sia considerato pure il suo contributo per il calcolo dell'entropia per l'ora in cui esso appartiene.

Questa procedura ricomincia ogni volta che nel flusso *BedEntropy* viene inserito un evento i cui istanti di inizio e di fine appartengano a due ore differenti.

5.1.3.3 Calcolo entropia oraria

Entrambi i percorsi in cui può snodarsi l'analisi degli eventi studiati terminano (come si vede nel diagramma in Figura 5.1) con la scrittura di un evento nel flusso *AnalyzedSegment* (gestito nella Sezione5.1.3.2.1). Se l'evento inserito presenta nella proprietà *approved* valore '1', volto a significare che l'evento in analisi è temporalmente il successivo all'ultimo per il quale è stata calcolata l'entropia, allora il calcolo di tale misura può essere svolto. In questa sezione viene perciò illustrato il passaggio che muove dal flusso *AnalyzedSegment* a *HourlyEntropy*, cioè calcola il contributo all'entropia dell'ora a cui appartiene l'evento considerato dal primo flusso.

Il processo di calcolo dell'entropia oraria è molto semplice: per ogni nuovo evento inserito in *AnalyzedSegment* avente *approved* '1' e *inserted '0'*, cioè per il quale debba essere calcolata l'entropia (*approved = 1*) e che questa non sia ancora stata inserita nel flusso *HourlyEntropy (inserted = 0),* viene calcolata l'entropia utilizzando l'interrogazione Query 14 che applica la formula introdotta nel Capitolo 2, Sezione 2.4.4.

In questo modo per ogni nuovo evento inserito in *AnalyzedSegment* sarà calcolata l'entropia per l'ora in cui tale evento appartiene considerando tutti gli eventi presenti in tale ora. Il risultato viene restituito come un nuovo evento all'interno del flusso *HourlyEntropy*.

Effettuata questa operazione segue un'interrogazione, la Query 15, in cui viene reinserito in *AnalyzedSegment* l'evento per il quale è appena stata calcolata l'entropia con l'interrogazione 14. In questo caso però alla proprietà *inserted* sarà attribuito il valore '1' volto a simboleggiare che il contributo dell'evento è stato calcolato nell'ultima entropia oraria inserita nel flusso *HourlyEntropy*.

Lo svolgimento di quest'ultima interrogazione garantisce la prosecuzione dell'intero processo di analisi, infatti soltanto gli eventi con proprietà *inserted = '1'* permettono di procedere con l'inserimento in *AnalyzedSegment* dell'evento successivo a quello appena analizzato.

5.1.3.3.1 Caso di evento in *AnalyzedSegment* con *approved = '0'*

Nella sezione presente viene analizzato un caso estremo: quando l'ultimo evento inserito in *AnalyzedSegment* presenta proprietà *approved = '0'* e, cioè, non appartiene né alla stessa ora né all'ora successiva a quella per la quale è stata calcolata l'ultima entropia. Questo vuol dire che fra i due eventi è presente almeno un'ora intera durante la quale non viene riscontrato nemmeno un evento di attivazione (si ricorda la Figura 5.6). Pertanto è necessario sviluppare una procedura che vada ad inserire iterativamente un'entropia nulla per ogni ora nella quale non è stata rilevata alcuna attività.

La procedura sviluppata si articola su tre interrogazioni:

i. Viene inserita un'entropia nulla nel flusso *HourlyEntropy* per l'ora successiva a quella dell'ultimo evento per il quale è stata calcolata. In questo modo viene fornita informazione riguardo l'assenza di attivazioni per quell'ora specifica. Per l'esecuzione di questo punto viene impiegata la Query 16 che nel diagramma di flusso (Figura 5.1) collega il flusso *AnalyzedEntropy* con *HourlyEntropy*.

ii. Se l'ora per la quale è stata appena inserita entropia nulla (*punto i*) in *HourlyEntropy* è immediatamente antecedente a quella dell'evento presente in *AnalyedSegment*, allora questo evento, tramite la Query 17, viene inserito nuovamente in *AnalyzedSegment* ma questa volta con *approved* impostato a '1', volto a significare che, ora che è stato colmato il vuoto di informazioni, può essere calcolata l'entropia dell'evento in analisi. A questo punto seguirà l'esecuzione della Query 14 presentata nella sezione precedente per il calcolo dell'entropia. Questo passaggio nel diagramma (Figura 5.1) è rappresentato con la freccia che ritorna nel flusso *AnalyzedSegment* da *HourlyEntropy*.

iii. Se l'ora per la quale è stata appena inserita un'entropia nulla (*punto i*) in *HourlyEntopry* non è immediatamente antecedente a quella dell'evento considerato da *AnalyzedSegment*, allora l'evento viene inserito nuovamente nel flusso *AnalyzedSegment* ed ancora una volta con valore '0' per la proprietà *approved*, perché sia simboleggiata la necessità di inserire almeno un altro elemento entropico nullo in *HourlyEntropy*. Questa operazione viene svolta dalla Query 18 ed è

rappresentata nel diagramma (Figura 5.1) dalla freccia che rientra nel flusso *AnalyzedSegment*. A questa seguirà una nuova esecuzione del *punto ii*.

In questo modo viene garantita una copertura informativa totale ed ordinata tale da permettere la restituzione di risultati continuativamente validi in termini di entropia calcolata ora dopo ora.

5.1.3.4 Scrittura su file di destinazione

Una volta calcolata l'entropia oraria nella sezione precedente per ogni singolo evento di attivazione che viene rilavato per il sensore del letto, rimane soltanto la gestione della scrittura su di un file di destinazione dell'entropia per il singolo giorno di analisi. L'obbiettivo è quello di produrre un file per ciascun giorno di analisi al cui interno siano raccolte le 24 entropie corrispondenti alle 24 ore che compongono una singola giornata. Poiché il calcolo dell'entropia oraria come presentato nella sezione precedente viene effettuato ogni volta che è rilevato un evento per il singolo sensore, risulta che per la singola ora analizzata potrebbero esistere un numero variabile di entropie a seconda del numero di attivazioni registrate dal sensore in un'ora specifica. Pertanto è necessario estrarre per ciascuna ora di analisi, l'ultimo valore di entropia calcolato. Questo può essere estratto e scritto nel file di destinazione solo nel momento in cui viene rilevato il primo evento dell'ora successiva. A questo scopo è stata adottata una routine descritta dal grafico presente nella Figura 5.1 che porta dal flusso *HourlyEntropy* al flusso in output *OutputHourlyEntropy*.

Il diagramma mostra che la procedura si basa sul confronto delle ore di appartenenza delle due entropie più recenti calcolate ed inserite nel flusso *HourlyEntropy*. In particolare verranno utilizzare due finestre: *HourlyEntropyPrev* destinata a raccogliere la prima attivazione per la quale è stata calcolata l'entropia (tramite la Query 19) e poi sempre il penultimo evento più recente inserito in *HourlyEntropy* mentre *HourlyEntropyNext* ospita sempre l'evento più recente (tutti gli eventi successici al primo sono inseriti qui tramite l'interrogazione Query 20). Se gli eventi presenti nelle due finestre appartengono a due ore differenti, significa che l'evento in *HourlyEntropyPrev* corrisponde all'ultima entropia calcolata per l'ora cui appartiene e quindi viene riportato nel file in output scrivendolo nel flusso *OutputHoulyEntropy* tramite la Query 21. A

questa interrogazione è associato un listener in grado di scrivere su di un file testuale le informazioni giornaliere estratte. Se gli eventi invece appartengono alla stessa ora, non è possibile stabilire se quello in *HourlyEntopyNext* sia l'ultimo per tale ora, potrebbe essere seguito da altri eventi appartenenti alla medesima ora. In entrambi i casi quindi al termine della valutazione, l'evento presente nella finestra *HourlyEntopyNext* è aggiunto in *HourlyEntropyPrev* (la Query 22 quando si è scritto nel flusso di output, la Query 23 in caso negativo).

In questo modo terminano le interrogazioni impiegate per l'analisi, gestione e calcolo dell'entropia oraria relativa al sensore del letto perché il risultato che otteniamo in questa ultima sezione è la produzione del risultato che cerchiamo: un file di testo per ogni giorno di analisi in cui siano riportate ordinatamente le entropie orarie calcolate per ogni singola ora di tale giorno.

5.1.3.5 Gestione assenza di attivazioni del sensore nella prima ora di analisi

Una delle assunzioni necessarie per una corretta valutazione dell'entropia oraria per il primo giorno è che la raccolta di dati inizi fin dalla prima ora di analisi. Questo perché vi è la possibilità, non remota, che nella prima ora di analisi il sensore scelto non rilevi alcuna attività e perciò, in assenza di informazioni estratte, nessuna entropia oraria possa essere calcolata, lasciando un vuoto non trascurabile, di informazioni. La Figura 5.8 mostra un esempio di questa situazione in cui non viene rilevata alcuna attivazione nella prima ora, la prima occorrenza sarà soltanto oltre 3600 secondi (un'ora) dall'inizio.

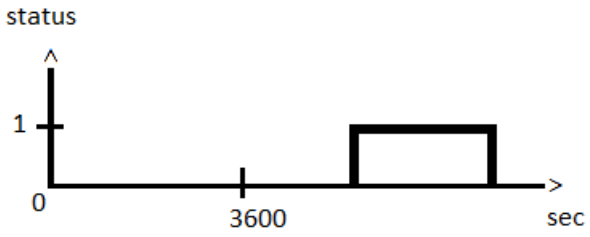

Figura 5.8: Esempio di situazione in cui la prima attivazione è rilevata solo dopo lo scoccare della prima ora di analisi

Per ovviare a questo problema la sezione presente offre una semplice soluzione attraverso una routine di tre interrogazioni che individui l'istante della prima attivazione del sensore per il primo giorno in analisi (Query 24 e 25) e verifichi se questo sia all'interno della prima ora. Se, quindi, tale istante non risiede all'interno della prima ora, l'interrogazione

Query 26 procede ad inserire nel flusso *HourlyEntropy* un evento che descriva la nullità di entropia per la prima ora di analisi.

In questo modo viene evitato una possibile, quanto fastidiosa incompletezza di risultati.

5.1.3.6 Entropia giornaliera

In modo pressoché analogo viene calcolata l'entropia giornaliera che, come per il caso precedente, viene inserita in un apposito file di testo. Il calcolo mostrato nella Sezione 5.1.3.3 viene infatti usato in maniera analoga per ottenere il valore dell'entropia per il singolo giorno di analisi. La differenza ricade esclusivamente nella variazione riguardo la durata del tempo sulla quale effettuare il conto: 24 ore (86400 secondi) invece della singola ora (3600 secondi). A differenza però dell'entropia oraria in cui ogni giorno è descritto da un nuovo file con le rispettive 24 entropie orarie, le entropie giornaliere vengono tutte raccolte in un unico file aggiornato al termine del singolo giorno con l'entropia complessiva delle 24 ore appena concluse applicando delle interrogazioni simili a quelle impiegate per l'entropia oraria nella sezione precedente (a questo scopo sono sviluppate le interrogazioni Query 27, 28, 29, 30 e 31).

L'applicazione di questo secondo indicatore è perfettamente incardinata all'interno di quello giornaliero, in questo modo è possibile ottenere da un unico indicatore ben due tipologie differenti di entropie.

5.1.3.7 Il risultato

Per quanto concerne l'entropia oraria, lavorando secondo la procedura presentata per ogni ora di analisi, ISVUS restituisce l'ultimo valore di entropia calcolata. Questo risulta essere l'entropia corretta per ciascuna ora. Tali valori orari saranno raccolti in un file ogni giorno nuovo e tale che i risultati orari calcolati siano chiaramente identificabili e distinguibili. La Figura 5.9 riporta un esempio di file giornalieri prodotti dall'analisi svolta in più giorni. La Figura 5.10 rappresenta uno stralcio di contenuto di uno di questi file. Esso riporta la singola entropia oraria calcolata e i rispettivi istanti di inizio e di fine dell'ora di analisi.

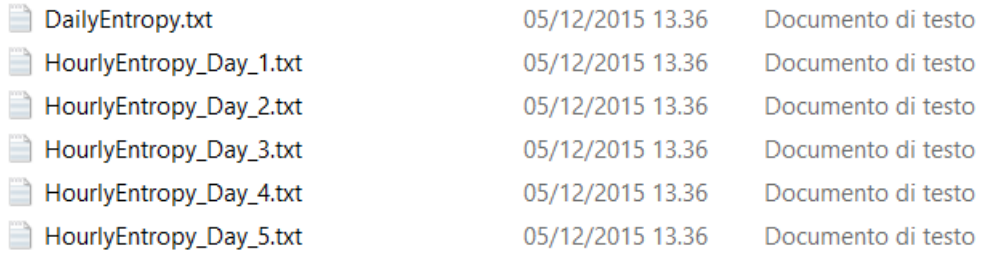

Figura 5.9: Esempio di elenco di file prodotti dai primi cinque giorni di analisi.

```
OutputHourlyEntropy { hourlyEntropy= 0.9889995147589445 initInterval= 1.0 finInterval= 3600.0 }
OutputHourlyEntropy { hourlyEntropy= 0.8014477749598377 initInterval= 3601.0 finInterval= 7200.0 }
OutputHourlyEntropy { hourlyEntropy= 0.9631672450918831 initInterval= 7201.0 finInterval= 10800.0 }
OutputHourlyEntropy { hourlyEntropy= 0.9886994082884974 initInterval= 10801.0 finInterval= 14400.0 }
OutputHourlyEntropy { hourlyEntropy= 0.806845691988262 initInterval= 14401.0 finInterval= 18000.0 }
OutputHourlyEntropy { hourlyEntropy= 0.843054496333918 initInterval= 18001.0 finInterval= 21600.0 }
OutputHourlyEntropy { hourlyEntropy= 0.7931876528975506 initInterval= 21601.0 finInterval= 25200.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 25201.0 finInterval= 28800.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 28801.0 finInterval= 32400.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 32401.0 finInterval= 36000.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 36001.0 finInterval= 39600.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 39601.0 finInterval= 43200.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 43201.0 finInterval= 46800.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 46801.0 finInterval= 50400.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 50401.0 finInterval= 54000.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 54001.0 finInterval= 57600.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 57601.0 finInterval= 61200.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 61201.0 finInterval= 64800.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 64801.0 finInterval= 68400.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 68401.0 finInterval= 72000.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 72001.0 finInterval= 75600.0 }
OutputHourlyEntropy { hourlyEntropy= 0.0 initInterval= 75601.0 finInterval= 79200.0 }
OutputHourlyEntropy { hourlyEntropy= 0.9738003694382253 initInterval= 79201.0 finInterval= 82800.0 }
OutputHourlyEntropy { hourlyEntropy= 0.9584334489746189 initInterval= 82801.0 finInterval= 86400.0 }
```

Figura 5.10: Esempio di contenuto di un file destinato all'entropia oraria.

Per quanto riguarda l'entropia giornaliera, il risultato prodotto dalla procedura proposta ha permesso di ottenere un'informazione di più alto livello. Il valore calcolato sarà inserito nel file di destinazione solo come viene rilevato il primo evento del giorno successivo. La Figura 5.11 è un esempio di risultato in cui viene riportata l'entropia giornaliera ed il giorno di analisi corrispondente.

```
OutputDailyEntropy{dailyEntropy=0.6007881085731356, dayAnalyzed=1.0}
OutputDailyEntropy{dailyEntropy=0.6453523005644963, dayAnalyzed=2.0}
OutputDailyEntropy{dailyEntropy=0.6254031831631358, dayAnalyzed=3.0}
OutputDailyEntropy{dailyEntropy=0.7154930146081835, dayAnalyzed=4.0}
OutputDailyEntropy{dailyEntropy=0.6823001219227773, dayAnalyzed=5.0}
OutputDailyEntropy{dailyEntropy=0.6583398223879314, dayAnalyzed=6.0}
OutputDailyEntropy{dailyEntropy=0.664307448544826, dayAnalyzed=7.0}
```

Figura 5.11: Esempio di contenuto del file testuale destinato alle entropie giornaliere.

Il medesimo risultato che viene riportato nei file in output è presentato anche a video, per ogni evento del sensore del letto *BedEntropy* riconosciuto viene calcolata la sua entropia oraria *HourlyEntropy* e quella giornaliera *DailyEntropy*. Nel momento in cui verrà trovato il primo evento dell'ora successiva, allora verrà mostrato a video il risultato *OutputHourlyEntropy* valevole per l'entropia oraria che sarà scritta nel file di destinazione; quando viene trovata la prima attivazione del giorno successivo sarà mostrato il risultato dell'entropia giornaliera *OutputDailyEntropy*. La Figura 5.12 mostra un esempio di sequenzialità di questi risultati.

```
Event received: BedEntropy{eventID=3, timeOn=1.0, initEvent=3545.0, finEvent=3545.0, initHour=0.0, finHour=0.0, withinSingleDay=1}
Event received: HourlyEntropy{hourlyEntropy=0.7750945958040781, partialSum=822.0, initInterval=1.0, finInterval=3600.0, timeOn=1.0, eventID=3.0, interval=0.0, countFactor=0}
Event received: DailyEntropy{dailyEntropy=0.07755303098145504, partialSum=822.0, initInterval=1.0, finInterval=86400.0, countFactor=0}
Event received: BedEntropy{eventID=4, timeOn=303.0, initEvent=9259.0, finEvent=9561.0, initHour=2.0, finHour=2.0, withinSingleDay=1}
Event received: HourlyEntropy{hourlyEntropy=0.0, partialSum=0.0, initInterval=3601.0, finInterval=7200.0, timeOn=0.0, eventID=4.0, interval=1.0, countFactor=0}
Event received: DailyEntropy{dailyEntropy=0.07755303098145504, partialSum=822.0, initInterval=1.0, finInterval=86400.0, countFactor=0}
Event received: HourlyEntropy{hourlyEntropy=0.4166931720532674, partialSum=303.0, initInterval=7201.0, finInterval=10800.0, timeOn=303.0, eventID=4.0, interval=2.0, countFactor=0}
Event received: OutputHourlyEntropy { hourlyEntropy= 0.7750945958040781 , initInterval= 1.0 , finInterval= 3600.0 }
Event received: DailyEntropy{dailyEntropy=0.10021218576990476, partialSum=1125.0, initInterval=1.0, finInterval=86400.0, countFactor=0}
Event received: OutputHourlyEntropy { hourlyEntropy= 0.0 , initInterval= 3601.0 , finInterval= 7200.0 }
Event received: BedEntropy{eventID=5, timeOn=4.0, initEvent=9563.0, finEvent=9566.0, initHour=2.0, finHour=2.0, withinSingleDay=1}
```

Figura 5.12: Risultati ISVUS.

5.2 IRAS: Indicatore per Riconoscre un'Attività Complessa

Il presente indicatore, IRAC (*Indicatore per Riconoscere un'Attività Complessa*), è stato sviluppato con lo scopo di poter astrarre una specifica attività complessa basandosi esclusivamente sull'interazione dell'abitante della casa con un insieme di sensori scelti. Il riconoscimento di questa attività, vincolato a delle prestabilite condizioni temporali, permette allo sviluppatore di scegliere arbitrariamente quali informazioni o stime estrarre da questa analisi.

5.2.1 Lo scenario

Anche per lo sviluppo di IRAC è stato adottato il riferimento al progetto ARAS, del quale sarà impiegata sia la struttura abitativa *HouseA*, sia il dataset fornito. Rispetto ai casi recedenti però, in questo contesto, la presenza dei due abitanti all'interno dell'abitazione sovente fornirà ambiguità per l'identificazione corretta dell'attività. A tale scopo quindi l'indicatore sarà sviluppato ed inizialmente verificato impiegando il dataset ARAS, ma in

un secondo momento questo verrà sostituito da quello di dati simulati costruito impiegando l'estensione al sistema SHARON introdotta nel Capitolo 3, Sezione 3.3. Questo dataset infatti simula la vita di un solo abitante nell'abitazione *HouseA* quindi permette di verificare a pieno la correttezza dei risultati. Esso permetterà inoltre di andare a testare il riconoscimento del cambiamento comportamentale dell'abitante.

Per lo sviluppo di IRAC pertanto sarà necessario individuare un'attività complessa e selezionare l'insieme di sensori che ne permetta il riconoscimento.

5.2.2 Come avviene il calcolo

Per effettuare la rilevazione corretta dell'attività scelta bisogna innanzitutto decidere quali debbano essere i sensori da prendere in considerazione, cioè quelli che rileveranno attivazioni nel momento in cui la persona monitorata stia svolgendo tale attività. Una volta presa questa decisione non resterà che sviluppare una serie di vincoli relazionali temporali tra le attivazioni rilevate dai sensori in modo che questi permettano il riconoscimento dell'attività scelta. Per svolgere questo compito viene studiato il comportamento generale che le perone seguono per l'attività ed è successivamente ricercato nel dataset a disposizione.

La valutazione perciò dell'attività dipenderà dall'analisi della combinazione delle attivazioni riconosciute dai sensori appartenenti al gruppo scelto: ad ogni nuovo evento proveniente dal flusso per i sensori scelti si è verificata la sussistenza di due condizioni che, qualora rispettate, permettano di riconoscere in maniera indiscutibile che la persona monitorata stia svolgendo l'attività scelta. Queste condizioni sono:

1) Gli eventi di attivazione devono provenire dai sensori scelti secondo alcune combinazioni stabilite (vi sia una sequenzialità di attivazioni che debba essere rispettata).
2) Tali eventi devono rispettare delle condizioni temporali fra di loro (dovranno essere rispettati dei vincoli temporali).

In questo modo l'indicatore identifica se la singola attivazione rilevata sia effettivamente un'occorrenza dell'attività che vogliamo riconoscere e, nel caso positivo,

andrà successivamente a valutare se tale evento possa essere considerato appartenente ad un insieme di eventi precedentemente accettati dal sistema. Il senso di questo passaggio è quello di procedere con l'accorpamento di singole attivazioni riconosciute da vari sensori come occorrenze distinte dell'attività scelta qualora questi si trovino molto vicini temporalmente fra di loro. Tutto ciò con lo scopo di identificare l'attività in tutta la sua durata di svolgimento.

5.2.3 Esempio IRAC: guardare la televisione

Per lo sviluppo di questa tipologia di indicatore IRAC è stato necessario decidere quale fra le possibili attività svolte dall'abitante della casa potesse essere estratta dall'analisi di un gruppo di sensori. La scelta è ricaduta sull'attività "*guardare la televisione*" poiché nella configurazione della casa *HouseA* il corretto riconoscimento di questa attività dipende da un gruppo di sensori installati nel soggiorno.

Nello specifico i sensori le cui attivazioni possono essere combinate allo scopo di rilevare l'attività di guardare la televisione sono cinque:

- Il sensore infrarosso installato sul telecomando (*sensorID = 3*) che rileva le interazioni fra il televisore ed il dispositivo per il controllo remoto. Le attivazioni rappresentano gli istanti in cui un pulsante del telecomando viene premuto ed il segnale viene mandato al televisore.
- I due sensori di pressione installati sotto i due divani (*sensorID = 4* e *sensorID = 5*) che rilevano quando la persona è seduta. Le attivazioni corrispondono alla rilevazione di una pressione sul singolo divano.
- I due sensori di distanza delle sedie (*sensorID = 6* e *sensorID = 7*) che rilevano quando queste vengono utilizzate restituendo un'attivazione ogni volta che si distanziano da una posizione prescelta. Per rilevare quando una persona è seduta, quindi la sedia sarà allontanata dalla posizione "sotto il tavolo".

Il rilevamento corretto dell'attività pertanto avviene attraverso un procedimento di identificazione degli eventi di attivazione dei sensori appena introdotti che, attraverso lo sviluppo delle due condizioni presentate nel punto precedente, verifica se questi siano effettivamente occorrenze dell'attività "*guardare la televisione*".

Per quanto concerne la *sequenzialità di attivazioni* si è stabilito che, perché sia un'occorrenza dell'attività guardare la televisione, deve essere un'attivazione del sensore del telecomando l'evento ad avviare l'analisi. Ad esso dovranno seguire le attivazioni rilevate dai sensori per sedersi.

A riguardo dei *vincoli temporali*, le attivazioni dei sensori coinvolti non devono distanziarsi tra loro per più di tre minuti (180 secondi) per poter essere considerate come un'unica occorrenza dell'attività. Quando infatti le attivazioni rilevate dai sensori scelti si presentano con una distanza temporale inferiore, allora queste saranno raggruppate per costituire un'unica occorrenza di guardare la televisione. In questo modo, associando tali eventi, è possibile costruire estese singole attività. Se la condizione invece non è rispettata, l'attivazione non può essere raggruppata alle precedenti.

Una volta che, analizzando le condizioni, si riescono ad estrarre le occorrenze di tale attività, queste saranno considerate per calcolare un'informazione: l'Active Time Ratio (ossia il tempo giornaliero) per cui viene svolta l'attività di guardare la televisione. L'informazione estratta è presentata nella Sezione 2.4.3 del Capitolo 2 come tasso di attività ed ha lo scopo di descrivere la frequenza con cui giornalmente la persona monitorata guarda la televisione, ossia quale sia la probabilità di osservare l'abitante mentre svolge tale attività.

La Figura 5.13 illustra la procedura con cui si svolge l'analisi di questo indicatore sui dati in ingresso fino all'estrazione del risultato inserito nel flusso di output. La notazione adottata è la stessa della Figura 5.1. Come per l'indicatore precedente, inoltre, le interrogazioni di IRAC sono raccolte nell'Appendice B, Sezione B2, però.

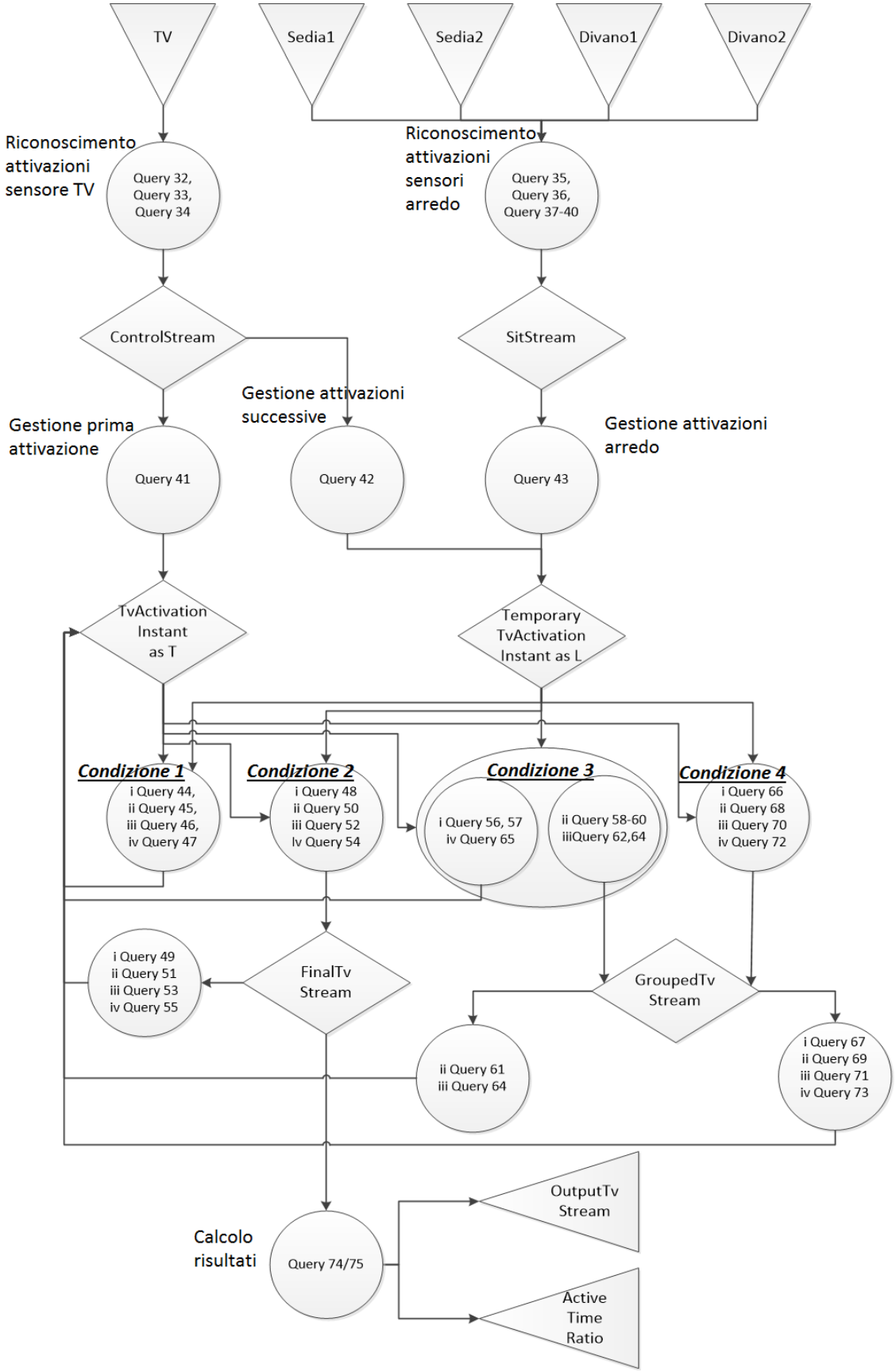

Figura 5.13: Diagramma di flusso di IRAC per riconoscere l'attività "guardare la tv".

5.2.3.1 Estrazione attivazioni dal flusso in ingresso

L'inizio dell'analisi prevede che da principio vengano identificati dal flusso in ingresso costruito dal Generatore di Eventi gli eventi di attivazione per il gruppo di sensori preso in considerazione. Poiché è stato detto che il segnale del telecomando è quello che permette di riconoscere l'inizio di un'occorrenza dell'attività guardare la televisione, allora la procedura con cui vengono riconosciute le attivazioni o disattivazioni del sensore è differente da quella impiegata per identificare gli stessi messaggi dai sensori di sedie e divani (come si vede anche dalla differenza di percorsi nella Figura 5.13).

Per il sensore del telecomando pertanto sono sviluppate tre interrogazioni: Query 32 e Query 33 impiegate per gestire la finestra *MovingControlEvents* sul flusso *SensorStream* in ingresso al sistema, e la Query 34 per riconoscere da tale finestra gli eventi interi di attivazione ed inserirli in un apposito flusso *ControlStream*, destinato a raccogliere tutte le attivazioni rilevate. In maniera analoga vengono proposte anche tre interrogazioni per riconoscere e gestire le attivazioni dei sensori di sedie e divani tramite un'altra finestra *MovingSitEvents* (interrogazioni Query 35 e 36) che viene impiegata per isolare le single attivazioni (Query 37, 38, 39, 40, una per ciascun sensore) nell'apposito flusso *SitStream*, che raccoglie le attivazioni di tutti e quattro i sensori dell'arredo.

5.2.3.1.1 Gestione attivazioni del telecomando

In questa sezione viene illustrata la modalità con cui sono gestite le attivazioni del telecomando non appena vengono inserite nel flusso *ControlStream*. Se l'evento aggiunto rappresenta la prima attivazione assoluta del sensore del telecomando, allora questo, tramite l'interrogazione Query 41, viene inserito nel flusso *TvActivationInstant*. Tutti i successivi, invece, sono inseriti tramite l'interrogazione Query 42 nel flusso *TemporaryTvActivationInstant.*

La gestione di questi due flussi è fondamentale perché gli eventi più recenti dei flussi flussi *TvActivationInstant* e *TemporaryTvActivationInstant* saranno sempre rispettivamente il penultimo evento più recente e quello più recente in assoluto. Il confronto continuo fra questi due elementi riconosciuti permette di identificare le occorrenze dell'attività "guardare la televisione".

Nel diagramma in Figura 5.13 le due interrogazioni 41 e 42 collegano il flusso *ControlStream* con *TvActivationInstant* e *TemporaryTvActivationInstant.*

5.2.3.1.2 Gestione attivazione degli arredi

Per quanto concerne la gestione degli eventi inseriti in *SitStream*, questi, non appena vengono riconosciuti, tramite l'interrogazione Query 43, sono inseriti direttamente nel flusso *TemporaryTvAvtivationInstant* poiché, come si è detto, per cominciare il riconoscimento di un'attività guardare la televisione, il primo sensore a rilevare attività deve essere quello del telecomando.

5.2.3.2 Routine di confronto: sequenza di eventi

La procedura di analisi si svolge nel modo seguente: ogni nuovo evento di attivazione viene inserito nel flusso *TemporaryTvActivationInstant* ed esso viene confrontato con il penultimo evento di attivazione trovato, presente come ultimo evento in *TvActivationInstant.* Come affermato nella Sezione 5.2.2, per riconoscere se l'ultima attivazione trovata è un'occorrenza oppure è parte di un'occorrenza dell'attività di guardare la televisione, si verificano delle condizioni.

La prima condizione che viene verificata riguarda l'ordinamento con cui le attivazioni dei sensore si presentano. Perché il confronto fra gli elementi nel flussi *TvActivationInstant* e *TemporaryTvActivationInstant* riconosca un'attività di guardare la televisione, si deve verificare dapprima quale sia l'identificativo delle attivazioni considerate nel flussi *TvActivationInstant* e *TemporaryTvActivationInstant.* Le situazioni di analisi sono quattro:

i. Il caso in cui l'identificativo del sensore, che ha rilevato l'ultimo ed il penultimo evento di attivazione (considerati rispettivamente dai flussi *TemporaryTvActivationInstant* e *TvActivationInstant*), sia per entrambi quello del telecomando (*sensorID = 3*).

ii. Il caso in cui la penultima attivazione (nel flusso *TvActivationInstant*) sia stata rilevata dal sensore del telecomando (*sensorID = 3*) mentre quella dell'ultima (considerata dal flusso *TemporaryTvActivationInstant*) sia stata rilevata da uno qualsiasi degli altri quattro sensori (*sensorID = 4 || 5 || 6 || 7*). Questi sensori sono considerati equivalenti ai fini dell'analisi.

iii. Questo caso è simmetrico al 2: il penultimo evento (quello da *TvActivationInstant*) è stato rilevato da uno dei quattro sensori del mobilio (*sensorID = 4 || 5 || 6 || 7*), mentre l'ultimo (quello in *TemporaryTvActivationInstant*) è stato rilevato dal telecomando (*sensorID = 3*).

iv. Il caso in cui sia l'ultimo che il penultimo evento di attivazione (rispettivamente in *TemporaryTvActivationInstant* e *TvActivationInstant*) siano stati rilevati da sensori del mobilio (*sensorID = 4 || 5 || 6 || 7*), anche combinati fra loro.

La casistica presente può essere incontrata ogni qualvolta avviene un confronto fra i due eventi di attivazione più recenti. Essa definisce due aspetti fondamentali: la casistica che permette di riconoscere l'inizio dello svolgimento dell'attività di guardare la televisione e l'analisi per controllare se i due eventi siano correlati fra loro e quindi appartenenti ad un'unica occorrenza dell'attività ricercata.

5.2.3.2.1 Confronto fra i due eventi più recenti

La sezione presente introduce l'analisi approfondita fra gli ultimi due eventi di attivazione rilevati e presenti nei flussi *TemporaryTvActivationInstant* (per l'ultimo) e *TvActivationInstnat* (per il penultimo). Questo confronto viene articolato come mostrato nella Figura 5.13 secondo quattro possibili condizioni che possono verificarsi.

Ciascuna di queste condizioni rappresenta la congiunzione di due condizioni:

- una ***condizione temporale*** che deve essere controllata fra i due eventi in analisi. Si valuta la differenza fra l'istante di inizio dell'evento più recente rilevato (*L.initActivation*) e quello finale del secondo evento più recente (*T.finActivation*). Il valore restituito da questa differenza è pari alla distanza temporale che intercorre fra i due eventi. Perché due eventi successivi possano definirsi correlati nel rappresentare un'occorrenza dell'attività "guardare la tv" non devono distare fra loro più di tre minuti (180 secondi). Quindi si controlla se questo valore è maggiore, uguale o minore a 180 secondi e sulla base di questo si decide se aggregare o meno i due eventi in analisi.

- Una ***condizione di aggregazione*** che verifica se il penultimo evento è stato aggregato in precedenza con il suo antecedente (quindi nel confronto immediatamente precedente). A questo scopo viene verificato il valore attribuito alla proprietà *aggregated* dell'evento presente in *TvActivationInstant* se è '0' (quindi non sia stato aggregato al suo precedente) oppure '1' (sia stato aggregato). Sulla base di questa valutazione quindi si decide se i due eventi in analisi, qualora debbano essere aggregati, se sono una nuova occorrenza (cioè non sono aggregabili a eventi precedenti) oppure possono essere aggiunti ad un'aggregazione già esistente e quindi ad accrescere temporalmente l'occorrenza riconosciuta.

Pertanto le quattro condizioni del diagramma in Figura 5.13 coprono questi casi:

- **Condizione 1:** *L.initActivation-T.finActivation > 180 & T.aggregated = 0*
- **Condizione 2:** *L.initActivation-T.finActivation > 180 & T.aggregated = 1*
- **Condizione 3:** *L.initActivation-T.finActivation <= 180 & T.aggregated = 0*
- **Condizione 4:** *L.initActivation-T.finActivation <= 180 & T.aggregated = 1*

Ciascuna di queste condizioni si può verificare per ognuno dei quattro confronti possibili introdotti nella Sezione 5.2.3.2, quindi per ciascuno di essi sono sviluppate apposite interrogazioni per la gestione della situazione.

Nelle sezioni a seguire vengono descritte caso per caso le quattro distinte situazioni generate dalle quattro condizioni appena introdotte.

5.2.3.2.1.1 Gestione *Condizione 1*

In questa sezione viene presentata la situazione in cui i due eventi più recenti (in analisi dai flussi *TvActivationInstant* e *TemporaryTvActivationInstant*) non sono temporalmente vicini abbastanza per essere considerati occorrenza della stessa attività (*L.initActivation-T.finActivation > 180*) e l'evento presente in *TvActivationInstant* non era stato aggregato al suo precedente (*T.aggregated = 0*).

Benché questa situazione possa presentarsi in tutti e quattro i casi illustrati nella Sezione 5.2.3.2, le interrogazioni sviluppate per ciascun caso presentano comportamento analogo: inserire nel flusso *TvActivationInstant* l'evento in considerazione da *TemporaryTvActivationInstant* così da permettere di ricominciare l'analisi non appena

una nuova attivazione viene riconosciuta ed inserita in *TemporaryTvActivationInstant*. Riprendendo l'elenco della Sezione 5.2.3.2, le interrogazioni, per i quattro casi di analisi in relazione ai sensori, sono:

i. Query 44,
ii. Query 45,
iii. Query 46,
iv. Query 47.

5.2.3.2.1.2 Gestione *Condizione 2*

In questa sezione viene presentata la situazione in cui i due eventi più recenti (in analisi dai flussi *TvActivationInstant* e *TemporaryTvActivationInstant*) non sono temporalmente vicini abbastanza per essere considerati occorrenza della stessa attività (*L.initActivation-T.finActivation > 180*), tuttavia l'evento presente in *TvActivationInstant* era stato aggregato al suo precedente (*T.aggregated = 1*).

Anche in questo caso la situazione può presentarsi nei quattro i casi illustrati nella Sezione 5.2.3.2 e le interrogazioni sviluppate presentano comportamento analogo: dapprima, con una prima interrogazione, viene inserito nel flusso *FinalTvStream* l'ultimo evento aggiunto al flusso *GroupedTvStream*; successivamente, con una seconda Query, viene messo nel flusso *TvActivationInstant* l'evento in considerazione da *TemporaryTvActivationInstant* così da permettere di ricominciare l'analisi non appena una nuova attivazione viene riconosciuta. Il flusso *GroupedTvStream* è quello in cui sono raccolte le aggregazioni di eventi qualora essi siano identificati come descrittori dell'attività guardare la tv. *FinalTvStream* invece è il flusso destinato a raccogliere le singole occorrenze finali costruite per descrivere i periodi in cui la persona monitorata guarda la televisione. Per ciascuna dei casi presentati nella Sezione 5.2.3.2, sono introdotte due interrogazioni:

i. Query 48 e Query 49
ii. Query 50 e Query 51
iii. Query 52 e Query 53
iv. Query 54 e Query 55.

5.2.3.2.1.3 Gestione *Condizione 3*

In questa sezione viene presentata la situazione in cui i due eventi più recenti (in analisi dai flussi *TvActivationInstant* e *TemporaryTvActivationInstant*) sono temporalmente vicini abbastanza per essere considerati occorrenza della stessa attività (*L.initActivation-T.finActivation <= 180*), tuttavia l'evento presente in *TvActivationInstant* non era stato aggregato al suo precedente (*T.aggregated = 0*).

Rispetto a prima, in questa situazione per ciascuno dei casi presentati nella Sezione 5.2.3.2 vi saranno comportamenti differenti:

i. In questo caso, essendo gli eventi entrambi rilevati dai sensori del telecomando, si tiene traccia (inserendo l'evento in *TvActivationInstant* considerato dal flusso *TemporaryTvActivationInstant*) del possibile accorpamento fra di essi salvando nella proprietà *followingInstant* il valore dell'istante in cui inizia l'attivazione del telecomando (Query 56) di *TvActivationInstant.* Se però l'evento di *TvActivationInstant* presenta proprietà *followingInstant* diversa da *'-1'* allora, quel valore viene mantenuto pure nel nuovo aggiunto (poiché significa che con il suo precedente era stata fatta la stessa analisi, Query 57).

ii. In questo caso siamo in presenza del riconoscimento dell'inizio di una nuova occorrenza dello svolgimento dell'attività "guardare la televisione", pertanto si raggruppano gli eventi. Per fare questo è dapprima necessario aspettare che termini l'evento di attivazione rilevato dal sensore dell'arredo (custodito in *TemporaryTvActivationInstant*). Quando tale attivazione è presente nel flusso *SitStream* allora tramite le interrogazioni Query 58, 59 e 60 i due eventi sono raggruppati ed inseriti come un unico evento dentro il flusso *GroupedTvStream*, destinato a raccogliere e formare via via le occorrenze di attività intere. Svolto questo, l'evento presente in *TemporaryTvActivationInstant* è inserito nel flusso *TvActivationInstant* (Query 61), per dare continuazione all'analisi.

iii. Se l'attivazione dell'arredo presente in *TvActivationInstant* è stata rilevata interamente, allora le condizioni non permettono di riconoscere alcuna attività (perché deve essere l'attivazione del telecomando ad avvenire per prima), pertanto l'interrogazione Query 62 inserisce in *TvActivationInstant* l'evento in considerazione da *TemporaryTvActivationInstant*. Nel caso in cui invece

l'attivazione dell'evento presente in *TvActivationInstant* non sia terminata di essere riconosciuta, allora gli eventi in analisi potranno essere associati a rappresentare un'occorrenza di guardare la tv, grazie a questa interazione. La Query 63 pertanto aspetta che sia rilevato completamente l'evento dell'arredo presente in *TvActivationInstant* e poi lo inserisce in *GroupedTvStream.* La Query 64 poi inserisce l'evento di *TemporaryTvActivationInstant* in *TvActivationInstant* per continuare l'analisi.

iv. Anche in questo caso le condizioni non permettono di riconoscere nulla di interessante dagli eventi in analisi, quindi l'interrogazione 65 inserisce l'evento di *TemporaryTvActivationInstant* nel flusso *TvActivationInstant.*

5.2.3.2.1.4 Gestione *Condizione 4*

In questa sezione viene presentata la situazione in cui i due eventi più recenti (in analisi dai flussi *TvActivationInstant* e *TemporaryTvActivationInstant*) sono temporalmente vicini abbastanza per essere considerati occorrenza della stessa attività (*L.initActivation-T.finActivation <= 180*), e l'evento presente in *TvActivationInstant* era stato aggregato al suo precedente (*T.aggregated = 0*).

Anche in questo caso, ciascuna delle quattro situazioni di confronto presenti nella Sezione 5.2.3.2 si comporta diversamente e presenta le proprie interrogazioni.

i. Nel caso in cui entrambi gli eventi a confronto siano rilevazioni del sensore del telecomando, e quello in *TvActivationInstant* sia stato accorpato con il suo evento antecedente (quindi presenta *T.aggregated = 1*), allora, tramite la Query 66 l'evento in *TemporaryTvActivationInstant* è associato al suo precedente estendendo l'ultimo evento presente nel flusso *GroupedTvStream* (del quale fa già parte l'evento in considerazione da *TvActivationInstant*). Tale estensione avviene mantenendo l'istante di inizio dell'ultimo evento in *GroupedTvStream*, ed inserendo come l'istante di fine l'istante dell'evento in *TemporaryTvActivationInstant*. Una volta svolto questo passaggio, la Query 67 mette l'evento di *TemporaryTvActivationInstant* nel flusso *TvActivationInstant* affinché sia continuata l'analisi.

ii. Questo caso, tramite la Query 68, aggrega l'evento seduta (una volta rilevato interamente) di *TemporaryTvActivationInstant* al precedente raggruppamento

(l'ultimo evento presente in *GroupedTvStream*), a cui è stato soggetto all'iterazione precedente l'evento presente in *TvActivationInstant,*. Svolto questo passaggio, per continuare l'analisi, l'evento seduta è inserito in *TvActivationInstant* tramite l'interrogazione Query 69.

iii. Nel terzo caso, invece l'accorpamento dei due eventi in considerazione avviene estendendo (tramite Query 70) l'ultimo evento custodito nel flusso *GroupedTvStream* fino a coprire anche l'evento del telecomando (presente in *TemporaryTvAvtivationInstant*). Svolto ciò, l'interrogazione Query 71 esegue l'inserimento nel flusso *TvActivationInstant* dell'evento di *TemporaryTvActivationInstant*, così da continuare l'analisi

iv. Nell'ultimo caso, in cui entrambi gli eventi in considerazione provengono da sensori installati sugli arredi, l'accorpamento avviene esclusivamente al termine della rilevazione dell'intera attivazione dell'evento presente in *TemporaryTvActivationInstant*. Anche in questo caso l'accorpamento avviene estendendo l'evento più recente presente nel flusso *GroupedTvStream* fino al termine dell'evento in *TemporaryTvActivationInstant* (Query 72). La Query 73 permette la continuazione dell'analisi inserendo l'evento in *TvActivationInstant* da *TemporaryTvActivationInstant*.

Con quest'ultimo caso analizzato, l'indicatore riesce ad identificare autonomamente ed in tempo reale ogni volta che si presenta un'occorrenza di guardare la televisione semplicemente confrontando costantemente le due attivazioni più recenti rilevate dai sensori presi in considerazione. I risultati, ossia le singole attività identificate ed assemblate, sono raccolte nel flusso *FinalTvStream* riportando informazioni riguardo la durata e gli estremi temporali di ciascuna occorrenza. Nelle sezioni a seguire vengono mostrate alcune tipologie di risultati estraibili da questo flusso.

5.2.3.3 Estrazione attività giornaliere rilevate

Uno dei risultati utili per valutare la bontà dei risultati è restituire tutte (e sole) le occorrenza riconosciute per l'attività "guardare la televisione". In particolare si è proceduto con la raccolta in un file testuale, nuovo per ogni giorno di analisi, in cui vengono raccolte tutte le attività rilevate per quel singolo giorno.

A questo scopo è stata adottata l'interrogazione Query 74 la quale, ad ogni cambio di giorno, inserisce nel flusso *OutputTvStream* tutte le attivazione riconosciute per il giorno con le loro proprietà di durata, istante di inizio ed istante di fine. Tramite un apposito listener legato al flusso *OutputTvStream* tali eventi restituiti sono inseriti anche nel file testuale ricercato.

5.2.3.4 Calcolo *Active Time Ratio* per l'attività

Nella sezione presente viene proposta la semplice modalità di come possa essere estratta un'informazione qualora venga astratta un'attività dalla combinazione di rilevazioni di sensori. Nel caso di questo esempio, al termine di ogni giorno viene calcolato l'Active Time Ratio, una stima che calcola un valore compreso fra 0 e 1 per definire quanto tempo la persona monitorata ha trascorso guardando la tv in un singolo giorno. Questa informazione, per esempio può essere un buono strumento per valutare l'abitudinarietà della persona monitorata nello svolgere specifiche attività identificate. L'interrogazione 75 svolge questo calcolo ogni volta che un giorno di analisi termina; essa impiega tutte e sole le attivazioni riconosciute per quel giorno specifico.

5.2.3.5 Il risultato

L'indicatore, IRAC, sviluppato in questo esempio permette di estrarre da un insieme scelto di sensori l'occorrenza di un'attività specifica, le cui attivazioni definiscano con sicurezza che la persona monitorata stia svolgendo tale attività. In questo esempio è stato proposto il riconoscimento dell'attività "guardare la televisione" impiegando il sensore che rileva la comunicazione fra telecomando e televisore ed i quattro relativi all'utilizzo dei due divani e delle due sedie. Le attività rilevate possono essere altre, ma l'approccio sviluppato in questo esempio potrà essere di spunto. Il riconoscimento di queste attività permette di estrarre delle specifiche informazioni a riguardo che possano permettere di analizzare lo stato di salute della persona monitorata e le sue abitudini.

In questo studio è stato rilevato l'*Active Time Ratio*, un'informazione presa dall'esempio di indice introdotto nel Capitolo 2 Sezione 2.4.3. Essa permette di definire la porzione giornaliera di tempo trascorso dalla persona monitorata mentre è intenta a guardare la televisione.

Ciò che viene mostrato a video dall'esecuzione di questo indicatore è la successione di inserimenti nei vari flussi *TvActivationInstant*, *TemporaryTvActivationInstant* e successivamente nel flusso *GroupedTvStream* ogni volta che l'analisi produce un'aggregazione fra eventi per costruire le occorrenze di attività. Infine *FinalTvStream* raccoglie le attività intere rilevate (la Figura 5.14 mostra un esempio di questo aspetto). Questo flusso viene poi impiegato per calcolare a inizio della giornata successiva la frequenza dell'attività.

```
Change of Day

Event received: FirstDailyActivation{instantFirstActivation=86401.0, count=0.0, sensorID=0}
Event received: FinalTvEvent{timeWatchingTv=0.0, initActivation=68674.0, finActivation=69724.0, dayEnded=1, caseAnalyzed=3}
Event received: ActiveTimeRatioPerDay{activeTimeRatio=20.432870370370374, dayAnalyzed=1.0}
Event received: TemporaryTvActivationInstant{eventID=481.0, duration=0.0, timestamp=0.0, initActivation=86414.0, finActivation=0.0, sensorID=7.0}
Event received: TvActivationInstant{eventID=481.0, duration=0.0, initActivation=86414.0, sensorID=7.0, followingInstant=-1.0, aggregated=0.0, addedForDayClosing=0}
Event received: SitEvent{sensorID=7, eventID=88.0, duration=17.0, initActivation=86414.0, finActivation=86430.0, parameter=0}
```

Figura 5.14: Esempio di risultati di IRAC.

Per quanto concerne la restituzione giornaliera delle attività riconosciute, la Figura 5.15 mostra un esempio di contenuto di uno di questi file testuali. Nel terzo giorno di analisi l'indicatore ha rilevato due occorrenza dell'attività guardare la televisione.

```
OutputTvEvent { timeWatchingTv= 2753.0 , initActivation= 224019.0 , finActivation= 226772.0 }
OutputTvEvent { timeWatchingTv= 2904.0 , initActivation= 238568.0 , finActivation= 241472.0 }
```

Figura 5.15: Esempio di contenuto di un giorno di rilevazioni.

5.3 ISVIS: Indicatore di Stile di Vita rispetto a tuttI i Sensori

L'indicatore presente, ISVIS (*Indicatore di Stile di Vita rispetto a tuttI i Sensori*), ha lo scopo di fornire uno strumento utile al progetto BRIDGe per la valutazione dello stile di vita della persona che risiede presso una struttura abitativa intelligente. L'analisi delle abitudini domestiche della persona non dipende più dalla considerazione di un gruppo ristretto di sensori come effettuato per gli indicatori sviluppati in precedenza, ma bensì impiega i dati provenienti da *tutti* i dispositivi installati nell'abitazione. Pertanto è avviata una continua valutazione delle condizioni di stato dei sensori: quali fra essi siano attivi o meno per ogni istante. In questo modo è possibile costruire un'informazione di alto livello riguardo l'attività domestica: l'entropia delle attività svolta in casa.

5.3.1 Lo Scenario

Lo scenario in cui si inserisce ISVIS è il medesimo adottato per le soluzioni precedenti: la struttura abitativa *HouseA* offerta dal progetto ARAS. Rispetto però a quanto sviluppato finora, per l'analisi presente sono impiegati contemporaneamente i dati provenienti da tutti e 20 i sensori installati nell'abitazione. L'informazione che viene impiegata dal sistema è lo stato assunto da ciascun sensore in un istante particolare. In questo modo, considerando al contempo tutti e venti gli stati dei rispettivi sensori, può essere valutata una cosiddetta *configurazione* istantanea dello stato della casa, necessaria per dare una valutazione globale della vita all'interno dell'abitazione.

5.3.2 Come avviene il calcolo

Il calcolo dell'informazione ricercata dall'indicatore avviene attraverso l'impiego di due strumenti che Esper mette a disposizione dell'utente:

- *Interrogazioni EPL*, sono applicate come nei casi precedenti in continuo all'arrivo di nuovi eventi dai sensori ed innestate fra di loro perché il termine di una possa attivarne una successiva.
- *Metodo di aggregazione*, è una funzione sviluppata dall'utente. L'aggregazione avviene fra eventi multipli restituiti dall'interrogazione nella cui clausola *select* la funzione viene invocata.

Per la valutazione dell'entropia della casa intera è stato necessario assumere alcune decisioni riguardo le modalità di approccio in una situazione in cui ad essere presi in considerazione sono tutti i sensori. L'entropia è stata definita come una misura dell'imprevedibilità di un messaggio proveniente da una sorgente. Pertanto nel contesto attuale la sorgente di tali messaggi risulta la casa intera tramite i suoi 20 sensori installati ed i messaggi che vengono inviati sono le *configurazioni istantanee* date dallo stato di ciascun sensore in un secondo specifico. Quindi, per esempio, in un secondo qualsiasi, viene estratta la configurazione della casa costituita dallo stato in quell'istante di ciascun sensore installato nell'abitazione: tale configurazione sarà una sequenza di venti '0' o '1', uno per ogni sensore.

Poiché per 20 sensori che emettono due stati possibili il numero potenziale di configurazioni è $2^{20} = 1048576$ e trattare ciascuna di esse come una sequenza di 0 ed 1 non è pratico, si è deciso di considerare ciascuna di queste configurazioni istantanee come la rappresentazione binaria univoca di un numero decimale che viene estratto ogni volta che un nuovo evento, e quindi una nuova configurazione, giunge al sistema di analisi. In questo modo, come mostrato nella Tabella 5.1 si ottiene un identificativo univoco per ciascuna configurazione, tale da permetterne un impiego più agevole per la valutazione dell'entropia.

Configurazione Sensori in Binario	Equivalente Decimale
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	40

Tabella 5.4: Esempio di conversione della configurazione da binaria in decimale.

Nello specifico, l'entropia è calcolata ogni volta che il sistema di interrogazioni EPL riconosce che la casa presenta una nuova configurazione, ossia un nuovo stato generale dovuto alla variazione di stato di un sensore. Per questa analisi è impiegata una finestra di 24 ore calcolata ogni volta a partire dall'istante di conclusione della configurazione più recente assunta dalla casa. L'entropia è calcolata a livello pratico tramite una funzione di aggregazione definita in loco per estrarre tutte e sole le configurazioni rilevate negli ultimi 86400 secondi (ossia nell'ultimo giorno) antecedenti l'evento (ossia la configurazione) più recete trovato. L'entropia è calcolata impiegando la formula espressa nella Sezione 2.4.4 del Capitolo 2 per ognuna delle configurazioni che si sono presentate nell'ultimo giorno. Calcolato il valore entropico di ciascuna configurazione, queste sono sommate di modo da restituire l'entropia generale della casa per il giorno analizzato dalla finestra. L'entropia totale, poiché calcolata come la somma di tutte le entropie delle singole configurazioni, può assumere un valore compreso fra 0 e 86400, in relazione alle possibili configurazioni assunte in un giorno intero. Il carattere entropico ed imprevedibile dello stile di vita è dato pertanto da un risultato alto, mentre un risultato basso, indica massima prevedibilità.

5.3.3 Esempio ISVIS: entropia orarie e giornaliera per un'abitazione intera

In questa sezione viene brevemente illustrata una possibile soluzione Esper per rispondere alla necessità di un'informazione interessante come l'entropia per l'intera struttura abitativa presentata nel progetto ARAS. L'analisi, basandosi sulla sintassi EPL, è eseguita in maniera *continua*: in ogni istante i nuovi dati prodotti dal Generatore di Eventi nella lettura del dataset adottato giungono in ingresso al sistema sotto forma di eventi e sono processati allo scopo di controllare qualora questi verifichino delle condizioni specifiche. Tali eventi rappresentano ciascuno il cambiamento di stato della configurazione della casa; il cambiamento è dato dalla mutazione di stato di almeno un sensore.

L'analisi pertanto può essere sviluppata secondo un flusso rappresentato nella Figura 5.15 che descrive le modalità di gestione degli eventi riscontrati da qualsiasi sensore nell'abitazione per poter estrarre, in conclusione di ogni giorno di analisi, l'entropia totale per la struttura abitativa.

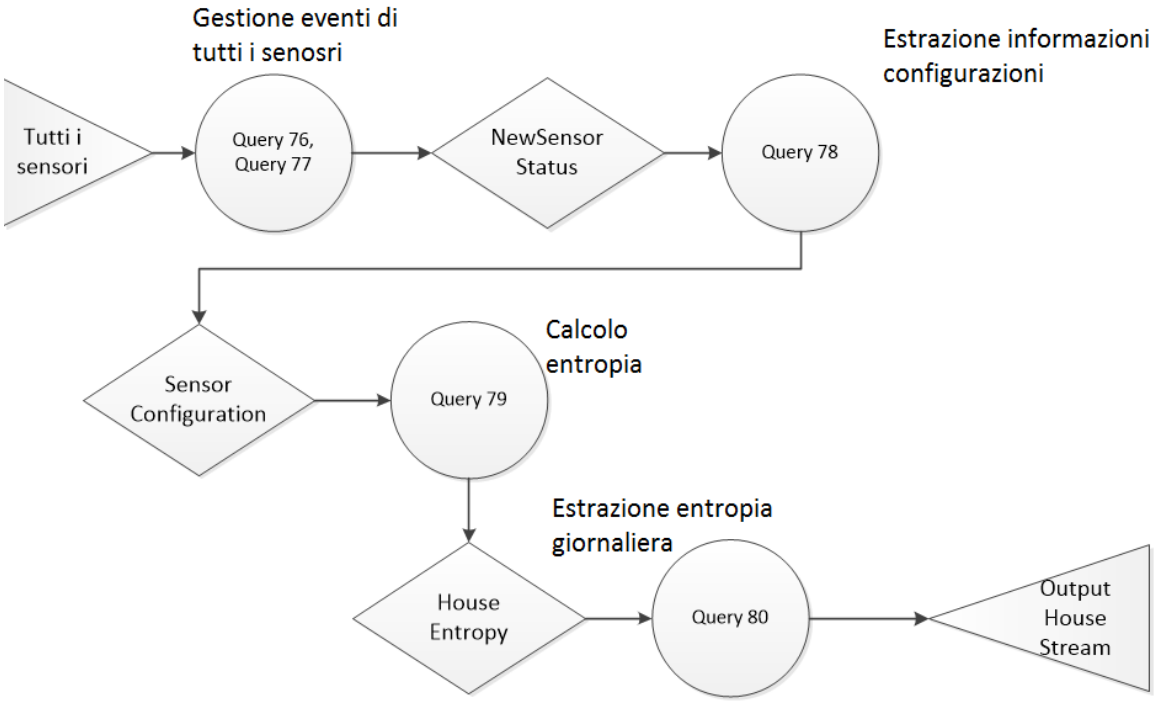

Figura 5.16: Diagramma di flusso di ISVIS per calcolo entropia di tutta la casa

Le interrogazioni sviluppate per questo processo sono raccolte nell'Appendice B, Sezione B3.

5.3.3.1 Gestione dei cambiamenti di configurazione

La sezione presente ha lo scopo di rilevare e definire lo stato generale delle condizioni dei sensori della casa ogni qualvolta uno di questi cambi di stato. Gli eventi vengono prodotti dal Generatore di Eventi che li inserisce in un flusso chiamato *SensorStream* in input al sistema di valutazione costituito da due interrogazioni. Ogni volta che avviene un cambiamento di configurazione rispetto alla situazione precedente, ossia un sensore qualsiasi cambia stato, le interrogazioni Query 76 e 77 calcolano la nuova configurazione generale della casa e la inseriscono come nuovo evento nel flusso *NewSensorStatus*, destinato a raccogliere le configurazioni il cui istante di inizio è stato appena rilevato. Nel dettaglio l'interrogazione 76 rileva quando un sensore qualsiasi inizia a mandare segnale '1' (riconosce attività), la 77 invece quando un sensore inizia ad inviare segnale '0' (termina di riconoscere attività).

Ogni volta che viene inserita una nuova configurazione nel flusso *NewSensorStatus*, la Query 78, esegue la completa estrazione delle informazioni relative alla singola configurazione di stato dei sensori, mantenuto fino all'evento appena rilevato (con una delle due interrogazioni 76 o 77). Le informazioni estratte vengono impiegate come proprietà per un nuovo evento (che descrive quindi in maniera ricercata la configurazione) inserito nel flusso *SensorConfiguration*, destinato a raccogliere tutte le configurazioni assunte dall'abitazione via via che queste vengono rilevate.

5.3.3.2 Calcolo dell'entropia

Con le interrogazioni precedenti vengono estratte le singole configurazioni assunte dai sensori installati nella casa come queste vengono riconosciute tramite l'analisi del flusso *SensorStream*. Attraverso la loro analisi sono poi estratte le informazioni complete riguardo le singole configurazioni ed i risultati sono inseriti nel flusso denominato *SensorConfiguration* (come riconoscibile dai primi passaggi del flusso in Figura 5.15). Quest'ultimo flusso viene impiegato per invocare una funzione sviluppata appositamente per il calcolo dell'entropia giornaliera. Questa funzione è definita di aggregazione poiché gestisce un numero ampio di eventi estratti tramite l'interrogazione che la invoca. Essa permette quindi il calcolo dell'entropia partendo dagli eventi ricevuti come input restituiti dall'interrogazione.

La Query 79 è quindi responsabile per la chiamata della funzione di aggregazione: essa viene invocata ogni volta che un nuovo evento (ossia una nuova configurazione) viene inserito nel flusso *SensorConfiguration*. L'interrogazione restituisce tutti e soli gli eventi (ossia le configurazioni) che sono stati rilevati nelle ultime 24 ore antecedenti l'ultimo evento (configurazione) aggiunto a *SensorConfiguration*. Gli eventi restituiti sono dati in ingresso alla funzione di aggregazione perché siano utilizzati per calcolare l'entropia relativa all'ultime 24 ore di analisi ogni volta che un evento è aggiunto a *SensorConfiguration*.

5.3.3.2.1 Funzione di aggregazione

La costruzione di una funzione di aggregazione sviluppata dall'utente in Esper prevede la definizione di due classi: una *factory* che attraverso appositi metodi valida i dati in ingresso ed in uscita dalla funzione e poi una classe per la vera e propria esecuzione della funzione. Nel caso presente la funzione di aggregazione è chiamata a calcolare l'entropia per tutti gli eventi restituiti dalla Query 79, quindi la funzione sarà chiamata *entropyComputation*, e la sua *factory EntropyComputationFactory*.
La routine svolta da *entropyComputation(K)* si articola secondo due semplici passi (*K* sono gli eventi restituiti dall'interrogazione e passati in input alla funzione):

i. **Passo 1:** una volta raccolti in un vettore chiamato *asArrInternalMemory* tutti gli eventi passati come input dalla Query 79, il primo e l'ultimo evento di tale array vengono confrontati per valutare se la finestra di analisi generi una delle situazioni illustrate nelle Figure 4.4 o 4.5 del Capitolo 4, e quindi se vi sia la necessità di estrarre un frammento di evento interno alla finestra per il primo evento. Nel caso in cui debba essere estratto suddetto frammento, questo è poi sovrascritto in *asArrInternalMemory* al posto dell'evento a cui apparteneva; in caso negativo, non viene apportata alcuna modifica al vettore.

ii. **Passo 2:** Terminata l'analisi inerente l'inizio della finestra, la funzione somma le *durationEvent* di ciascuna configurazione presente fra quelle interne al vettore *asArrInternalMemory*. Una volta calcolata la durata totale per configurazione, questa è impiegata per calcolare l'entropia per quella specifica configurazione

all'interno della finestra giornaliera in analisi. Calcolata l'entropia per ciascuna configurazione, queste sono sommate e restituiscono il valore di entropia valido per le ultime 24 ore di analisi per la casa intera.

La funzione, tramite la Query 79, restituisce un evento che viene inserito nel flusso *HouseEntropy* le cui proprietà sono l'entropia calcolata e l'istante di termine dell'ultimo evento inserito in *SensorConfiguration* il quale ha attivato l'analisi.

L'interrogazione Query 80 permette di estrarre solo l'entropia calcolata al termine di ogni giornata di analisi. In questo modo viene inserito nel flusso *OutputHouseStream* il risultato giornaliero, che, attraverso un opportuno listener, viene inserito anche in un file testuale che possa permettere analisi successive

In questo modo è fornito il risultato cercato, utile per valutare l'attività giornaliera svolta dall'abitante della casa.

5.3.3.3 Il risultato

I risultati di questo indicatore vengono inseriti in *HouseEntropy* un flusso destinato a raccogliere le entropie orarie calcolate ogni volta che viene trovato un nuovo evento in *SensorStream*. Associata alla Query 80 vi è un listener che permette di restituire i risultati al termine di ogni giorno in un apposito file di destinazione di cui la Figura 5.17 ne mostra un esempio.

```
OutputHouseEvent { dailyHouseEntropy= 2.5055688661806697 , dayAnalyzed= 1.0 }
OutputHouseEvent { dailyHouseEntropy= 2.217937342541168 , dayAnalyzed= 2.0 }
OutputHouseEvent { dailyHouseEntropy= 2.5419845686269005 , dayAnalyzed= 3.0 }
OutputHouseEvent { dailyHouseEntropy= 2.2084319432269908 , dayAnalyzed= 4.0 }
OutputHouseEvent { dailyHouseEntropy= 2.1972380481948903 , dayAnalyzed= 5.0 }
OutputHouseEvent { dailyHouseEntropy= 1.998213293863987 , dayAnalyzed= 6.0 }
OutputHouseEvent { dailyHouseEntropy= 2.0419419845991698 , dayAnalyzed= 7.0 }
OutputHouseEvent { dailyHouseEntropy= 2.171824011925842 , dayAnalyzed= 8.0 }
OutputHouseEvent { dailyHouseEntropy= 2.4758180547842428 , dayAnalyzed= 9.0 }
OutputHouseEvent { dailyHouseEntropy= 2.126700611541789 , dayAnalyzed= 10.0 }
```

Figura 5.17: Esempio di risultati giornalieri di ISVIS.

Nello specifico il risultato che viene mostrato a video è la sequenza di inserimenti nei tre flussi citati in precedenza. Dapprima ci sarà un inserimento in *NewSensorStatus*, che rappresenta una nuova configurazione trovata, poi un inserimento in *SensorConfiguration*

relativo alle informazioni complessive della configurazione precedente a quella appena inserita in *NewSensorStatus.* Successivamente abbiamo per ultimo il valore dell'entropia oraria calcolata rispetto all'ultimo evento inserito in *SensorConfiguration*, ed inserito nel flusso *HouseEntropy*. L'istante preso come riferimento sarà quello in cui termina la configurazione più recente e dalla quale verrà calcolata la finestra di 3600 secondi valevole per l'ultima ora. Un esempio di risultato è mostrato nella Figura 5.18, esso rappresenta l'inizio dell'analisi con le prime entropie calcolate.

```
Event received: NewSensorStatus{status=0.0, previousStatus=0.0, timestamp=1.0, countFactor=0}
Event received: NewSensorStatus{status=32768.0, previousStatus=0.0, timestamp=11.0, countFactor=0}
Event received: SensorConfiguration{status=0.0, durationEvent=10.0, initInstant=1.0, finInstant=10.0}
Event received: HouseEntropy { entropy= 0.0 , timestamp= 10.0 }
Event received: NewSensorStatus{status=0.0, previousStatus=0.0, timestamp=12.0, countFactor=0}
Event received: SensorConfiguration{status=32768.0, durationEvent=1.0, initInstant=11.0, finInstant=11.0}
Event received: HouseEntropy { entropy= 0.4394969869215134 , timestamp= 11.0 }
Event received: NewSensorStatus{status=8.0, previousStatus=0.0, timestamp=41.0, countFactor=0}
Event received: SensorConfiguration{status=0.0, durationEvent=29.0, initInstant=12.0, finInstant=40.0}
Event received: HouseEntropy { entropy= 0.16866093149667025 , timestamp= 40.0 }
```

Figura 5.18: Esempio di risultati mostrati a video.

In questo modo ISVIS è in grado di fornire uno strumento per l'analisi dello stile di vita di una persona monitorata all'interno di una struttura abitativa domotica. Collezionare un corposo quantitativo di dati permetterà infatti di definire i caratteri abitudinari nello stile di vita nel breve e nel lungo termine e di impiegare queste informazioni per il riconoscimento di deviazioni che segnalino improvvisi cambiamenti dovuti a problemi oppure degradazioni a lungo termine, a segnale dell'insorgenza di patologie. Come per gli indicatori precedenti, questo modello può essere applicato in contemporanea ad altri indicatori per estrarre allo stesso tempo informazioni differenti ma che possano essere combinate nell'ottenere informazioni complesse e ricche semanticamente per l'analisi.

Capitolo 6

Risultati sperimentali

In questo capitolo vengono discussi i risultati ottenuti processando il flusso di dati in ingresso al sistema tramite gli indicatori sviluppati con Esper nel capitolo precedente. Inizialmente sono confrontate le caratteristiche delle configurazioni dei dataset adottati e come queste possano influire sui risultati (Sezione 6.1) mentre nella seconda parte sono illustrati i risultati (Sezione 6.2). In ultimo (Sezione 6.3), per le attività complesse estratte dagli indicatori viene applicato un processo di validazione che permette di valutare la bontà dei risultati in termini di precisione e recupero (*precision* e *recall*).

6.1 Configurazioni adottate

In questa sezione, sono discusse le configurazioni adottate allo scopo di ottenere risultati significativi. In primo luogo, per lo sviluppo degli indicatori, è stato adottato il dataset ARAS. Tramite un cosiddetto Generatore di Eventi è stato possibile simulare il monitoraggio di due persone per 30 giorni all'interno della struttura abitativa *HouseA*. Lavorando in questo modo è stato possibile realizzare gli indicatori presentati nei capitoli precedenti utilizzando i 20 sensori installati nell'abitazione. Tuttavia i risultati ottenuti presentano una grande limitazione: l'analisi dei dati provenienti dai sensori non permette di distinguere quali siano le attività svolte da una piuttosto che dall'altra persona. Pertanto il risultato ricavato è una commistione delle attività svolte da entrambi gli inquilini nello stesso momento. Per questo motivo ARAS, nella tesi, viene adottato solo per la costruzione degli indicatori.

Per superare questa situazione, che non rispecchia il reale obbiettivo del progetto BRIDGe, è stato infatti adottato per verificare la bontà dei risultati ottenuti il dataset di dati simulati. A questo scopo quindi è stato impiegato SHARON, nella sua estensione

introdotta nella Sezione 3.3.2, perché simulasse la vita di una persona sola nell'ambiente abitativo *HouseA* per 90 giorni e ne fornisse le informazioni a livello di stato di sensori. Inoltre per la simulazione sono stati configurati i seguenti cambiamenti comportamentali (detti *drift*, che modificano le abitudini):

a) il tempo trascorso dormendo diventa via via più irregolare,
b) le attività complesse, come guardare la televisione, fare la doccia, pranzare e cenare, qualora svolte, richiedono crescente tempo.

Il dataset prodotto al termine di questa simulazione è costituito da due elementi, una struttura in cui i dati non presentano cambiamenti comportamentali (che prende il nome di *NoDrift*) ed una in cui viene svolta la stessa routine ma in presenza di cambiamenti comportamentali (*Drift*).

Attraverso l'impiego di questo dataset pertanto è possibile provare il corretto funzionamento degli indicatori sviluppati: monitorare la vita di una persona che vive da sola, e riconoscere come si modifichi il suo stile di vita tramite i cambiamenti comportamentali introdotti nella fase di configurazione. Questo secondo aspetto permette di validare il corretto funzionamento degli indicatori riscontrando le variazioni nelle attività secondo le configurazioni inserite. In questo capitolo pertanto sono presentati solo i risultati ottenuti con i dati simulati *Drift* e *NoDrift* perché più utili nel verificare il corretto funzionamento degli indicatori (permettendo di riconoscere i cambiamenti comportamentali) rispetto ai dati di ARAS che talvolta, in fase di sviluppo, ha generato risultati ambigui o molto difficili da interpretare.

6.2 Risultati

In questa sezione vengono illustrati i risultati ottenuti applicando l'analisi degli indicatori sviluppati sul dataset di dati simulati su entrambe le collezioni *NoDrift* e *Drift*. Tali risultati sono collezionati direttamente dagli indicatori in appositi file testuali di destinazione i quali sono impiegati per la costruzione di grafici che confrontino le due situazioni: svolgimento della stessa routine in assenza ed in presenza di cambiamenti comportamentali (Sezione da 6.2.1 a 6.2.5). Inoltre possono essere utilizzati anche per la valutazione di precisione e recupero (Sezione 6.2.6), misure impiegate nella teoria dell'informazione per valutare la corretta identificazione di informazioni. Nelle sezioni

seguenti sono introdotti i risultati per i cinque indicatori proposti in Capitolo 4 e Capitolo 5, mantenendo l'ordinamento con cui erano stati presentati.

6.2.1 Periodi giornalieri di riposo

La Figura 6.1 mostra graficamente le ore totali per singolo giorno in cui il sensore del letto ha rilevato una pressione su di esso. Il valore, calcolato giornalmente da ITASS (Sezione 4.2), può essere approssimato come stima del tempo trascorso dalla persona a letto riposando. Nella figura sono mostrati due andamenti: quello in assenza di cambiamenti comportamentali (tratto nero), e quello in presenza (tratto blu). Il grafico mostra chiaramente la differenza che intercorre fra le due situazioni. Per quanto riguarda i risultati senza drift, è facile riscontrare come vi sia periodicità nell'andamento e, soprattutto, non vi siano grandi discostamenti di valori. In presenza di drift, invece, la perdita di periodicità e costanza si manifesta progressivamente con il passare dei giorni: così il tratto blu si dimostra molto più frastagliato ed irregolare dal ventesimo giorno di analisi, con vistosi picchi sia in positivo che in negativo. Dal totale giornaliero trascorso dormendo viene calcolato sull'intera finestra analizzata la media di ore trascorse giornalmente dormendo. In questo caso la perdita di regolarità, causata dall'introduzione di cambiamenti comportamentali, porta ad un leggero incremento del tempo medio trascorso dormendo: nello specifico si passa dalle 9.0535 ore medie in assenza di cambiamenti comportamentali, alle 9.4233 in presenza di essi.

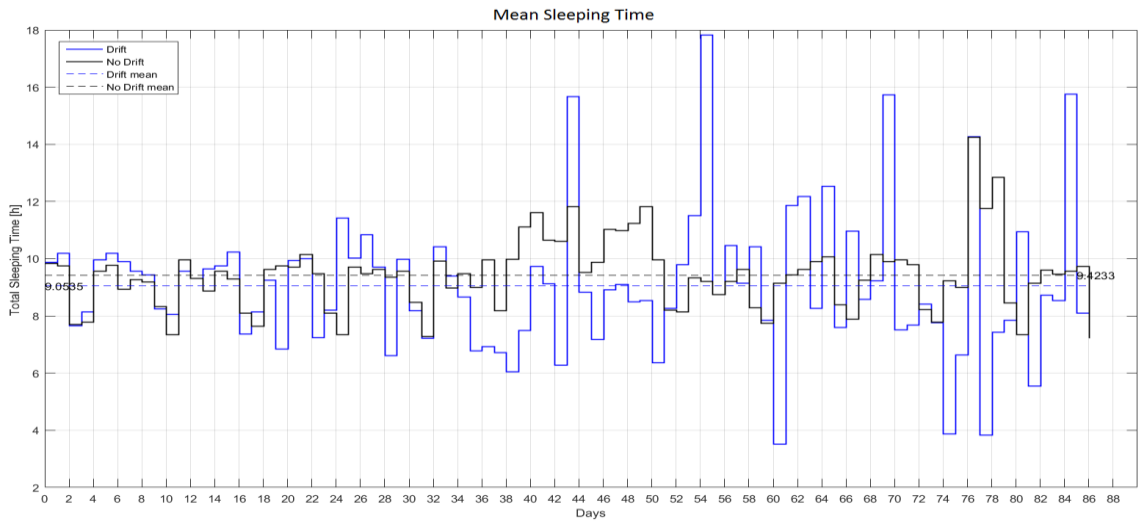

Figura 6.1: Grafico del tempo di riposo per 87 giorni.

6.2.2 Stime del consumo domestico di acqua

La Figura 6.2 raccoglie le informazioni riguardo il consumo di acqua giornaliero (espresso in litri) rilevato attraverso la valutazione delle informazioni provenienti dal singolo sensore della cabina della doccia. Dall'analisi dei risultati estratti da ISCS (Sezione 4.3) per il sanitario, si evince come, in assenza di cambiamenti comportamentali (tratto nero), l'acqua consumata giornalmente nella doccia assuma dei valori molto simili periodicamente. In presenza di cambiamenti comportamentali (tratto blu) si osserva come vi sia un progressivo aumento del consumo di acqua dal giorno 22 al 58 fino a stabilirsi attorno ai 60 litri di massimo. Questa variazione è dovuta alla configurazione inserita in SHARON che distende il tempo in cui l'attività "fare la doccia" viene eseguita. L'incremento del consumo giornaliero viene messo in luce anche dal calcolo della media per singolo andamento: in assenza di drift la stima di consumo giornaliero è di 21.9749 litri, mentre in presenza di drift questo valore sale fino ai 25.2628 litri.

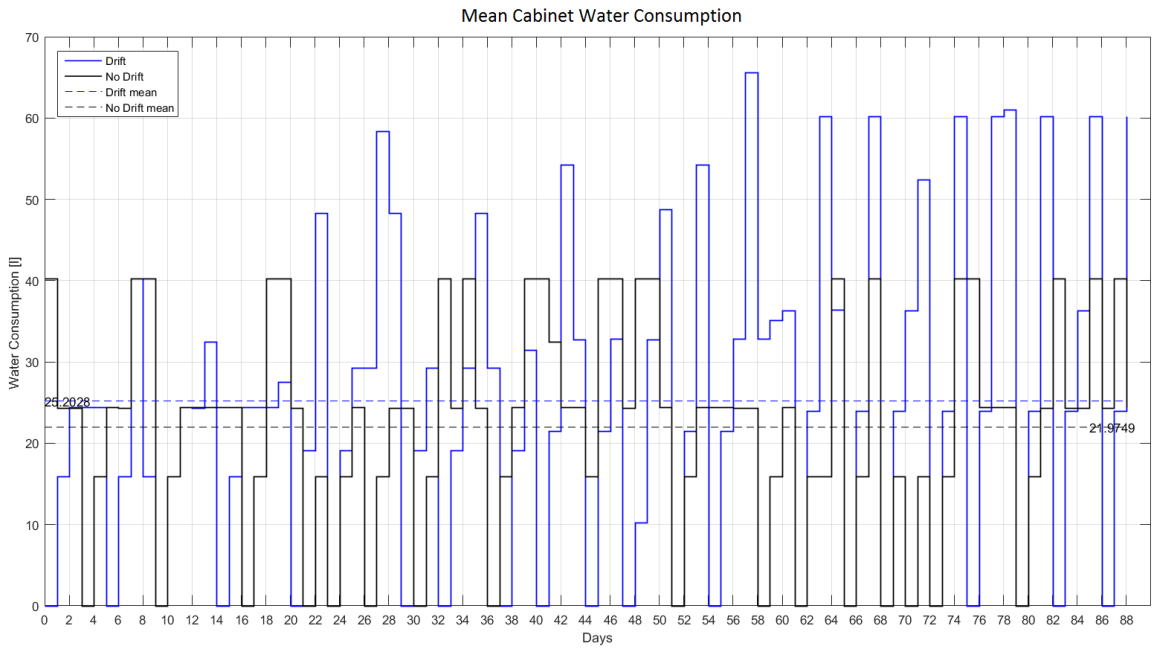

Figura 6.2: Grafico del consumo di acqua rilevato dalla doccia in 88 giorni.

Il risultato appena trovato è inserito in un contesto in cui ISCS considera i contributi di tutti i sensori relativi ai sanitari, allo scopo di ottenerne il consumo globale nell'appartamento analizzato. Pertanto, al pari del grafico appena valutato, possono essere estratti anche quelli del rubinetto del lavandino e dello sciacquone: gli unici sensori che

permettono di valutare il consumo di acqua nell'appartamento *HouseA*. Le Figure 6.3 e 6.4 illustrano la variazione del consumo quotidiano nel caso in cui sia presente un cambiamento comportamentale (in blu) o meno (in nero). Per quanto concerne il rubinetto (Figura 6.3) vi è un chiaro incremento del consumo (seppur non in maniera costante), riflesso dell'incremento dell'attività della doccia; per quanto riguarda invece lo scarico, questo non presenta variazioni degne di nota (benchè i due andamenti procedano su livelli di consumo differenti), poiché la routine dell'abitante per questo aspetto non viene modificata.

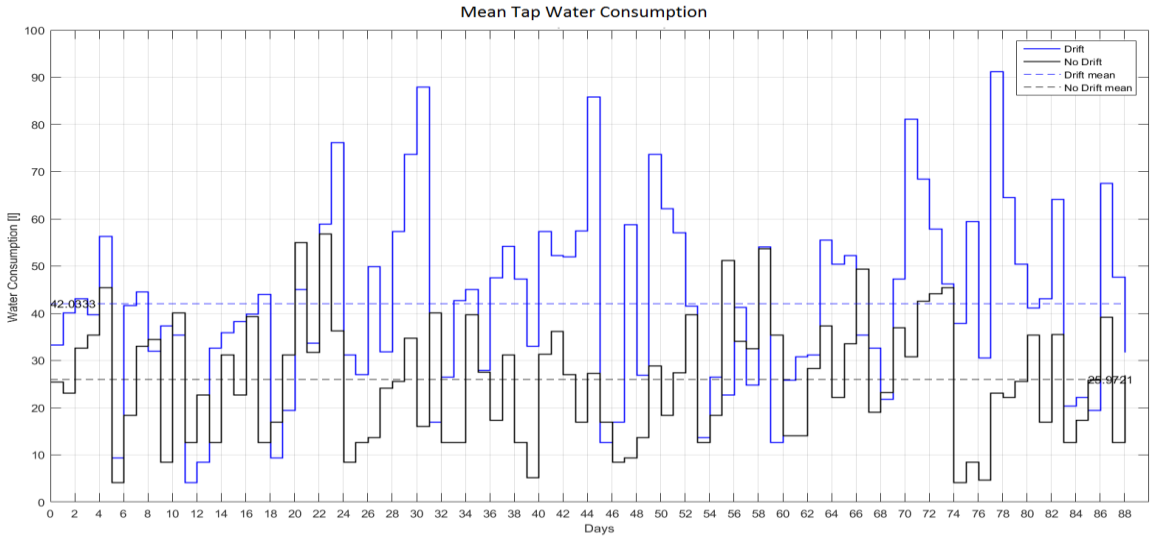

Figura 6.3: Grafico del consumo di acqua rilevato dal rubinetto su 88 giorni.

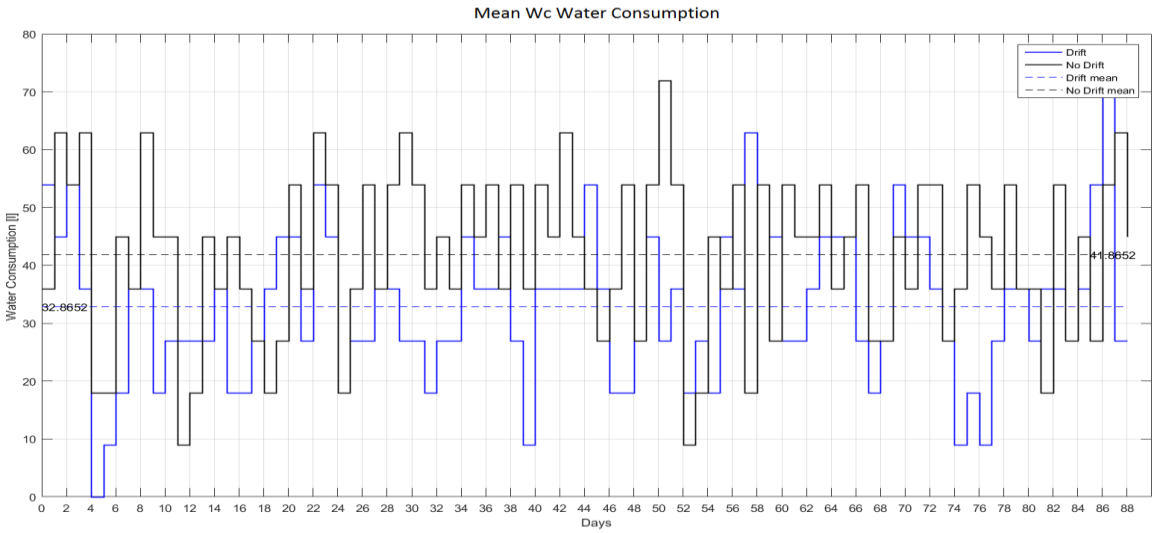

Figura 6.4: Grafico del consumo di acqua rilevato dallo scarico su 88 giorni.

Infine è interessante vedere come gli andamenti rappresentati nelle figure precedenti caratterizzino anche l'andamento del consumo totale giornaliero di acqua nell'appartamento in presenza o in assenza di cambiamenti comportamentali. La Figura 6.5 esemplifica questa situazione sempre grazie ai risultati calcolati da ISCS. In particolare si può osservare chiaramente come i cambiamenti comportamentali (tratto blu) riguardo i consumi di doccia e rubinetto provochino un aumento graduale dei consumi giornalieri per l'abitazione complessiva rispetto alla situazione (in nero) in cui tali cambiamenti non sono previsti. Questa variazione si riflette inoltre nel grande divario che viene a crearsi fra le medie delle due situazioni: in presenza di cambiamenti il consumo medio giornaliero per la casa intera si attesta a 112.1676 litri, mentre, in assenza di drift, si ferma a 76.4274 litri.

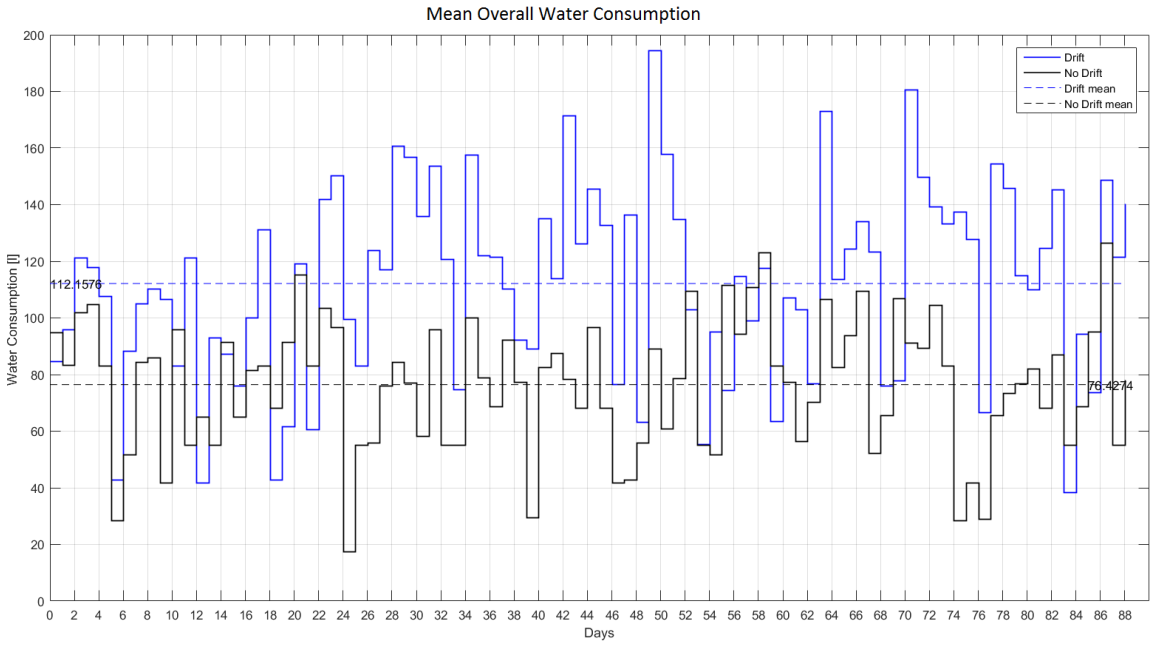

Figura 6.5: Grafico del consumo generale di acqua nell'intera abitazione per gli 88 giorni di analisi monitorati.

6.2.3 Entropia giornaliera per il sensore del letto

La Figura 6.6 mostra invece i risultati ottenuti in termini di entropia misurata sul singolo sensore con ISVUS (Sezione 5.1). In questo caso è stato utilizzato il sensore del letto. L'entropia giornaliera misura il grado di prevedibilità o imprevedibilità dell'attività dell'abitante della casa in relazione al sensore del letto. Il profilo di andamento, in

assenza di cambiamenti comportamentali (tratto nero), dimostra una forte periodicità, come d'altronde espresso nella costanza dell'attività e già visto nella stessa condizione nella Figura 6.1. Il valore attorno cui si attesta è 0.9539, una media alta poiché significa che nelle 24 ore (per le quali avviene la valutazione dell'entropia) la probabilità di rilevazione di attività o inattività da parte del sensore è prossima all'equità fra i due casi (si ricorda che 1 è il valore di equiprobabilità). Nella situazione invece in cui sono presenti cambiamenti comportamentali (tratto blu) il profilo si dimostra molto più frastagliato, conseguenza dell'irregolarità nel riposare introdotta nel comportamento dell'abitante della casa. Il valore medio perde di quota in questo caso, 0.88989, poiché risente del calo delle ore di sonno (quindi una probabilità di rilevazione di attività inferiore) e della maggiore irregolarità nello svolgere questa attività. Ne risultano quindi dei vistosi minimi di imprevedibilità.

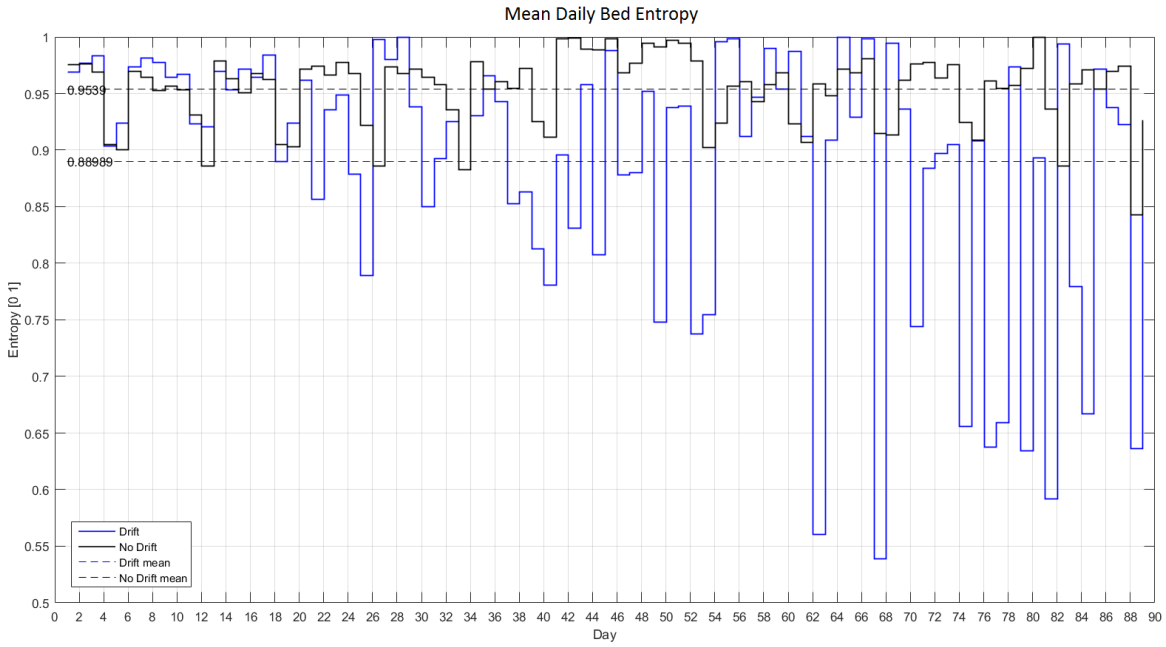

Figura 6.6: Grafico dell'entropia per il sensore del letto su 88 giorni di analisi.

6.2.4 Activity Time Ratio giornaliero per l'attività "guardare la televisione"

La Figura 6.7 rappresenta la valutazione dell'Activity Time Ratio giornaliero rilevato. IRAC (Sezione 5.2) riconosce per ogni giorno di analisi quali sono gli istanti in cui la

persona monitorata sta guardando la televisione e, al termine di ogni giornata, calcola un valore (compreso fra 0 e 1) che esprime il tempo trascorso sulle 24 ore svolgendo questa attività. Il profilo nero rappresenta l'andamento giornaliero in assenza di cambiamenti comportamentali: come si può notare, la struttura presenta una discreta periodicità e, soprattutto, i valori non eccedono mai particolarmente. Il profilo blu invece rappresenta il comportamento riscontrato quando lo svolgimento dell'attività presenta drift. Il grafico permette di confermare la configurazione introdotta che prevedeva un allungamento del tempo dedicato a guardare la televisione. L'aumento infatti avviene progressivamente e innalza di molto il valore medio calcolato su 89 giorni di analisi: da 0.068319 si sale significativamente fino a 0.15953. In questo modo viene verificata l'entità della variazione di abitudine configurata per SHARON.

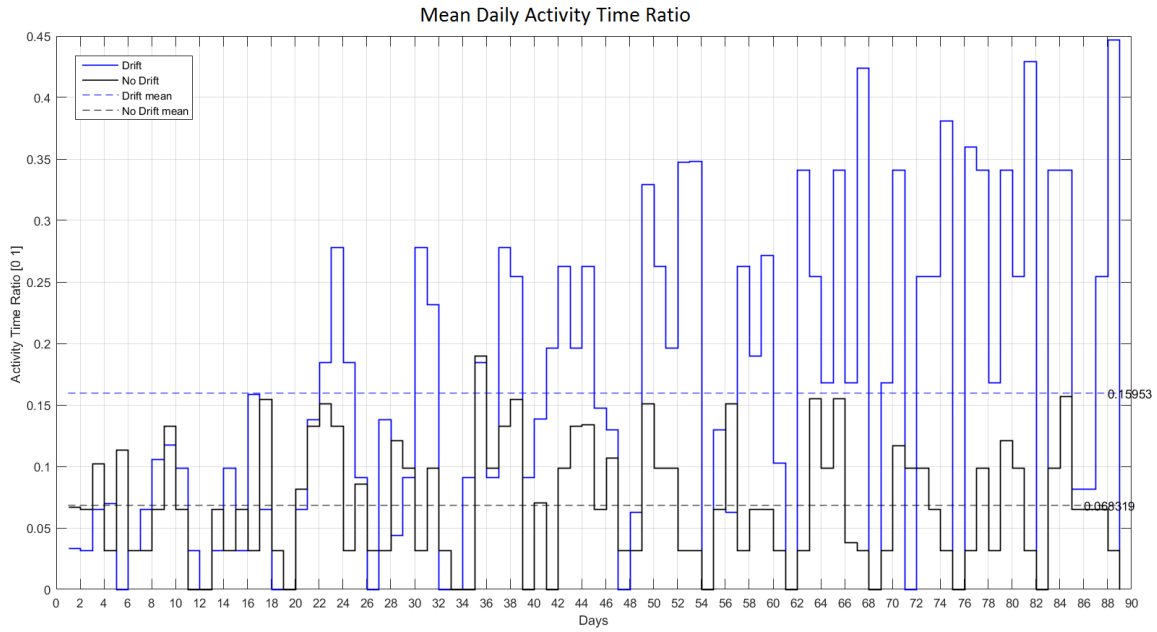

Figura 6.7: Grafico dell'Activity Time Ratio giornaliero per 89 giorni analizzati.

6.2.5 Entropia giornaliera per l'intera abitazione

La Figura 6.8 rappresenta l'informazione estratta tramite l'ultimo degli indicatori sviluppati, ISVIS: l'entropia giornaliera generale della casa calcolata impiegando in contemporanea tutti i sensori installati nell'abitazione. Rispetto alla situazione per il singolo sensore, in questo caso, l'entropia rappresenta una misura dell'imprevedibilità della configurazione generale della casa (data da ciascuna combinazione possibile di stato

dei sensori). Il profilo in nero rappresenta il caso in cui nel sistema di simulazione non siano stati inseriti dei cambiamenti comportamentali, quello in blu invece li prevede. Come si può notare, nel passaggio da una situazione all'altra, non ci sono delle differenze marcanti poiché le variazioni di comportamento introdotte non influiscono sul numero di attività svolte e quindi non portano ad una variazione del livello di prevedibilità o imprevedibilità della configurazione generale dell'appartamento. Sarebbe stata possibile una variazione riscontrabile nell'andamento del grafico delle due situazioni qualora si fosse configurato lo svolgimento di un numero maggiore o minore di attività nel singolo giorno.

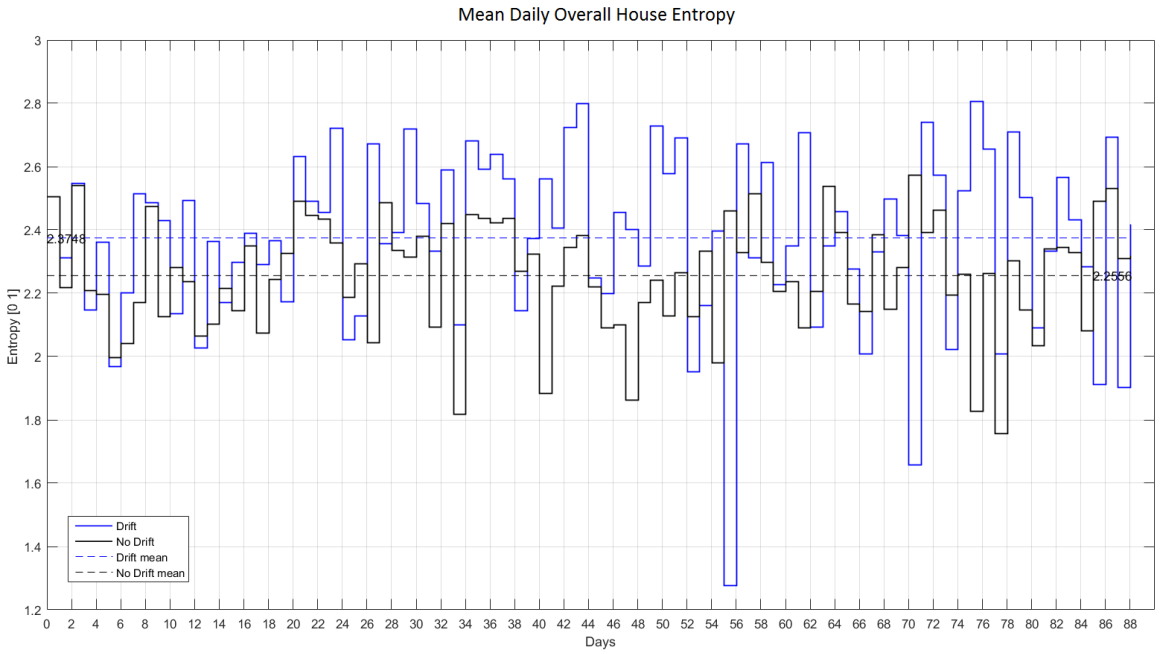

Figura 6.8: Grafico dell'entropia per tutti i sensori della casa su 88 giorni di analisi.

6.2.6 Precisione e recupero

Un discorso a parte deve essere effettuato per le informazioni estratte tramite l'indicatore presentato nella Sezione 4.5. Come si è visto, ISVIS permette di estrarre dall'interazione di un gruppo finito di sensori un'attività complessa. Nell'esempio studiato è stata riconosciuta l'attività "guardare la televisione" sulla base delle relazioni fra le attivazioni rilevate dai sensori del telecomando e quelle delle sedie e dei divani. I risultati prodotti sono i periodi giornalieri durante i quali la persona monitorata svolge tale attività. Come

abbiamo visto, l'Activity Time Ratio permette di calcolare quanto tempo venga trascorso guardando la televisione nelle 24 ore che costituiscono una giornata. Tuttavia lo scopo principale di questo indicatore è quello di identificare l'attività "guardare la televisione" correttamente, ossia quando essa venga realmente svolta dalla persona monitorata. Per misurare il grado di correttezza dei risultati ottenuti si utilizzano le annotazione messe a disposizione dai dataset adottati. Esse vengono confrontate giornalmente istante per istante con i valori restituiti da ISVIS allo scopo di valutare se il riconoscimento dell'attività è avvenuto correttamente o meno. Per misurare il grado di correttezza, la teoria dell'informazione dispone di due misure:

- *Precisione*: misura, sul tempo totale (espresso in secondi) delle attività "guardare la tv" individuate giornalmente dall'indicatore, la frazione di secondi giornalieri in cui l'attività è stata riconosciuta correttamente.
- *Recupero*: misura, sul tempo totale (espresso in secondi) delle attività "guardare la tv" definito dalle annotazioni del dataset, la frazione di secondi giornalieri in cui l'attività è stata individuata correttamente.

In particolare la *precisione* misura con un valore da 0 a 1 il grado di correttezza giornaliero di identificazione dell'attività rispetto alle annotazioni espresse; '1' rappresenta il valore massimo di precisione, 0 quello minimo. Il *recupero* usa la stessa scala di valori per misurare, invece, l'identificazione di attività fra le sole annotazioni di guardare la televisione custodite dal dataset nella ventunesima colonna. La Tabella 6.1 mostra le possibili casistiche del confronto: le colonne rappresentano gli stati risultati dall'indicatore, quindi se in un secondo specifico si stia guardando la televisione (prima colonna) oppure no (seconda colonna); le righe invece riportano le annotazioni per il secondo in analisi, e quindi se per il dataset si stia guardando la televisione (prima riga) oppure no (seconda riga).

		Risultato indicatore	
		TV	NO TV
Annotazione dataset	TV	TP	FN
	NO TV	FP	TN

Tabella 6.5: Casistica dei risultati restituiti.

Il confronto fra gli stati istantanei restituiti dall'indicatore e quelli riportati nel dataset possono presentare quattro tipologie di casistiche necessarie per la valutazione di precisione e recupero e riportati sempre nella tabella precedente. Essi sono:

- *True Positive (TP)*: sono tutti i secondi in cui ISVIS ha correttamente identificato lo svolgimento dell'attività guardare la televisione.
- *False Negative (FN)*: sono tutti i secondi in cui ISVIS non ha riconosciuto che la persona monitorata stesse guardando la televisione, stato dichiarato con un'apposita annotazione per il dataset.
- *False Positive (FP)*: sono tutti i secondi nei quali ISVIS ha riconosciuto lo svolgimento dell'attività guardare la televisione, ma nella realtà delle annotazioni del dataset non lo erano.
- *True Negative (TN)*: sono tutti i secondi in cui ISVIS ha riconosciuto che non si stesse guardando la televisione, e l'annotazione dà ragione.

Questi risultati sono impiegati nelle formule di precisione e recupero:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Le formule sono impiegate per ciascun giorno analizzato allo scopo di valutare la correttezza dei risultati ottenuti.

Le tabelle seguenti mostrano la valutazione di precisione e recupero per ciascuno degli 89 giorni analizzati dal dataset sintetico in cui viene svolta la stessa routine ma in assenza di cambiamenti comportamentali (Tabella 6.2) ed in presenza di essi (Tabella 6.3) .

Day	Precision	Recall	Day	Precision	Recall	Day	Precision	Recall
1	1	0,949346	31	1	0,93268	61	NaN	NaN
2	1	0,924673	32	1	0,9	62	1	0,9
3	1	0,933228	33	NaN	NaN	63	0,868685	0,949268
4	1	0,9	34	NaN	NaN	64	1	0,93268
5	0,591006	0,949183	35	0,892385	0,949157	65	0,868666	0,949106
6	1	0,9	36	1	0,93268	66	1	0,9
7	1	0,9	37	1	0,93701	67	1	0,9
8	1	0,925	38	0,868106	0,949101	68	NaN	NaN
9	1	0,936846	39	NaN	NaN	69	1	0,9
10	1	0,925	40	0,477318	0,94902	70	0,859002	0,94902
11	NaN	NaN	41	NaN	NaN	71	1	0,932898
12	NaN	NaN	42	1	0,93268	72	1	0,93268
13	1	0,924673	43	1	0,937173	73	1	0,924673
14	1	0,9	44	1	0,937136	74	1	0,9
15	1	0,924673	45	1	0,924673	75	NaN	NaN
16	1	0,9	46	0,629349	0,948856	76	1	0,9
17	0,868126	0,949265	47	1	0,9	77	1	0,933115
18	1	0,9	48	1	0,9	78	1	0,9
19	NaN	NaN	49	0,890413	0,949265	79	0,831568	0,949237
20	0,821807	0,94951	50	1	0,93268	80	1	0,93268
21	1	0,936846	51	1	0,933115	81	1	0,9
22	0,890421	0,949346	52	1	0,9	82	NaN	NaN
23	1	0,937092	53	1	0,9	83	1	0,932898
24	1	0,9	54	NaN	NaN	84	0,86979	0,949275
25	0,39222	0,94902	55	1	0,925	85	1	0,924673
26	1	0,9	56	0,890413	0,949265	86	1	0,925
27	1	0,9	57	1	0,9	87	1	0,924346
28	0,831536	0,94902	58	1	0,924346	88	1	0,9
29	1	0,932898	59	1	0,924346	89	NaN	NaN
30	1	0,9	60	1	0,9			

Tabella 6.6: Precisione e recupero per gli 89 giorni di analisi in assenza di drift.

Day	Precision	Recall	Day	Precision	Recall	Day	Precision	Recall
1	1	0,948693	31	1	0,939624	61	NaN	NaN
2	1	0,9	32	NaN	NaN	62	1	0,937309
3	1	0,924673	33	NaN	NaN	63	1	0,933079
4	0,482072	0,94902	34	1	0,924765	64	1	0,924873
5	NaN	NaN	35	1	0,937148	65	1	0,937246
6	1	0,9	36	1	0,925	66	1	0,924873
7	1	0,924673	37	1	0,941275	67	0,824911	0,962055
8	0,669904	0,974444	38	0,919733	0,949484	68	NaN	NaN
9	0,85903	0,949237	39	1	0,924765	69	1	0,924873
10	1	0,933115	40	0,674648	0,949178	70	1	0,937246
11	1	0,9	41	1	0,933113	71	NaN	NaN
12	NaN	NaN	42	1	0,93717	72	1	0,933164
13	1	0,9	43	1	0,933113	73	1	0,933079
14	1	0,932898	44	1	0,937252	74	0,907045	0,949714
15	1	0,9	45	0,901324	0,949587	75	NaN	NaN
16	0,871412	0,949286	46	1	0,924835	76	0,95951	0,949714
17	1	0,924673	47	NaN	NaN	77	1	0,937436
18	NaN	NaN	48	1	0,9	78	1	0,925
19	NaN	NaN	49	1	0,939802	79	1	0,937309
20	1	0,924673	50	1	0,937417	80	1	0,933079
21	1	0,933177	51	1	0,933113	81	0,815589	0,962182
22	1	0,937148	52	0,958045	0,949571	82	NaN	NaN
23	1	0,941275	53	0,774728	0,961922	83	1	0,937309
24	1	0,937148	54	NaN	NaN	84	1	0,937309
25	1	0,924765	55	1	0,924835	85	1	0,9
26	NaN	NaN	56	1	0,9	86	1	0,9
27	1	0,93302	57	1	0,937252	87	1	0,933079
28	1	0,9	58	1	0,931001	88	0,782648	0,962055
29	1	0,924531	59	1	0,938463	89	NaN	NaN
30	1	0,941432	60	0,648411	0,949505			

Tabella 6.7: Precisione e recupero per gli 89 giorni di analisi in presenza di drift.

Le Tabelle 6.2 e 6.3 raccolgono i cvalori di precisione e recupero per lo svolgimento delle stesse routine di attività in assenza di cambiamenti comportamentali (la prima) ed in presenza di questi (la seconda). I risultati calcolati per entrambe le situazioni permettono di valutare positivamente l'operato dell'indicatore nella procedura di riconoscimento dell'attività "guardare la televisione" impiegando semplicemente le informazioni proveniente da un gruppo ristretto di cinque sensori. Per quanto riguarda la precisione, in ambo le tabelle essa assume valore 1 nella maggioranza dei giorni analizzati (59 su 89), il che significa che, per ciascuno di tali giorni in cui questa misura restituisce tale valore, ogni secondo identificato come guardare la tv, lo è effettivamente. Questo risultato permette di affermare che nella maggioranza dei casi, sia in presenza sia in assenza di cambiamenti comportamentali, le attività trovate sono state identificate correttamente. Questo risultato viene rafforzato se si osserva che anche i valori restanti sono tutti prossimi al massimo, solamente due giorni su 89 nella Figura 6.2 presentano una precisione inferiore allo 0.5 (risultato scarso), mentre nella Figura 6.3 è uno solo. Laddove la precisione assume valore *NaN* è perché in quel giorno specifico non è

presente alcuna attività "guardare la televisione" e, giustamente, non ne è stata riconosciuta alcuna occorrenza dal sensore.

Per quanto concerne il recupero, si nota come questa, misura in entrambe le situazioni rappresentate dalle tabelle, non sia mai inferiore a 0,9 questo significa che la maggioranza di secondi giornalieri del dataset, in cui viene indicata l'attività guardare la televisione, viene riconosciuta correttamente da ISVIS. Anche in questo caso il valore NaN rappresenta la corretta assenza di rilevazioni dell'attività scelta per il singolo giorno.

Precisione e recupero in questa analisi permettono quindi di validare i risultati ottenuti dall'indicatore confrontandoli con quanto dichiarato nel dataset. La verifica infatti permette di misurare la correttezza dei risultati estratti da ISVIS. Sia la precisione che il recupero presentano valori molto alti in entrambe le situazioni, questo permette di definire l'affidabilità e la robustezza dell'indicatore che restituisce risultati credibili e prossimi alla correttezza sia in condizioni abitudinarie di vita che in quelle in cui sono presenti vistose variazioni in tali comportamenti. Concludere una valutazione così positiva per un indicatore complesso come questo permette di poter aprire la strada al riconoscimento in tempo reale di altre tipologie di attività complesse tramite i soli dati dei sensori installati in un'abitazione intelligente.

Il processo di validazione svolto in questo capitolo permette quindi di attribuire valore agli indicatori sviluppati riconoscendone l'efficacia per gli obbiettivi per i quali sono stati costruiti e quindi garantendo per i risultati calcolati.

Capitolo 7

Conclusioni e lavori futuri

7.1 Conclusioni

In questa tesi sono stati proporti degli approcci utili per effettuare analisi di eventi a tempo reale all'interno di un sistema Ambient Assisted Living. Le informazioni, estratte dai flussi di dati in ingresso tramite indicatori quantitativi sviluppati usando la sintassi EPL di Esper, possono essere impiegate per valutare condizioni di vita ed abitudini della persona monitorata all'interno di una *Smart Home*. L'obbiettivo di questo sistema è quello di supportare la vita di una persona anziana che vive da sola fornendo, a chi se ne prende cura, una collezione varia di informazioni in tempo reale riguardo la routine tramite rilevazioni con sensori, non invasivi, installati nell'abitazione intelligente.

In una situazione sperimentale come questa, in assenza di una struttura abitativa da monitorare, la tesi ha ricercato dataset disponibili pubblicamente che presentassero caratteristiche utili per gli obbiettivi del progetto BRIDGe. A questo scopo è stato adottato il dataset prodotto dal progetto ARAS poiché offre una buona copertura temporale e la struttura abitativa *HouseA* descrive un ambiente ideale per il progetto BRIDGe.

In seguito, per poter costruire un sistema di indicatori quantitativi che analizzano tali dati, è stato necessario sviluppare un Generatore di Eventi: una routine che, leggendo ordinatamente il dataset scelto, mandasse tali dati in ingresso al sistema perché venissero analizzati. Una volta sviluppato questo strumento fondamentale è stato possibile costruire degli indicatori quantitativi tramite interrogazioni EPL, linguaggio dichiarativo di Esper, simile al ben più noto SQL.

Lo scopo principale di questa tesi è stato di estrarre in tempo reale varie tipologie di informazioni riguardo le attività quotidiane e le abitudini, impiegando esclusivamente i dati in ingresso al sistema come provenienti da sensori binari installati nell'abitazione

HouseA adottata. Le informazioni proposte sono cinque e coprono diversi aspetti della vita di una persona: da quelli più semplici e concreti come la valutazione delle stime giornaliere delle ore di riposo e dei consumi domestici di acqua, ad aspetti più astratti come il riconoscimento dello svolgimento di attività complesse ed infine la valutazione dell'imprevedibilità dello stile di vita, misurando l'entropia sia per un singolo sensore che per tutti i sensori installati nell'abitazione.

Per mostrare la correttezza delle informazioni estratte è stato adottato in un secondo momento un dataset sviluppato all'interno del progetto BRIDGe tramite l'impiego di un sistema, detto SHARON, in grado di simulare la sequenzialità di attività giornaliere di una persona nell'abitazione ARAS considerata. La configurazione di questo sistema permette, tra l'altro, l'introduzione di cambiamenti comportamentali (detti *drift*) nello stile di vita, così da modificare nel tempo le abitudini dell'abitante simulato. Pertanto è stato possibile estrarre, tramite gli indicatori sviluppati, le informazioni per entrambe le situazioni: in presenza ed in assenza di drift. Il confronto di tali risultati ottenuti ha permesso di mettere in luce i cambiamenti comportamentali introdotti in fase di configurazione. In questo modo viene fornito uno strumento utile anche in un sistema reale (e non più solo simulato) per la valutazione dello stile di vita e del benessere della persona monitorata. Verificare infatti il mantenimento o il deterioramento di comportamenti abitudinari può essere impiegato come utile misurazione delle condizioni e della qualità di vita e di salute di una persona fragile risiedente in un'abitazione controllata da un sistema AAL.

7.2 Lavori futuri

I risultati ottenuti da questa tesi possono essere impiegati come il punto di partenza per una varietà di lavori sia in termini di applicazione effettiva del sistema sviluppato che di estensione dell'analisi svolta.

Poiché il sistema presente è proposto in un ambiente di simulazione, l'applicazione di questo all'interno di un sistema che monitori una reale struttura abitativa apre le porte a un'ampia possibilità di miglioramenti. Innanzitutto applicare questo sistema in un contesto di vero monitoraggio a tempo reale richiede un lavoro di corretta configurazione perché il flusso di dati in ingresso sia correttamente fornito agli indicatori e la

connessione con i sensori sia robusta ed efficace. L'utilizzo inoltre del tempo reale, e non più di quello simulato dal Generatore di Eventi, può permettere un miglioramento nell'utilizzo di memoria da parte del sistema usando finestrature che ospitino i dati dei flussi in ingresso e che abbiano validità limitata nel tempo, cioè che espirino dopo il trascorrere di un tempo prestabilito.

Ogni sistema AAL può avere interessi differenti riguardo le informazioni da ricercare, a questo scopo il lavoro presente ha sviluppato dei modelli di indicatori che possano essere impiegati, anche come ispirazione, per questo tipo di ricerche per lavori futuri. Inoltre gli indicatori stessi sono stati creati in modo da essere combinati ed innestati fra di loro: il grado di combinazione di questi permette di ottenere altre informazioni o di accrescerne la ricchezza informativa.

Altre ricerche potranno lavorare adottando tipologie di sensori differenti, per esempio di temperatura, che misurino direttamente i consumi oppure anche indossabili. Questi infatti, pur nelle differenze specifiche, potranno essere impiegati per completare l'analisi, offrire nuovi punti di vista da considerare, oppure migliorare i risultati estratti.

Interessante sarebbe inoltre la possibilità di gestire, in caso di condizioni di simulazione (quando sono adottati dataset), la possibilità di confermare quando una persona svolge più azioni contemporaneamente. Pertanto, nel momento in cui sono costruiti i dataset, sarebbe auspicabile in futuro una revisione della procedura con cui vengono fornite le annotazioni delle attività svolte, ed in particolare permettere di definire se in un istante specifico la persona stia svolgendo una o più attività, e quali.

I risultati prodotti possono essere utilizzati inoltre per costruire e verificare sistemi di analisi continua delle informazioni estratte per la valutazione del loro andamento e per comunicare istantaneamente, tramite un sistema di messaggistica, quando vengono riscontrate anomalie di sistema o di comportamento oppure per garantire sulla regolarità nello svolgimento delle attività.

Appendice A

In questo capitolo sono raccolte tutte le interrogazioni che sono state sviluppate per gli esempi di indicatori sviluppati nel Capitolo 4. Il capitolo è organizzato dividendosi in Sezioni differenti a seconda dell'indicatore cui appartengono le interrogazioni: Sezione A1 per le interrogazioni di ITASS che rileva il livello di attività per il singolo sensore del letto (Capitolo 4, Sezione 4.2), Sezione A2 per le interrogazioni di ISCS volte a stimare il consumo domestico complessivo di acqua (Capitolo 4, Sezione 4.3).

A1 Interrogazioni ITASS

In questa sezione sono presentate in ordine le interrogazioni atte a processare i dati in analisi dal flusso in ingresso al sistema allo scopo di rilevare, ogni volta che un nuovo evento di attivazione del sensore del letto viene rilevato, il tempo complessivo, espresso in secondi, in cui il sensore ha rilevato attività nelle ultime 24 ore.

Query 1:

```
1  public static String movingEventsWindowDefinition =
2  "create window MovingEvents.win:length(2)
3   as select * from SensorStream"
```

Query 2:

```
1  public static String populatingMovingEvents =
2  "insert into MovingEvents
3   select *
4   from SensorStream
5   where sensorID=20"
```

Query 3:

```
public static String sleepingEventSingleDuration =
"insert into SleepingStream
 select (b.timestamp - a.timestamp) as duration,
         a.timestamp+1 as initActivation,
         b.timestamp as finActivation
 from pattern [every ((a = MovingEvents(status = 1))
             -> ((b = MovingEvents(status = 0))))]"
```

Query 4:

```
public static String
  totalSleepingCaseOfOnlyActivationsInside =
 "insert into TotalSleepingStream
  select (select sum(K.duration)
     from SleepingStream.win:keepall() as K
     where K.initActivation >= L.finActivation-86400)
     as totalDuration
  from SleepingStream.std:lastevent() as L
  where not exists(select *
     from SleepingStream.win:keepall() as S
     where S.initActivation < L.finActivation-86400
     and S.finActivation >= L.finActivation-86400)"
```

Query 5:

```
public static String
     totalSleepingCaseOfInitialActivation =
 "insert into TotalSleepingStream
  select 1+(select sum(K.duration)
     from SleepingStream.win:keepall() as K
     where K.initActivation >= L.finActivation-86400)
    +  (S.finActivation - (L.finActivation-86400))
    as totalDuration
  from SleepingStream.win:keepall() as S,
       SleepingStream.std:lastevent() as L
   where S.initActivation < L.finActivation-86400
     and S.finActivation >= L.finActivation-86400"
```

Query 6:

```
1 public static String sendResultToOutputFile =
2 "insert into OutputDailySleepingStream
3  select (select L.totalDuration
4       from TotalSleepingStream.std:lastevent() as L)
5       as totalDuration,
6         (select Math.floor((L.fin-43200)/86400)
7       from TotalSleepingStream.std:lastevent() as L)
8       as dayAnalyzed
9  from SleepingStream.std:lastevent() as S
10 where S.finActivation not in (select K.fin
11       from TotalSleepingStream.win:keepall() as K)
12   and Math.floor((S.finActivation-43200)/86400)
13       > (select Math.floor((L.fin-43200)/86400)
14       from TotalSleepingStream.std:lastevent() as L)"
```

A2 Interrogazioni ISCS

In questa sezione sono mostrate ordinatamente le interrogazioni atte a processare i dati in analisi dal flusso in ingresso al sistema allo scopo di rilevare, ogni volta che un nuovo evento di attivazione del sensore del letto viene rilevato, il consumo omplessivo complessivo di acqua nelle ultime 24 ore sia per la totalità della casa, sia per i singoli sensori adottati.

Query 7:

```
1 public static String waterEventsWindowDefinition =
2 "create window WaterMovingEvents.win:length(2)
3  as select * from SensorStream"
```

Query 8:

```
1  public static String populatingMovingEvents =
2  "insert into MovingEvents
3   select *
4   from SensorStream"
```

Query 9:

```
1  public static String
2  tapWaterSingleActivationConsumption =
3  "insert into TapStream
4   select a.sensorID as sensorID,
5        (b.timestamp - a.timestamp) as duration,
6        0.08*(b.timestamp-a.timestamp) as consumption,
7        a.timestamp+1 as initActivation,
8        b.timestamp as finActivation
9   from pattern [every (
10       (a = WaterMovingEvents(status = 1 and
11        sensorID=17)) -> (b = WaterMovingEvents
12       (status = 0 and sensorID = 17)))] "
```

Query 10:

```
1  public static String
2  cabinetWaterSingleActivationConsumption =
3  "insert into CabinetStream
4   select a.sensorID as sensorID,
5        (b.timestamp - a.timestamp) as duration,
6        0.133*(b.timestamp-a.timestamp) as consumption
7        a.timestamp+1 as initActivation,
8        b.timestamp as finActivation
9   from pattern [every (
10       (a = WaterMovingEvents(status = 1 and
11        sensorID=18)) -> (b = WaterMovingEvents
12       (status = 0 and sensorID = 18)))] "
```

Query 11:

```
1   public static String findWcActivationInstants =
2   "insert into ActivationInstant
3    select a.timestamp as instant,
4           a.sensorID as sensorID
5    from pattern [every(a = WaterMovingEvents
6                 (sensorID = 18 and status = 1))] "
```

Query 12:

```
1  public static String findWcDeactivationInstant =
2  "insert into DeactivationInstant
3   select a.timestamp  as instant, 1 as activated,
4          a.sensorID as sensorID
5   from pattern [every (a = WaterMovingEvents
6                (sensorID = 18 and status = 0))] "
```

Query 13:

```
public static String wcDefinitiveFallingInstant =
"insert into WcStream
 select 9 as consumption,
     D.instant as finActivation, 18 as sensorID
 from DeactivationInstant.std:lastevent() as D,
      Time.std:lastevent() as T
 where exists (select *
       from ActivationInstant.win:keepall() as A
       where A.instant < D.instant)
   and 0 = (select count(*)
       from ActivationInstant.win:keepall() as A
       where A.instant > D.instant
       and A.instant <= D.instant + 10 )
   and T.time > D.instant +10
   and not exists (select *
       from WcStream.win:keepall()
       where finActivation = D.instant)"
```

Query 14:

```
public static String
insertLastTapActivationIntoOverall =
"insert into TemporaryWaterStream
 select T.sensorID as sensorID,
        T.duration as duration,
        T.consumption as consumption,
        T.initActivation as initActivation,
        T.finActivation as finActivation
 from TapStream.std:lastevent() as T"
```

Query 15:

```
public static String
insertLastCabinetActivationIntoOverall =
"insert into TemporaryWaterStream
 select T.sensorID as sensorID,
        T.duration as duration,
        T.consumption as consumption,
        T.initActivation as initActivation,
        T.finActivation as finActivation
 from CabinetStream.std:lastevent() as T"
```

Query 16:

```
1  public static String
2  insertLastWcActivationIntoOverall =
3  "insert into TemporaryWaterStream
4   select T.sensorID as sensorID,
5          T.consumption as consumption,
6          T.initActivation as initActivation,
7          T.finActivation as finActivation
8   from WcStream.std:lastevent() as T"
```

Query 17:

```
1  public static String
2        findEventualInitialTapFragments =
3  "insert into WaterStreamFragmentation " +
4   select 0.08*(K.finActivation -
5          (L.finActivation-86400)+1)
6          as fragmentation,
7          L.finActivation as timestamp,
8          K.sensorID as sensorID
9   from TemporaryWaterStream.win:keepall() as K,
10       TemporaryWaterStream.std:lastevent() as L
11  where K.initActivation < L.finActivation - 86400
12    and K.finActivation >= L.finActivation - 86400
13    and K.sensorID = 17 "
```

Query 18:

```
1  public static String
2        findEventualInitialCabinetFragments =
3  "insert into WaterStreamFragmentation " +
4   select 0.133*(K.finActivation -
5          (L.finActivation-86400)+1)
6          as fragmentation,
7          L.finActivation as timestamp,
8        K.sensorID as sensorID
9   from TemporaryWaterStream.win:keepall() as K,
10       TemporaryWaterStream.std:lastevent() as L
11  where K.initActivation < L.finActivation - 86400
12    and K.finActivation >= L.finActivation - 86400
13    and K.sensorID = 14 "
```

Query 19:

```
1  public static String sumInitialFragments =
2  "insert into SumInitialFragments " +
3   select sum(K.fragmentation) as sumFragments,
4          L.finActivation as instant
5   from WaterStreamFragmentation.win:keepall() as K,
6        TemporaryWaterStream.std:lastevent() as L
7   where L.finActivation = K.timestamp "
```

Query 20:

```
1  public static String findNullTapEvent =
2  "insert into NullTapStream
3   select (b.timestamp - a.timestamp) as duration,
4           a.sensorID as sensorID,
5           a.timestamp+1 as initEvent,
6           b.timestamp as finEvent
7   from pattern [every ((a = WaterMovingEvents
8                    (status = 0 and sensorID=17))
9                 -> (b = WaterMovingEvents(status = 1
10                   and sensorID = 17)))]"
```

Query 21:

```
1  public static String findNullCabinetEvent =
2  "insert into NullCabinetStream
3   select (b.timestamp - a.timestamp) as duration,
4           a.sensorID as sensorID,
5           a.timestamp+1 as initEvent,
6           b.timestamp as finEvent
7   from pattern [every ((a = WaterMovingEvents
8                    (status = 0 and sensorID=14))
9                 -> (b = WaterMovingEvents(status = 1
10                   and sensorID = 14)))]"
```

Query 22:

```
1  public static String findLastTapEvent =
2  "insert into NullTapFragment
3   select (T.finActivation - max(N.finEvent))*0.08
4          as fragmentation,
5           N.sensorID as sensorID,
6           T.finActivation as timestamp
7   from NullTapStream.win:keepall() as N,
8        TemporaryWaterStream.std:lastevent() as T
9   where N.finEvent < T.finActivation
10  and not exists (select *
11      from TemporaryWaterStream.win:keepall() as K
12      where K.finActivation <= T.finActivation
13        and K.finActivation > N.finEvent
14        and K.sensorID = N.sensorID)
15  and not exists (select *
16      from TapStream.win:keepall() as C
17      where C.finActivation >= T.finActivation)"
```

Query 23:

```
1  public static String findLastCabinetEvent =
2  "insert into NullCabinetFragment
3   select (T.finActivation - max(N.finEvent))*0.133
4          as fragmentation,
5           N.sensorID as sensorID,
6           T.finActivation as timestamp
7   from NullCabinetStream.win:keepall() as N,
8        TemporaryWaterStream.std:lastevent() as T
9   where N.finEvent < T.finActivation
10  and not exists (select *
11      from TemporaryWaterStream.win:keepall() as K
12      where K.finActivation <= T.finActivation
13        and K.finActivation > N.finEvent
14        and K.sensorID = N.sensorID)
15  and not exists (select *
16      from CabinetStream.win:keepall() as C
17      where C.finActivation >= T.finActivation)"
```

Query 24:

```
1 public static String
2 sumFinalFragmentsCaseOnlyOneTapFragment =
3 "insert into SumFinalFragments
4  select N.fragmentation as sumFragments,
5         L.finActivation as instant
6  from TemporaryWaterStream.std:lastevent() as L,
7       NullTapFragment.std:lastevent() as N
8  where N.timestamp = L.finActivation "
```

Query 25:

```
1 public static String
2 sumFinalFragmentsCaseOnlyOneCabinetFragment =
3 "insert into SumFinalFragments
4  select N.fragmentation as sumFragments,
5         L.finActivation as instant
6  from TemporaryWaterStream.std:lastevent() as L,
7       NullCabinetFragment.std:lastevent() as N
8  where N.timestamp = L.finActivation "
```

Query 26:

```
1 public static String
2 sumFinalFragmentsCaseTwoFragments=
3 "insert into SumFinalFragments
4  select T.fragmentation + C.fragmentation
5         as sumFragments,
6         L.finActivation as instant
7  from TemporaryWaterStream.std:lastevent() as L,
8       NullTapFragment.win:keepall() as T,
9       NullCabinetFragment.win:keepall() as C
10 where T.timestamp = L.finActivation
11   and C.timestamp = L.finActivation "
```

Query 27:

```
1  public static String overallSumWithInternalsOnly =
2  "insert into WaterStream
3   select sum(S.consumption)
4          as totalWaterConsumption,
5          L.finActivation as timestamp
6   from TemporaryWaterStream.std:lastevent() as L,
7        TemporaryWaterStream.win:keepall() as S
8   where S.initActivation >= L.finActivation -86400 "
```

Query 28:

```
1  public static String
2  overallSumWithInitialFragmentsOnly =
3  "insert into WaterStream
4   select F.sumFragments + sum(S.consumption)
5          as totalWaterConsumption,
6          L.finActivation as timestamp
7   from TemporaryWaterStream.std:lastevent() as L,
8        SumInitialFragments.std:lastevent() as F,
9        TemporaryWaterStream.win:keepall() as S
10  where F.instant = L.finActivation
11    and S.initActivation >= L.finActivation -86400
12    and not exists (select *
13         from SumFinalFragments.win:keepall() as K
14         where K.instant = L.finActivation)"
```

Query 29:

```
1  public static String
2  overallSumWithFinalFragmentsOnly =
3  "insert into WaterStream
4   select F.sumFragments + sum(S.consumption)
5          as totalWaterConsumption,
6          L.finActivation as timestamp
7   from TemporaryWaterStream.std:lastevent() as L,
8        SumFinalFragments.std:lastevent() as F,
9        TemporaryWaterStream.win:keepall() as S
10  where F.instant = L.finActivation
11    and S.initActivation >= L.finActivation -86400
12    and not exists (select *
13         from SumInitialFragments.win:keepall() as K
14         where K.instant = L.finActivation)"
```

Query 30:

```
1   public static String overallSumWithAllFragments =
2   "insert into WaterStream
3    select I.sumFragments + F.sumFragments +
4           sum(S.consumption) as totalWaterConsumption,
5           L.finActivation as timestamp,
6    from TemporaryWaterStream.std:lastevent() as L,
7         SumFinalFragments.std:lastevent() as F,
8         SumInitialFragments.std:lastevent() as I,
9         TemporaryWaterStream.win:keepall() as S
10   where F.instant = L.finActivation
11     and I.instant = L.finActivation
12     and S.initActivation >= L.finActivation -86400"
```

Query 31:

```
1  public static String sendWaterResultToOutputFile =
2  "insert into OutputWaterStream
3   select L.totalWaterConsumption
4          as totalWaterConsumption,
5          Math.floor(L.timestamp/86400) as dayAnalyzed
6   from WaterStream.std:lastevent() as L
7   where ((L.timestamp/86400) >=
8        (1 + (select dayAnalyzed
9        from OutputWaterStream.std:lastevent() as T ))
10     or not exists (select *
11         from OutputWaterStream.win:keepall()))"
```

Query 32:

```
1  public static String
2  overallTapConsumptionCaseComplete =
3  "insert into TotalTapStream
4   select (select sum(K.consumption)
5      from TapStream.win:keepall() as K
6      where K.initActivation >= L.finActivation-86400)
7      + 0.08*((S.finActivation - (L.finActivation -
8      86400))+1)  as totalTapConsumption,
9      L.finActivation as timestamp
10  from TapStream.win:keepall() as S,
11       TapStream.std:lastevent() as L
12  where S.initActivation < L.finActivation-86400
13    and S.finActivation >= L.finActivation-86400"
```

Query 33:

```
1  public static String
2  overallTapConsumptionCaseOnlyInternals =
3  "insert into TotalTapStream
4   select (select sum(K.consumption)
5      from TapStream.win:keepall() as K
6      where K.initActivation >= L.finActivation-86400)
7      as totalTapConsumption,
8      L.finActivation as timestamp
9   from TapStream.std:lastevent() as L
10  where not exists (select *
11     from TapStream.win:keepall() as S
12     where S.initActivation < L.finActivation-86400
13     and S.finActivation >= L.finActivation-86400)"
```

Query 34:

```
1  public static String
2  overallCabinetConsumptionCaseComplete =
3  "insert into TotalCabinetStream
4   select (select sum(K.consumption)
5      from CabnetStream.win:keepall() as K
6      where K.initActivation >= L.finActivation-86400)
7      + 0.133*((S.finActivation - (L.finActivation -
8      86400))+1)  as totalCabinetConsumption,
9      L.finActivation as timestamp
10  from CabinetStream.win:keepall() as S,
11       CabinetStream.std:lastevent() as L
12  where S.initActivation < L.finActivation-86400
13    and S.finActivation >= L.finActivation-86400"
```

Query 35:

```
1  public static String
2  overallCabinetConsumptionCaseOnlyInternals =
3  "insert into TotalCabinetStream
4   select (select sum(K.consumption)
5      from CabinetStream.win:keepall() as K
6      where K.initActivation >= L.finActivation-86400)
7      as totalCabinetConsumption,
8      L.finActivation as timestamp
9   from CabinetStream.std:lastevent() as L
10  where not exists (select *
11     from CabinetStream.win:keepall() as S
12     where S.initActivation < L.finActivation-86400
13     and S.finActivation >= L.finActivation-86400)"
```

Query 36:

```
1   public static String overallWcConsumption =
2   "insert into TotalWcStream
3    select sum(K.consumption) as totalWcConsumption
4    from WcStream.win:keepall() as K,
5         WcStream.std:lastevent() as L
6    where K.finActivation >= L.finActivation -86400 "
```

Appendice B

In questo capitolo sono raccolte tutte le interrogazioni che sono state sviluppate per gli esempi di indicatori sviluppati nel Capitolo 5. Il capitolo è organizzato dividendosi in Sezioni differenti a seconda dell'indicatore cui appartengono le interrogazioni: Sezione B1 per le interrogazioni di ISVUS che rileva l'entropia del singolo sensore del letto (Capitolo 5, Sezione 5.1), Sezione B2 per le interrogazioni di IRAC volte a rilevare le occorrenze dell'attività "guardare la televisione"(Capitolo 5, Sezione 5.2) e Sezione B3 per le interrogazioni sviluppate per estrarre ISVIS, l'entropia di tutta la casa, considerando tutti i sensori allo stesso tempo (Capitolo 5, Sezione 5.3).

B1 Interrogazioni ISVUS

In questa sezione sono presentate in ordine le interrogazioni atte a processare i dati in analisi dal flusso in ingresso al sistema allo scopo di calcolare l'entropia per ogni ora di analisi e per ogni giorno. Le interrogazioni si riferiscono all'indicatore sviluppato nel Capitolo 5, Sezione 5.1.

Query 1:

```
1 public static String bedMovingEventsWindowDefinition=
2 "create window BedMovingEvents.win:length(2)
3  as select * from SensorStream"
```

Query 2:

```
1 public static String populatingBedMovingEventsWindow=
2 "insert into BedMovingEvents
3  select *
4  from SensorStream
5  where sensorID=20"
```

Query 3:

```
public static String extractionEventsEntropyCaseOne =
"insert into BedEntropy
 select count(*) as eventID,
        (b.timestamp - a.timestamp) as timeOn,
         a.timestamp+1 as initEvent,
         b.timestamp as finEvent,
         Math.floor((a.timestamp)/3600) as initHour,
         Math.floor((b.timestamp-1)/3600) as finHour,
         1 as withinSingleDay
 from pattern [every ((a = BedMovingEvents
         (status = 1)) -> (b = BedMovingEvents
         (status = 0)))]
 where Math.floor((a.timestamp)/86400) =
       Math.floor((b.timestamp-1)/86400)"
```

Query 4:

```
public static String extractionEventsEntropyCaseOne =
"insert into BedEntropy
 select count(*) as eventID,
        (b.timestamp - a.timestamp) as timeOn,
         a.timestamp+1 as initEvent,
         b.timestamp as finEvent,
         Math.floor((a.timestamp)/3600) as initHour,
         Math.floor((b.timestamp-1)/3600) as finHour,
         0 as withinSingleDay
 from pattern [every ((a = BedMovingEvents
         (status = 1)) -> (b = BedMovingEvents
         (status = 0)))]
 where Math.floor((a.timestamp)/86400) !=
       Math.floor((b.timestamp-1)/86400)"
```

Query 5:

```
 public static String BedEntropyStandardComputation =
 "insert into SegmentationBedEntropy
  select E.eventID as eventID,
         E.initHour as interval, E.timeOn as timeOn,
         E.initEvent as initEvent,
         E.finEvent as finEvent,
         1 as notExceedingSingleInterval
  from BedEntropy.std:lastevent() as E
  where E.initHour = E.finHour "
```

Query 6:

```
1  public static String
2  bedEntropyFirstPartStandardComputation =
3  "insert into SegmentationBedEntropy
4   select E.eventID as eventID,
5          E.initHour as interval,
6      (3600*(E.initHour+1) - E.initEvent) as timeOn,
7          E.initEvent as initEvent,
8       3600*(E.initHour + 1) as finEvent
9   from BedEntropy.std:lastevent() as E
10  where E.initHour != E.finHour "
```

Query 7:

```
1  public static String multipleIntermediates =
2  "insert into MultipleInternals
3   select E.eventID as eventID,
4          E.initHour as interval,
5         (3600*(E.initHour + 1) +1) as initEvent,
6          3600*(E.finHour) as finEvent
7   from BedEntropy.std:lastevent() as E
8   where E.initHour != E.finHour
9     and E.finHour-E.initHour > 1 "
```

Query 8:

```
1 public static String iterationOverMultipleInternals =
2 "insert into SegmentationBedEntropy
3  select M.eventID as eventID, M.interval as interval,
4         M.initEvent as initEvent,
5         3599+M.initEvent as finEvent, 3600 as timeOn
6  from MultipleInternals.std:lastevent() as M
7  where initEvent = (select 1 + A.finEvent
8        from AnalyzedSegment.std:lastevent() as A
9        where M.eventID = A.eventID and A.inserted=1)"
```

Query 9:

```
1  public static String saveSecondPart =
2  "insert into MultipleInternals
3   select A.eventID as eventID,
4          (A.interval+1) as interval,
5          (A.initEvent + 3600) as initEvent,
6          (select M.finEvent
7           from MultipleInternals.std:lastevent() as M
8           where M.eventID = A.eventID) as finEvent
9   from AnalyzedSegment.std:lastevent() as A
10  where A.inserted=1 and A.timeOn = 3600
11    and A.finEvent != (select M.finEvent
12         from MultipleInternals.std:lastevent() as M
13         where M.eventID = A.eventID)"
```

Query 10:

```
1  public static String
2  bedEntropySecondHalfStandardComputation =
3  "insert into SegmentationBedEntropy
4   select A.eventID as eventID,
5          A.interval+1 as interval,
6          (select (E.finEvent - A.finEvent+1)
7           from BedEntropy.std:lastevent() as E
8           where E.eventID = A.eventID) as timeOn,
9           A.finEvent+1 as initEvent,
10         (select E.finEvent
11          from BedEntropy.std:lastevent() as E
12          where E.eventID = A.eventID) as finEvent
13  from AnalyzedSegment.std:lastevent() as A " +
14  where A.inserted=1
15    and A.interval + 1 = (select E.finHour
16         from BedEntropy.std:lastevent() as E
17         where E.eventID = A.eventID)
18  order by initEvent"
```

Query 11:

```
1  public static String approveHourlyEntropyCaseOne =
2  "insert into AnalyzedSegment
3   select L.eventID as eventID,
4          L.interval as interval,L.timeOn as timeOn,
5          L.initEvent as initEvent,
6          L.finEvent as finEvent,
7          0 as inserted, 1 as approved
8   from SegmentationBedEntropy.std:lastevent() as L
9   where L.eventID = 1"
```

Query 12:

```
1  public static String approveHourlyEntropyCaseTwo =
2  "insert into AnalyzedSegment
3   select L.eventID as eventID,
4          L.interval as interval,L.timeOn as timeOn,
5          L.initEvent as initEvent,
6          L.finEvent as finEvent,
7          0 as inserted, 1 as approved
8   from SegmentationBedEntropy.std:lastevent() as L
9   where (L.eventID = 0 or L.eventID > 1)
10        and ((L.interval = 1 + (select interval
11              from HourlyEntropy.std:lastevent()))
12        or (L.interval = (select interval
13          from HourlyEntropy.std:lastevent()))))"
```

Query 13:

```
1 public static String approveHourlyEntropyCaseThree =
2  "insert into AnalyzedSegment
3   select L.eventID as eventID,
4          L.interval as interval, L.timeOn as timeOn,
5          L.initEvent as initEvent,
6          L.finEvent as finEvent,
7          0 as inserted, 0 as approved
8   from SegmentationBedEntropy.std:lastevent() as L
9   where (L.eventID = 0 or L.eventID > 1)
10      and L.interval != 1 + (select interval
11             from HourlyEntropy.std:lastevent())
12      and L.interval != (select interval
13                from HourlyEntropy.std:lastevent())"
```

Query 14:

```
1  public static String hourEntropyComputation =
2  "insert into HourlyEntropy
3   select (select (-1)*(
4  ((sum(S.timeOn)/3600)*Math.log(sum(S.timeOn)/3600))
5          /Math.log(2) + (
6  (sum(S.timeOn)/3600)*Math.log(1sum(S.timeOn)/3600))
7          /Math.log(2))
8      from SegmentationBedEntropy.win:keepall() as S
9      where S.interval=A.interval) as hourlyEntropy,
10    (select sum(S.timeOn)
11     from SegmentationBedEntropy.win:keepall() as S
12     where S.interval = A.interval) as partialSum,
13    (select S.timeOn
14    from SegmentationBedEntropy.std:lastevent() as S
15     where S.interval = A.interval
16       and S.eventID = A.eventID) as timeOn,
17     A.interval*3600 + 1 as initInterval,
18    (A.interval+1)*3600 as finInterval,
19     A.eventID as eventID, A.interval as interval
20   from AnalyzedSegment.std:lastevent() as A
21   where A.approved = 1 and A.inserted = 0 "
```

Query 15:

```
1  public static String setInsertedIntoAnalyzed =
2  "insert into AnalyzedSegment " +
3   select H.eventID as eventID,
4          H.interval as interval,
5          1 as approved, 1 as inserted
6         (select timeOn
7          from AnalyzedSegment.std:lastevent())
8          as timeOn,
9         (select initEvent
10         from AnalyzedSegment.std:lastevent())
11         as initEvent,
12        (select finEvent
13         from AnalyzedSegment.std:lastevent())
14         as finEvent
15   from HourlyEntropy.std:lastevent() as H
16   where H.eventID = (select eventID
17         from AnalyzedSegment.std:lastevent())
18   and H.interval = (select interval
19       from AnalyzedSegment.std:lastevent())"
```

Query 16:

```
1  public static String nullIntervalsAdd =
2  "insert into HourlyEntropy
3   select 0 as hourlyEntropy, 0 as partialSum,
4      0 as timeOn, A.eventID as eventID,
5      1 + (select finInterval
6           from HourlyEntropy.std:lastevent())
7      as initInterval,
8   3600 + (select finInterval
9           from HourlyEntropy.std:lastevent())
10     as finInterval,
11     1 + (select interval
12          from HourlyEntropy.std:lastevent())
13     as interval
14  from AnalyzedSegment.std:lastevent() as A
15  where A.approved = 0 "
```

Query 17:

```
1  public static String turnIntoApproved =
2  "insert into AnalyzedSegment
3   select H.eventID as eventID,
4          H.interval+1 as interval,
5          1 as approved, 0 as inserted,
6       (select timeOn
7        from AnalyzedSegment.std:lastevent())
8        as timeOn,
9       (select initEvent
10       from AnalyzedSegment.std:lastevent())
11       as initEvent,
12      (select finEvent
13       from AnalyzedSegment.std:lastevent())
14       as finEvent,
15  from HourlyEntropy.std:lastevent() as H
16  where hourlyEntropy = 0
17        and H.interval + 1 = (select interval
18        from AnalyzedSegment.std:lastevent()
19        where approved = 0)"
```

Query 18:

```
public static String stillCannotApprove =
"insert into AnalyzedSegment
 select H.eventID as eventID, 0 as inserted,
        0 as approved,
     (select interval
      from AnalyzedSegment.std:lastevent())
      as interval,
      (select timeOn
       from AnalyzedSegment.std:lastevent())
       as timeOn,
      (select initEvent
       from AnalyzedSegment.std:lastevent())
       as initEvent,
      (select finEvent
       from AnalyzedSegment.std:lastevent())
       as finEvent,
 from HourlyEntropy.std:lastevent() as H
 where hourlyEntropy = 0
       and H.interval + 1 != (select interval
       from AnalyzedSegment.std:lastevent()
       where approved = 0)"
```

Query 19:

```
public static String
windowOfPreviousHourlyEntropyEventPopulate =
"insert into HourlyEntropyPrev
 select a.hourlyEntropy as hourlyEntropy,
        a.initInterval as initInterval,
        a.finInterval as finInterval
 from pattern [every (a = HourlyEntropy)]
 group by a.countFactor
 having count(*) = 1"
```

Query 20:

```
public static String
windowOfLatestHourlyEntropyEventPopulate =
"insert into HourlyEntropyPrev
 select a.hourlyEntropy as hourlyEntropy,
        a.initInterval as initInterval,
        a.finInterval as finInterval
 from pattern [every (a = HourlyEntropy)]
 group by a.countFactor
 having count(*) > 1"
```

Query 21:

```
public static String
comparisonBetweenLastTwoHourlyEntropyEvents =
"insert into OutputHourlyEntropy
 select P.hourlyEntropy as hourlyEntropy,
        P.initInterval as initInterval,
        P.finInterval as finInterval
 from HourlyEntropyPrev.std:lastevent() as P,
      HourlyEntropyNext.std:lastevent() as N
 where P.initInterval != N.initInterval
  and not exists (select *
      from OutputHourlyEntropy.win:keepall() as O
      where O.hourlyEntropy = P.hourlyEntropy
      and O.initInterval = P.initInterval)"
```

Query 22:

```
public static String
updateHourlyEntropyPrevWithNextCaseTwo =
"insert into HourlyEntropyPrev
 select (select N.hourlyEntropy
       from HourlyEntropyNext.std:lastevent() as N)
        as hourlyEntropy,
        (select N.initInterval
       from HourlyEntropyNext.std:lastevent() as N)
        as initInterval,
        (select N.finInterval
       from HourlyEntropyNext.std:lastevent() as N)
        as finInterval
 from OutputHourlyEntropy.std:lastevent() as O
 where O.hourlyEntropy = (select P.hourlyEntropy
      from HourlyEntropyPrev.std:lastevent() as P)"
```

Query 23:

```
public static String
updateHourlyEntropyPrevWithNextCaseOne =
"insert into HourlyEntropyPrev
 select N.hourlyEntropy as hourlyEntropy,
        N.initInterval as initInterval,
        N.finInterval as finInterval
 from HourlyEntropyPrev.std:lastevent() as P,
      HourlyEntropyNext.std:lastevent() as N
 where P.hourlyEntropy != N.hourlyEntropy
   and P.initInterval = N.initInterval"
```

Query 24:

```
1  public static String windowForFirstBedActivation =
2  "create window FirstActivationWindow.win:length(1)
3   as select * from FirstDailyActivation"
```

Query 25:

```
1  public static String
2   populateWindowForFirstBedActivation =
3  "insert into FirstActivationWindow
4   select S.timestamp as instantFirstActivation
5   from SensorStream as S
6   where S.sensorID = 20 and S.status = 1
7   group by S.sensorID
8   having count(*)=1"
```

Query 26:

```
1  public static String initialNullIntervals =
2  "insert into HourlyEntropy
3   select 0 as hourlyEntropy,
4          1 as initInterval, 3600 as finInterval
5   from FirstActivationWindow
6   where instantFirstActivation > 3600 "
```

Query 27:

```
1 public static String
2 windowOfPreviousDailyEntropyEventPopulate =
3 "insert into DailyEntropyPrev
4  select a.dailyEntropy as dailyEntropy,
5         a.initInterval as initInterval,
6         a.finInterval as finInterval
7  from pattern [every (a = DailyEntropy)]
8  group by a.countFactor
9  having count(*) = 1"
```

Query 28:

```
1 public static String
2 windowOfLatestDailyEntropyEventPopulate =
3 "insert into DailyEntropyPrev
4  select a.dailyEntropy as dailyEntropy,
5         a.initInterval as initInterval,
6         a.finInterval as finInterval
7  from pattern [every (a = DailyEntropy)]
8  group by a.countFactor
9  having count(*) > 1"
```

Query 29:

```
public static String
comparisonBetweenLastTwoDailyEntropyEvents =
"insert into OutputDailyEntropy
 select P.dailyEntropy as dailyEntropy,
        (Math.floor(P.initInterval/86400) + 1)
         as dayAnalyzed
 from DailyEntropyPrev.std:lastevent() as P,
      DailyEntropyNext.std:lastevent() as N
 where P.initInterval != N.initInterval
   and not exists (select *
       from OutputDailyEntropy.win:keepall() as O
       where O.dailyEntropy = P.dailyEntropy
         and O.dayAnalyzed =
             Math.floor(P.initInterval/86400) + 1)"
```

Query 30:

```
public static String
updateDailyEntropyPrevWithNextCaseOne =
"insert into DailyEntropyPrev
 select N.dailyEntropy as dailyEntropy,
        N.initInterval as initInterval,
        N.finInterval as finInterval
 from DailyEntropyPrev.std:lastevent() as P,
      DailyEntropyNext.std:lastevent() as N
 where P.dailyEntropy != N.dailyEntropy
   and P.initInterval = N.initInterval"
```

Query 31:

```
public static String
updateDailyEntropyPrevWithNextCaseTwo =
"insert into DailyEntropyPrev
 select (select N.dailyEntropy
         from DailyEntropyNext.std:lastevent() as N)
         as dailyEntropy,
        (select N.initInterval
         from DailyEntropyNext.std:lastevent() as N)
         as initInterval,
        (select N.finInterval
         from DailyEntropyNext.std:lastevent() as N)
         as finInterval
 from OutputDailyEntropy.std:lastevent() as O
 where O.dailyEntropy = (select P.dailyEntropy
       from DailyEntropyPrev.std:lastevent() as P) "
```

B2 Interrogazioni IRAC

In questa sezione vengono introdotte le interrogazioni adottate per costruire un indicatore in grado di riconoscere dai soli dati restituiti dai sensori se la persona monitorata stia guardando la televisione. Le interrogazioni si riferiscono all'indicatore sviluppato nel Capitolo 5, Sezione 5.2.

Query 32:

```
1 public static String movingControlEventsDefinition =
2 "create window MovingControlEvents.win:length(2)
3  as select * from SensorStream"
```

Query 33:

```
1 public static String movingControlEventsPopulate =
2 "insert into MovingControlEvents
3  select *
4  from SensorStream
5  where sensorID = 3"
```

Query 34:

```
1  public static String controlActivationStream =
2  "insert into ControlStream
3   select count(*) as eventID, a.sensorID as sensorID,
4          a.timestamp+1 as initActivation,
5          b.timestamp as finActivation
6   from pattern [every ((a = MovingControlEvents
7       (status = 1)) -> ((b = MovingControlEvents
8       (status = 0))))]
9   where not exists (select *
10        from WaitedSitStream.std:lastevent()
11        where (sensorID = 5 OR sensorID = 4
12              OR sensorID = 6 OR sensorID = 7)
13          and pending = 1)"
```

Query 35:

```
1 public static String movingSitEventsDefinition =
2 "create window MovingSitEvents.win:length(2)
3  as select * from SensorStream"
```

Query 36:

```
1 public static String movingSitEventsPopulate =
2 "insert into MovingSitEvents
3  select *
4  from SensorStream
5  where sensorID = 4 OR sensorID = 5 OR sensorID = 6
6     OR sensorID = 7"
```

Query 37:

```
1  public static String
2  couchEventSingleDurationCaseOne=
3  "insert into SitStream
4   select count(*) as eventID, a.sensorID as sensorID,
5         (b.timestamp - a.timestamp) as duration,
6          a.timestamp+1 as initActivation,
7          b.timestamp as finActivation
8   from pattern [every ((a = MovingSitEvents(status=1
9                and sensorID = 5)) ->
10               ((b = MovingSitEvents(status = 0
11               and sensorID = 5))))]
12  where (a.timestamp+1 = (select T.initActivation
13        from WaitedSitStream.std:lastevent() as T
14        where T.sensorID = 5)
15     OR a.timestamp+1 = (select T.initActivation
16        from TvActivationInstant.std:lastevent() as T
17        where T.sensorID = 5))
18    "and a.timestamp + 1 not in (select initActivation
19        from SitStream.win:keepall())"
```

Query 38:

```
1  public static String
2  couchEventSingleDurationCaseTwo=
3  "insert into SitStream
4   select count(*) as eventID, a.sensorID as sensorID,
5          (b.timestamp - a.timestamp) as duration,
6           a.timestamp+1 as initActivation,
7           b.timestamp as finActivation
8   from pattern [every ((a = MovingSitEvents(status=1
9                 and sensorID = 4)) ->
10                ((b = MovingSitEvents(status = 0
11                and sensorID = 4))))]
12 where (a.timestamp+1 = (select T.initActivation
13         from WaitedSitStream.std:lastevent() as T
14         where T.sensorID = 4)
15      OR a.timestamp+1 = (select T.initActivation
16         from TvActivationInstant.std:lastevent() as T
17         where T.sensorID = 4))
18    "and a.timestamp + 1 not in (select initActivation
19         from SitStream.win:keepall())"
```

Query 39:

```
1  public static String
2  couchEventSingleDurationCaseThree =
3  "insert into SitStream
4   select count(*) as eventID, a.sensorID as sensorID,
5          (b.timestamp - a.timestamp) as duration,
6           a.timestamp+1 as initActivation,
7           b.timestamp as finActivation
8   from pattern [every ((a = MovingSitEvents(status=1
9                 and sensorID = 6)) ->
10                ((b = MovingSitEvents(status = 0
11                and sensorID = 6))))]
12  where (a.timestamp+1 = (select T.initActivation
13         from WaitedSitStream.std:lastevent() as T
14         where T.sensorID = 6)
15      OR a.timestamp+1 = (select T.initActivation
16         from TvActivationInstant.std:lastevent() as T
17         where T.sensorID = 6))
18    "and a.timestamp + 1 not in (select initActivation
19         from SitStream.win:keepall())"
```

Query 40:

```
1  public static String
2  couchEventSingleDurationCaseFour =
3  "insert into SitStream
4   select count(*) as eventID, a.sensorID as sensorID,
5          (b.timestamp - a.timestamp) as duration,
6           a.timestamp+1 as initActivation,
7           b.timestamp as finActivation
8   from pattern [every ((a = MovingSitEvents(status=1
9                 and sensorID = 7)) ->
10                ((b = MovingSitEvents(status = 0
11                and sensorID = 7))))]
12  where (a.timestamp+1 = (select T.initActivation
13         from WaitedSitStream.std:lastevent() as T
14         where T.sensorID = 7)
15     OR a.timestamp+1 = (select T.initActivation
16         from TvActivationInstant.std:lastevent() as T
17         where T.sensorID = 7))
18    "and a.timestamp + 1 not in (select initActivation
19         from SitStream.win:keepall())"
```

Query 41:

```
1  public static String findFirstControlActivation =
2  "insert into TvActivationInstant
3   select a.eventID as eventID, a.sensorID as sensorID,
4          a.initActivation as initActivation,
5          a.finActivation as finActivation,
6          -1 as followingInstant
7   from pattern [every (a = ControlStream)]
8   where a.eventID = 1";
```

Query 42:

```
1  public static String findOtherControlActivation =
2  "insert into TemporaryTvActivationInstant
3   select a.eventID as eventID, a.sensorID as sensorID,
4          a.initActivation as initActivation,
5          a.finActivation as finActivation,
6          -1 as followingInstant
7   from pattern [every (a = ControlStream)]
8   where a.eventID > 1";
```

Query 43:

```
public static String
tvEventAllOtherActivationInstants =
"insert into TemporaryTvActivationInstant
 select count(*) as eventID, a.sensorID as sensorID,
        a.timestamp+1 as initActivation
 from pattern [every (a = MovingSitEvents(status = 1
              and (sensorID=5 OR sensorID = 4 OR
              sensorID=6 OR sensorID=7)))]
 where not exists (select *
       from WaitedSitStream.std:lastevent()
       where pending = 1)
   and exists (select *
       from ControlStream.win:keepall()) ";
```

Query 44:

```
public static String
afterATvSignalItArrivesATvSignalNotOkCaseOne =
"insert into TvActivationInstant
 select L.eventID as eventID, L.sensorID as sensorID,
        L.initActivation as initActivation,
        -1 as followingInstant, 0 as aggregated,
        L.finActivation as finActivation
from TemporaryTvActivationInstant.std:lastevent() as L
    TvActivationInstant.std:lastevent() as T
 where L.sensorID = 3 and T.sensorID = 3
   and L.initActivation - T.finActivation > 180
   and T.aggregated = 0 "
```

Query 45:

```
public static String
afterATvSignalItArrivesACouchSignalNotOkCaseOne =
"insert into TvActivationInstant " +
 select L.eventID as eventID, L.sensorID as sensorID,
        L.initActivation as initActivation,
        -1 as followingInstant, 0 as aggregated
from TemporaryTvActivationInstant.std:lastevent() as L
    TvActivationInstant.std:lastevent() as T
 where (L.sensorID = 5 OR L.sensorID=4 OR
        L.sensorID = 6 OR L.sensorID = 7)
   and T.sensorID = 3 and T.aggregated = 0
   and L.initActivation - T.finActivation > 180 ";
```

Query 46:

```
1 public static String
2 arrivalOfATvSignalAfterCouchCaseOne =
3 "insert into TvActivationInstant
4  select L.eventID as eventID, L.sensorID as sensorID,
5         L.initActivation as initActivation,
6         0 as aggregated, -1 as followingInstant,
7         L.finActivation as finActivation
8from TemporaryTvActivationInstant.std:lastevent() as L
9     TvActivationInstant.std:lastevent() as T
10  where L.sensorID = 3 and (T.sensorID = 5
11        OR T.sensorID = 4 OR T.sensorID = 6
12        OR T.sensorID = 7) and T.aggregated = 0
13    and ((exists (select *
14        from SitStream.win:keepall() as K
15        where K.initActivation = T.initActivation)
16    and L.initActivation - (select K.finActivation
17        from SitStream.win:keepall() as K
18        where K.initActivation=T.initActivation)>180)
19     OR T.addedForDayClosing = 1)"
```

Query 47:

```
1 public static String afterCouchFollowsCouchCaseOne =
2 "insert into TvActivationInstant
3  select L.eventID as eventID, L.sensorID as sensorID,
4         L.initActivation as initActivation,
5         0 as aggregated, -1 as followingInstant
6from TemporaryTvActivationInstant.std:lastevent() as L
7     TvActivationInstant.std:lastevent() as T
8  where (L.sensorID = 5 OR L.sensorID = 4
9         OR L.sensorID = 6 OR L.sensorID = 7)
10    and (T.sensorID = 5 OR T.sensorID = 4
11         OR T.sensorID = 6 OR T.sensorID = 7)
12    and T.aggregated = 0
13    and L.initActivation - (select K.finActivation
14        from SitStream.win:keepall() as K
15        where K.initActivation=T.initActivation)>180"
```

Query 48:

```
public static String
afterATvSignalItArrivesATvSignalAggregatedNotOkCaseTwo=
"insert into FinalTvStream
 select (select initActivation
         from GroupedTvStream.std:lastevent())
         as initActivation,
        (select finActivation
         from GroupedTvStream.std:lastevent())
         as finActivation,
        (select finActivation
         from GroupedTvStream.std:lastevent()) -
             (select initActivation
              from GroupedTvStream.std:lastevent())
         as timeWatchingTv
from TemporaryTvActivationInstant.std:lastevent() as T
    TvActivationInstant.std:lastevent() as L
 where T.sensorID = 3 and L.sensorID = 3
   and L.aggregated = 1 and T.initActivation -
       (select finActivation
        from GroupedTvStream.std:lastevent()) > 180
   and (select approved
        from GroupedTvStream.std:lastevent()) = 1
   and (select finActivation
        from GroupedTvStream.std:lastevent())
        not in (select finActivation
                from FinalTvStream.win:keepall()
                where dayEnded!= 1)"
```

Query 49:

```
public static String
updateTvActivationAfterTvSignalAggregatedNotOkCaseTwo=
"insert into TvActivationInstant
 select T.initActivation as initActivation,
        T.eventID as eventID, T.sensorID as sensorID,
        -1 as followingInstant, 0 as aggregated,
        T.finActivation as finActivation
from TemporaryTvActivationInstant.std:lastevent() as T
    FinalTvStream.std:lastevent() as F
 where T.sensorID = 3 and (select sensorID
       from TvActivationInstant.std:lastevent()) = 3
   and (select aggregated
        from TvActivationInstant.std:lastevent()) = 1
   and F.dayEnded = 0 and F.finActivation =
       (select finActivation
        from GroupedTvStream.std:lastevent())
   and T.initActivation NOT IN (select couchExpected
       from WaitForDayClosing.win:keepall())"
```

Query 50:

```
1 public static String
2 arrivalCouchEventAfterTvSignalApprovedCaseTwo=
3 "insert into FinalTvStream
4  select (select initActivation
5          from GroupedTvStream.std:lastevent())
6          as initActivation,
7         (select finActivation
8          from GroupedTvStream.std:lastevent())
9          as finActivation,
10        (select finActivation
11         from GroupedTvStream.std:lastevent()) -
12             (select initActivation
13              from GroupedTvStream.std:lastevent())
14         as timeWatchingTv
15from TemporaryTvActivationInstant.std:lastevent() as T
16     TvActivationInstant.std:lastevent() as L
17  where (T.sensorID = 4 OR T.sensorID = 5 OR
18         T.sensorID = 6 or T.sensorID = 7)
19    and L.sensorID = 3
20    and L.aggregated = 1 and T.initActivation -
21        (select finActivation
22         from GroupedTvStream.std:lastevent()) > 180
23    and (select approved
24         from GroupedTvStream.std:lastevent()) = 1
25    and (select finActivation
26         from GroupedTvStream.std:lastevent())
27         not in (select finActivation
28                 from FinalTvStream.win:keepall()
29                 where dayEnded!= 1)"
```

Query 51:

```
1 public static String
2 updateTvActivationAfterCouchTvApprovedCase =
3 "insert into TvActivationInstant
4  select T.initActivation as initActivation,
5         T.eventID as eventID, T.sensorID as sensorID,
6         -1 as followingInstant, 0 as aggregated,
7         T.finActivation as finActivation
8from TemporaryTvActivationInstant.std:lastevent() as T
9     FinalTvStream.std:lastevent() as F
10  where (T.sensorID = 4 OR T.sensorID = 5 OR
11         T.sensorID = 6 OR T.sensorID = 7)
12    and (select sensorID
13         from TvActivationInstant.std:lastevent()) = 3
14    and (select aggregated
15          from TvActivationInstant.std:lastevent()) = 1
16    and F.dayEnded = 0 and T.initActivation -
17        (select G.finActivation
18         from TvActivationInstant.std:lastevent() as G)> 180
19    and T.initActivation NOT IN (select couchExpected
20         from WaitForDayClosing.win:keepall())"
```

Query 52:

```
1 public static String
2 arrivalOfATvSignalAfterCouchAggregatedCase =
3 "insert into FinalTvStream
4  select (select initActivation
5          from GroupedTvStream.std:lastevent())
6          as initActivation,
7         (select finActivation
8          from GroupedTvStream.std:lastevent())
9          as finActivation,
10        (select finActivation
11         from GroupedTvStream.std:lastevent())
12      - (select initActivation
13         from GroupedTvStream.std:lastevent())
14         as timeWatchingTv
15from TemporaryTvActivationInstant.std:lastevent() as L
16    TvActivationInstant.std:lastevent() as T
17  where L.sensorID = 3 and (T.sensorID = 5 OR
18       T.sensorID = 4 OR T.sensorID = 6 OR
19       T.sensorID = 7) and exists (select *
20          from SitStream.win:keepall() as K
21          where K.initActivation = T.initActivation)
22      and T.aggregated = 1 and (select approved
23          from GroupedTvStream.std:lastevent()) = 1
24      and L.initActivation - (select K.finActivation
25          from SitStream.win:keepall() as K
26          where K.initActivation
27          = T.initActivation)>180)
28      and (select finActivation
29          from GroupedTvStream.std:lastevent())
30          not in (select finActivation
31          from FinalTvStream.win:keepall())"
```

Query 53:
```
public static String
updateTvActivationInstantAfterTvSignalCouchAggregatedCase=
 "insert into TvActivationInstant
  select L.eventID as eventID, L.sensorID as sensorID,
         L.initActivation as initActivation,
         0 as aggregated, -1 as followingInstant,
         L.finActivation as finActivation
from TemporaryTvActivationInstant.std:lastevent() as L
     FinalTvStream.std:lastevent() as F
  where F.finActivation = (select finActivation
        from GroupedTvStream.std:lastevent())
    and L.sensorID = 3 and ((select sensorID
        from TvActivationInstant.std:lastevent()) = 5
        OR (select sensorID
        from TvActivationInstant.std:lastevent()) = 4
        OR (select sensorID
        from TvActivationInstant.std:lastevent()) = 6
        OR (select sensorID
        from TvActivationInstant.std:lastevent()) = 7)
    and (select aggregated
        from TvActivationInstant.std:lastevent()) = 1"
```

Query 54:

```
public static String
afterACouchSignalItArrivesACouchSignalNotOkCaseTwo =
"insert into FinalTvStream
 select (select initActivation
         from GroupedTvStream.std:lastevent())
         as initActivation,
        (select finActivation
         from GroupedTvStream.std:lastevent())
         as finActivation,
        (select finActivation
         from GroupedTvStream.std:lastevent())
      - (select initActivation
         from GroupedTvStream.std:lastevent())
         as timeWatchingTv
from TemporaryTvActivationInstant.std:lastevent() as L
    TvActivationInstant.std:lastevent() as T
 where (L.sensorID = 5 OR L.sensorID = 4 OR
        L.sensorID = 6 OR L.sensorID = 7)
   and (T.sensorID = 5 OR T.sensorID = 4 OR
        T.sensorID = 6 OR T.sensorID = 7)
   and T.addedForDayClosing != 1 and T.aggregated = 1
   and 1 = (select approved
       from GroupedTvStream.std:lastevent())
   and ( (L.initActivation - (select K.finActivation
       from SitStream.win:keepall() as K
       where K.initActivation=T.initActivation)> 180)
      OR (L.initActivation-(select lastControlInstant
       from GroupedTvStream.std:lastevent())>1800) )
  "and (select finActivation
       from GroupedTvStream.std:lastevent()) not in
       (select finActivation
       from FinalTvStream.win:keepall())"
```

Query 55:

```
1  public static String
2  updateTvActivationInstantAfterCouchCaseTwo =
3  "insert into TvActivationInstant
4   select L.eventID as eventID, L.sensorID as sensorID,
5          L.initActivation as initActivation,
6          0 as aggregated, -1 as followingInstant
7  from TemporaryTvActivationInstant.std:lastevent() as L
8       FinalTvStream.std:lastevent() as F
9   where F.initActivation = (select initActivation
10         from GroupedTvStream.std:lastevent())
11    and (L.sensorID = 5 OR L.sensorID = 4 OR
12         L.sensorID = 6 OR L.sensorID = 7)
13    and (5 = (select sensorID
14         from TvActivationInstant.std:lastevent())
15     OR 4 = (select sensorID
16         from TvActivationInstant.std:lastevent())
17     OR 6 = (select sensorID
18         from TvActivationInstant.std:lastevent())
19     OR 7 = (select sensorID
20         from TvActivationInstant.std:lastevent())
21    and 1 = (select aggregated
22         from TvActivationInstant.std:lastevent())
23    and F.dayEnded = 0 and L.initActivation NOT IN
24         (select couchExpected
25         from WaitForDayClosing.win:keepall())"
```

Query 56:

```
1  public static String
2  afterATvSignalItArrivesATvSignalOkCaseOne =
3  "insert into TvActivationInstant
4   select L.eventID as eventID, L.sensorID as sensorID,
5          T.initActivation as followingInstant,
6          L.initActivation as initActivation,
7          L.finActivation as finActivation
8  from TemporaryTvActivationInstant.std:lastevent() as L
9     TvActivationInstant.std:lastevent() as T
10 where L.sensorID = 3 and T.sensorID = 3
11   and not exists (select *
12        from TvActivationInstant.win:keepall() as K
13        where K.initActivation = L.initActivation)
14   and L.initActivation - T.finActivation <= 180
15   and T.followingInstant = -1 and T.aggregated = 0"
```

Query 57:

```
public static String
afterATvSignalItArrivesATvSignalOkCaseTwo =
"insert into TvActivationInstant
 select L.eventID as eventID, L.sensorID as sensorID,
        T.initActivation as followingInstant,
        L.initActivation as initActivation,
        L.finActivation as finActivation
from TemporaryTvActivationInstant.std:lastevent() as L
   TvActivationInstant.std:lastevent() as T
where L.sensorID = 3 and T.sensorID = 3
  and T.eventID != L.eventID and
  and not exists (select *
      from TvActivationInstant.win:keepall() as K
      where K.initActivation = L.initActivation)
  and L.initActivation - T.finActivation <= 180
  and T.followingInstant != -1 and T.aggregated = 0"
```

Query 58:

```
public static String okFromWaitedCouchEventCaseOne =
"insert into GroupedTvStream
 select (select L.initActivation
       from TvActivationInstant.std:lastevent() as L)
       as initActivation,
       C.finActivation as finActivation,
       1 as approved, (select L.initActivation
       from TvActivationInstant.std:lastevent() as L)
       as lastControlInstant
 from SitStream.std:lastevent() as C,
     WaitedSitStream.std:lastevent() as W
 where C.initActivation = W.initActivation
   and W.pending = 1 and (5 = (select sensorID
   from TemporaryTvActivationInstant.std:lastevent())
   OR 4 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent())
   OR 6 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent())
   OR 7 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent()))
   and 3 = (select sensorID
  from TvActivationInstant.std:lastevent())
   and -1 = (select L.followingInstant
  from TvActivationInstant.std:lastevent() as L)
   and 0 = (select aggregated
  from TvActivationInstant.std:lastevent())
   and ( (exists (select *
  from GroupedTvStream.win:keepall())
   and (select Math.floor((G.initActivation-1)/86400)
       from GroupedTvStream.std:lastevent() as G)
          = Math.floor((W.initActivation-1)/86400) )
      OR (not exists (select *
       from GroupedTvStream.win:keepall())) )"
```

Query 59:

```
public static String
okFromWaitedCouchEventCaseOneBis=
"insert into GroupedTvStream
 select (select L.followingInstant
       from TvActivationInstant.std:lastevent() as L)
       as initActivation,
       C.finActivation as finActivation,
       1 as approved,
      (select L.initActivation
       from TvActivationInstant.std:lastevent() as L)
       as lastControlInstant
 from SitStream.std:lastevent() as C,
      WaitedSitStream.std:lastevent() as W
 where C.initActivation = W.initActivation
      and W.pending = 1 and (5 = (select sensorID
   from TemporaryTvActivationInstant.std:lastevent())
    OR 4 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent())
    OR 6 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent())
    OR 7 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent()))
   and 3 = (select sensorID
  from TvActivationInstant.std:lastevent())
   and -1 != (select L.followingInstant
  from TvActivationInstant.std:lastevent() as L)
  "and 0 = (select aggregated
  from TvActivationInstant.std:lastevent())"
```

Query 60:

```
1  public static String
2  okFromWaitedCouchEventCaseOneAlternative =
3  "insert into GroupedTvStream
4   select C.initActivation as initActivation,
5          C.finActivation as finActivation,
6          1 as approved, (select L.initActivation
7         from TvActivationInstant.std:lastevent() as L)
8          as lastControlInstant
9   from SitStream.std:lastevent() as C,
10       WaitedSitStream.std:lastevent() as W
11  where C.initActivation = W.initActivation
12    and W.pending = 1
13    and (5 = (select sensorID
14    from TemporaryTvActivationInstant.std:lastevent())
15         OR 4 = (select sensorID
16    from TemporaryTvActivationInstant.std:lastevent())
17         OR 6 = (select sensorID
18    from TemporaryTvActivationInstant.std:lastevent())
19         OR 7 = (select sensorID
20   from TemporaryTvActivationInstant.std:lastevent()))
21   and 3 = (select sensorID
22       from TvActivationInstant.std:lastevent())
23   and -1 = (select L.followingInstant
24       from TvActivationInstant.std:lastevent() as L)
25   and 0 = (select aggregated
26       from TvActivationInstant.std:lastevent())
27  "and (select Math.floor((G.initActivation-1)/86400)
28       from GroupedTvStream.std:lastevent() as G)
29        != Math.floor((W.initActivation-1)/86400)"
```

Query 61:

```
1 public static String
2 updateTvActivationAfterCouchEvent =
3 "insert into TvActivationInstant
4  select (select L.initActivation
5from TemporaryTvActivationInstant.std:lastevent() as L)
6        as initActivation, " +
7  select (select L.eventID
8from TemporaryTvActivationInstant.std:lastevent() as L)
9        as eventID
10  select (select L.sensorID
11from TemporaryTvActivationInstant.std:lastevent() as L)
12       as sensorID
13       1 as aggregated, -1 as followingInstant
14  from GroupedTvStream.std:lastevent() as G,
15      SitStream.std:lastevent() as C
16  where G.finActivation = (select C.finActivation
17       from SitStream.std:lastevent() as C)
18    and 0 = (select aggregated
19       from TvActivationInstant.std:lastevent())
20    and (5 = (select sensorID
21    from TemporaryTvActivationInstant.std:lastevent())
22    and (4 = (select sensorID
23    from TemporaryTvActivationInstant.std:lastevent())
24    and (6 = (select sensorID
25    from TemporaryTvActivationInstant.std:lastevent())
26    and (7 = (select sensorID
27   from TemporaryTvActivationInstant.std:lastevent()))
28    and 3 = (select sensorID
29       from TvActivationInstant.std:lastevent())
30   and C.initActivation NOT IN (select couchExpected
31       from WaitForDayClosing.win:keepall()) "
```

Query 62:

```
1 public static  String
2 arrivalOfATvSignalAfterCouchCaseThree =
3 "insert into TvActivationInstant
4  select L.initActivation as initActivation,
5         L.sensorID as sensorID, L.eventID as eventID,
6         -1 as followingInstant, 0 as aggregated,
7         L.finActivation as finActivation
8from TemporaryTvActivationInstant.std:lastevent() as L
9     TvActivationInstant.std:lastevent() as T
10  where L.sensorID = 3 and (T.sensorID = 5 OR
11        T.sensorID=4 OR T.sensorID=6 OR T.sensorID=7)
12    and exists (select *
13        from SitStream.win:keepall() as K
14        where K.initActivation = T.initActivation)
15    and T.aggregated = 0 and L.initActivation -
16       (select K.finActivation
17        from SitStream.win:keepall() as K
18        where K.initActivation=T.initActivation)<=180"
```

Query 63:

```
1 public static String
2 okFromWaitedCouchEventFromCouchEventCaseFive =
3 "insert into GroupedTvStream
4  select C.initActivation as initActivation,
5         C.finActivation as finActivation,
6         1 as approved, (select initActivation
7    from TemporaryTvActivationInstant.std:lastevent())
8         as lastControlInstant
9  from WaitedSitStream.std:lastevent() as W,
10      SitStream.std:lastevent() as C
11 where W.initActivation = C.initActivation
12   and W.pending = 1
13   and 3 = (select sensorID
14   from TemporaryTvActivationInstant.std:lastevent())
15   and (5 = (select sensorID
16        from TvActivationInstant.std:lastevent())
17         OR 4 = (select sensorID
18         from TvActivationInstant.std:lastevent())
19         OR 6 = (select sensorID
20         from TvActivationInstant.std:lastevent())
21         OR 7 = (select sensorID
22         from TvActivationInstant.std:lastevent()))
23   and 0 = (select aggregated
24         from TvActivationInstant.std:lastevent()) "
```

Query 64:

```
1 public static String
2 updateTvActivationAfterCouchCaseFive =
3 "insert into TvActivationInstant
4  select L.eventID as eventID, L.sensorID as sensorID,
5         L.initActivation as initActivation,
6         1 as aggregated, -1 as followingInstant,
7         L.finActivation as finActivation
8from TemporaryTvActivationInstant.std:lastevent() as L
9     , GroupedTvStream.std:lastevent() as G
10  where G.initActivation = (select initActivation
11        from TvActivationInstant.std:lastevent())
12    and 0 = (select aggregated
13        from TvActivationInstant.std:lastevent())
14    and 3 = (select sensorID
15    from TemporaryTvActivationInstant.std:lastevent())
16    and (5 = (select sensorID
17         from TvActivationInstant.std:lastevent())
18         OR 4 = (select sensorID
19         from TvActivationInstant.std:lastevent())
20         OR 6 = (select sensorID
21         from TvActivationInstant.std:lastevent())
22         OR 7 = (select sensorID
23         from TvActivationInstant.std:lastevent()))
24    and L.initActivation NOT IN (select couchExpected
25        from WaitForDayClosing.win:keepall())"
```

Query 65:

```
1 public static String
2 afterCouchEventComesACouchEventNotAggregatedCase =
3 "insert into TvActivationInstant
4  select L.initActivation as initActivation,
5         L.sensorID as sensorID, L.eventID as eventID,
6         -1 as followingInstant, 0 as aggregated
7from TemporaryTvActivationInstant.std:lastevent() as L
8    TvActivationInstant.std:lastevent() as T
9 where (L.sensorID = 5 OR L.sensorID = 4 OR
10        L.sensorID = 6 OR L.sensorID = 7)
11   and (T.sensorID = 5 OR T.sensorID = 4 OR
12        T.sensorID = 6 OR T.sensorID = 7)
13   and L.initActivation != T.initActivation
14   and T.aggregated = 0
15   and L.initActivation != T.initActivation
16   and exists (select * from SitStream.std:lastevent()
17       where initActivation = T.initActivation)
18   and L.initActivation - (select K.finActivation
19       from SitStream.win:keepall() as K
20       where K.initActivation=T.initActivation)<=180 "
```

Query 66:

```
1  public static String
2  afterATvSignalArrivesTvSignalAggregated =
3  "insert into GroupedTvStream
4   select 1 as approved, (select initActivation
5           from GroupedTvStream.std:lastevent())
6           as initActivation,
7          "T.initActivation as finActivation,
8           T.initActivation as lastControlInstant
9  from TemporaryTvActivationInstant.std:lastevent() as T
10     TvActivationInstant.std:lastevent() as L
11   where T.sensorID = 3 and L.sensorID = 3
12     and L.aggregated = 1
13     and T.initActivation != L.initActivation
14     and T.initActivation - (select finActivation
15         from GroupedTvStream.std:lastevent()) <= 180 "
```

Query 67:

```
1  public static String
2  updateAfterArrivalTwoTvSignalsAggregated =
3  "insert into TvActivationInstant
4   select T.eventID as eventID, T.sensorID as sensorID,
5          T.initActivation as initActivation,
6          -1 as followingInstant, 1 as aggregated,
7          T.finActivation as finActivation
8  from TemporaryTvActivationInstant.std:lastevent() as T
9      GroupedTvStream.std:lastevent() as G
10   where T.sensorID = 3 and 3 = (select L.sensorID
11         from TvActivationInstant.std:lastevent() as L)
12     and 1 = (select aggregated
13         from TvActivationInstant.std:lastevent())
14     and G.finActivation = T.initActivation "
```

Query 68:

```
1  public static String okWaitedCouchEventCaseTwo =
2  "insert into GroupedTvStream
3   select (select initActivation
4           from GroupedTvStream.std:lastevent())
5           as initActivation, 1 as approved,
6           C.finActivation as finActivation,
7          (select L.initActivation
8         from TvActivationInstant.std:lastevent() as L)
9           as lastControlInstant
10  from SitStream.std:lastevent() as C,
11       WaitedSitStream.std:lastevent() as W
12  where C.initActivation = W.initActivation
13    and W.pending = 1
14    and 1 = (select aggregated
15        from TvActivationInstant.std:lastevent())
16    and (5 = (select sensorID
17    from TemporaryTvActivationInstant.std:lastevent())
18    and (4 = (select sensorID
19    from TemporaryTvActivationInstant.std:lastevent())
20    and (6 = (select sensorID
21    from TemporaryTvActivationInstant.std:lastevent())
22    and (7 = (select sensorID
23    from TemporaryTvActivationInstant.std:lastevent())
24    and 3 = (select sensorID
25        from TvActivationInstant.std:lastevent())"
```

Query 69:

```
1 public static String
2 updateTvActivationAfteracseTwo =
3 "insert into TvActivationInstant
4  select (select L.initActivation
5from TemporaryTvActivationInstant.std:lastevent() as L)
6        as initActivation, " +
7  select (select L.eventID
8from TemporaryTvActivationInstant.std:lastevent() as L)
9        as eventID
10  select (select L.sensorID
11from TemporaryTvActivationInstant.std:lastevent() as L)
12       as sensorID
13       1 as aggregated, -1 as followingInstant
14  from GroupedTvStream.std:lastevent() as G,
15       SitStream.std:lastevent() as C
16  where G.finActivation = (select C.finActivation
17       from SitStream.std:lastevent() as C)
18    and 1 = (select aggregated
19       from TvActivationInstant.std:lastevent())
20    and (5 = (select sensorID
21    from TemporaryTvActivationInstant.std:lastevent())
22    and (4 = (select sensorID
23    from TemporaryTvActivationInstant.std:lastevent())
24    and (6 = (select sensorID
25    from TemporaryTvActivationInstant.std:lastevent())
26    and (7 = (select sensorID
27   from TemporaryTvActivationInstant.std:lastevent()))
28    and 3 = (select sensorID
29       from TvActivationInstant.std:lastevent())
30   and (select L.initActivation
31from TemporaryTvActivationInstant.std:lastevent() as L)
32     not in (select T.initActivation
33         from TvActivationInstant.win:keepall() as T"
```

Query 70:

```
1 public static String
2 arrivalOfATvSignalAfterCouchAggregatedCaseTwo =
3 "insert into GroupedTvStream
4  select (select initActivation
5          from GroupedTvStream.std:lastevent())
6          as initActivation, 1 as approved,
7          L.initActivation as finActivation,
8          L.initActivation as lastControlInstant
9from TemporaryTvActivationInstant.std:lastevent() as L
10     TvActivationInstant.std:lastevent() as T
11  where L.sensorID = 3 and (T.sensorID = 5
12        OR T.sensorID = 4 OR T.sensorID = 6
13        OR T.sensorID = 7)
14    and exists (select *
15        from SitStream.win:keepall() as K
16        where K.initActivation = T.initActivation)
17    and T.aggregated = 1 and T.addedForDayClosing != 1
18    and L.initActivation - (select K.finActivation
19        from SitStream.win:keepall() as K
20        where K.initActivation=T.initActivation)<=180"
```

Query 71:

```
1 public static String
2 updateAfterTvSignalFollowingCouchAggregatedCaseTwo =
3 "insert into TvActivationInstant
4  select L.eventID as eventID, L.sensorID as sensorID,
5         L.initActivation as initActivation,
6         1 as aggregated, -1 as followingInstant,
7         L.finActivation as finActivation
8from TemporaryTvActivationInstant.std:lastevent() as L
9     GroupedTvStream.std:lastevent() as G
10  where L.initActivation = G.finActivation
11    and L.sensorID = 3
12    and (5 = (select sensorID
13        from TvActivationInstant.std:lastevent())
14         OR 4 = (select sensorID
15         from TvActivationInstant.std:lastevent())
16         OR 6 = (select sensorID
17         from TvActivationInstant.std:lastevent())
18         OR 7 = (select sensorID
19         from TvActivationInstant.std:lastevent()))
20    and 1 = (select aggregated
21        from TvActivationInstant.std:lastevent())"
```

Query 72:

```
1  public static String
2  okFromWaitedCouchEventFromCouchEventCaseFour =
3  "insert into GroupedTvStream
4   select (select initActivation
5           from GroupedTvStream.std:lastevent())
6           as initActivation,
7           C.finActivation as finActivation,
8          (select approved
9           from GroupedTvStream.std:lastevent())
10          as approved,
11         (select lastControlInstant
12          from GroupedTvStream.std:lastevent())
13          as lastControlInstant
14  from SitStream.std:lastevent() as C,
15       WaitedSitStream.std:lastevent() as W
16    where C.initActivation = W.initActivation
17          and W.pending = 1 and (5 = (select sensorID
18    from TemporaryTvActivationInstant.std:lastevent())
19          OR 4 = (select sensorID
20    from TemporaryTvActivationInstant.std:lastevent())
21          OR 6 = (select sensorID
22    from TemporaryTvActivationInstant.std:lastevent
23          OR 7 = (select sensorID
24    from TemporaryTvActivationInstant.std:lastevent())
25     and (5 = (select sensorID
26         from TvActivationInstant.std:lastevent())
27          OR 4 = (select sensorID
28             from TvActivationInstant.std:lastevent())
29          OR 6 = (select sensorID
30             from TvActivationInstant.std:lastevent())
31          OR 7 = (select sensorID
32             from TvActivationInstant.std:lastevent())
33     and 1 = (select aggregated
34         from TvActivationInstant.std:lastevent()) "
```

Query 73:

```
public static String
updateTvActivationAfterGroupingCaseFour =
"insert into TvActivationInstant
 select L.eventID as eventID, L.sensorID as sensorID
        ,L.initActivation as initActivation,
        1 as aggregated, -1 as followingInstant
from TemporaryTvActivationInstant.std:lastevent() as L
    GroupedTvStream.std:lastevent() as G
 where G.finActivation = (select C.finActivation
       from SitStream.std:lastevent() as C
       where C.initActivation = L.initActivation)
   and 1 = (select aggregated
       from TvActivationInstant.std:lastevent())
   and (5 = (select sensorID
    from TemporaryTvActivationInstant.std:lastevent())
         OR 4 = (select sensorID
    from TemporaryTvActivationInstant.std:lastevent())
         OR 6 = (select sensorID
    from TemporaryTvActivationInstant.std:lastevent())
         OR 7 = (select sensorID
  from TemporaryTvActivationInstant.std:lastevent()))
   and (5 = (select sensorID
       from TvActivationInstant.std:lastevent())
        OR 4 = (select sensorID
       from TvActivationInstant.std:lastevent())
        OR 6 = (select sensorID
       from TvActivationInstant.std:lastevent())
        OR 7 = (select sensorID
       from TvActivationInstant.std:lastevent())
   and not exists (select *
       from TvActivationInstant.win:keepall() as T
       where T.initActivation = L.initActivation) "
```

Query 74:

```
public static String sendAllDailyResultsToFile =
"insert into OutputTvStream
 select K.timeWatchingTv as timeWatchingTv,
        K.initActivation as initActivation,
        K.finActivation as finActivation
 from FinalTvStream.std:lastevent() as L,
      FinalTvStream.win:keepall() as K
 where L.dayEnded = 1
   and K.initActivation > L.finActivation -86400 "
```

Query 75:

```
public static String computationOfActiveTimeRatio =
"insert into ActiveTimeRatioPerDay
 select (select (sum(K.timeWatchingTv))/864
         from FinalTvStream.win:keepall() as K
         where K.initActivation != L.initActivation
         and Math.floor((K.initActivation-1)/86400)=
          Math.floor((L.initActivation-1)/86400))
   + (select (max(K.timeWatchingTv))/864
      from FinalTvStream.win:keepall() as K
      where K.initActivation = L.initActivation)
    as activeTimeRatio,
    Math.floor((L.initActivation-1)/86400)+1
    as dayAnalyzed
 from FinalTvStream.std:lastevent() as L
 where L.dayEnded = 1 "
```

B3 Interrogazioni ISVIS

In questa sezione vengono introdotte le interrogazioni impiegate per lo sviluppo dell'indicatore presentato nella Sezione 5.3 del Capitolo 5.

Query 76:

```
public static String activationEvent =
"insert into NewSensorStatus
 select Math.pow(2.0,a.sensorID)
        + (select L.status
        from NewSensorStatus.std:lastevent() as L)
        as status,
 a.timestamp+1 as timestamp
 from pattern [every (a = SensorStream)]
 where a.status = 1 "
```

Query 77:

```
public static String deactivationEvent =
"insert into NewSensorStatus
 select (-1)*Math.pow(2.0,a.sensorID) +
         + (select L.status
         from NewSensorStatus.std:lastevent() as L)
         as status,
         a.timestamp+1 as timestamp
 from pattern [every (a = SensorStream)]
 where a.status = 0 "
```

Query 78:

```
1  public static String
2  eventsConfigurationInformation =
3  "insert into SensorConfiguration
4   select (select L.status
5           from NewSensorStatus.std:lastevent() as L)
6           as status,
7          1 + (a.timestamp - (select L.timestamp
8          from NewSensorStatus.std:lastevent() as L))
9          as durationEvent,
10        (select L.timestamp
11         from NewSensorStatus.std:lastevent() as L)
12         as initInstant,
13         a.timestamp as finInstant
14 from pattern [every (a = SensorStream)]
15 where a.timestamp+1
16       not in (select timestamp
17       from NewSensorStatus.win:keepall())
18 and a.timestamp != 0"
```

Query 79:

```
1  public static String computationOfDurations =
2  "insert into HouseEntropy " +
3   select entropyComputation(K)
4   from SensorConfiguration.std:lastevent() as L,
5        SensorConfiguration.win:time(3600) as K
6   where K.finInstant > L.finInstant-3600 "
```

Query 80:

```
1 public static String sendHouseEntropyResultsToFile =
2 "insert into OutputHouseStream
3  select L.entropy as dailyHouseEntropy,
4         Math.floor(L.timestamp/86400) as dayAnalyzed
5   from HouseEntropy.std:lastevent() as L
6   where (Math.floor(L.timestamp/86400) =
7         1 + (select T.dayAnalyzed
8         from OutputHouseStream.std:lastevent() as T )
9      or not exists (select *
10        from OutputHouseStream.win:keepall()))"
```

Bibliografia

[1] Asaf Adi and Opher Etzion, "Amit - the situationManager," *The VLDB Journal — The International Journal on Very Large Data Bases*, vol. 13, no. 2, pp. 177-203, Maggio 2004.

[2] Ryan Aipperspach, Elliot Cohen, and John Canny, "Modeling Human Behavior from Simple Sensors in the Home," in Pervasive Computing., 2006, pp. 337-348.

[3] Mert Akdere, Uğur Çetintemel, and Nesime Tatbul, "Plan-based complex event detection across distributed sources," in *Proceedings of the VLDB Endowment*, 2008, pp. 66-77.

[4] Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy, "ARAS human activity datasets in multiple homes with multiple residents," in *PervasiveHealth '13 Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare* , 232-235, p. 2013.

[5] Elias Alevizos and Alexander Artikis, "Being Logical or Going with the Flow? A Comparison of Complex Event Processing Systems," in *Artificial Intelligence: Methods and Applications*., 2014, pp. 460-474.

[6] Alexander Artikis, M Sergot, and G Paliours, "Run-time composite event recognition," in *In Proceeding International Conference on Distributed Event-Based Systems (DEBS)*, 2012, pp. 69-80.

[7] Alexander Artikis, M Weidlich, A Gal, V Kalogeraki, and D Gunopulos, "Self-adaptive event recognition for intelligent transport management," in *IEEE International Conference on Big Data*, Silicon Valley, 2013, pp. 319-325.

[8] Assistive Technology Group (ATG) of Politecnico di Milano. [Online]. http://atg.deib.polimi.it

[9] Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang, "Sensor Positioning for Activity Recognition Using Wearable Accelerometers," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 4, pp. 320-329, Agosto 2011.

[10] Louis Atallah and Guang-Zhong Yang, "Review: The use of pervasive sensing for behaviour profiling - a survey," *Pervasive and Mobile Computing*, vol. 5, no. 5, pp. 447-464, Ottobre 2009.

[11] Seung-Ho Baeg, Jae-Han Park, Jaehan Koh, Kyung-Woo Park, and Moon-Hong Baeg, "Building a Smart Home Environment for Service Robots Based on RFID and Sensor Networks," in *International Conference on Control, Automation and Systems 2007*, Seoul, 2007, pp. 1078-1082.

[12] Roland Balter, "JORAM: The Open Source Enterprise Service Bus," ScalAgent Distributed Technologies SA, Échirolles, 2004.

[13] Jakob Bardram, "Hospitals of the Future - Ubiquitous Computing support for Medical Work in Hospitals.," in *In UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, Seattle, 2003.

[14] Roger Barga, Jonathan Goldstein, Mohamed Ali, and Mingsheng Hong, "Consistent streaming through time: A vision for event stream processing," *arXiv preprint cs/0612115*, Dicembre 2006.

[15] Boğaziçi University Department of Computer Engineering. (2013) Sito Web ARAS datasets. [Online]. http://www.cmpe.boun.edu.tr/aras/

[16] Lars Brenna et al., "Cayuga: a high-performance event processing engine," in *SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data* , 2007, pp. 1100-1102.

[17] Juan Luis Carús Candás et al., "An automatic data mining method to detect abnormal human behaviour using physical activity measurements," *Pervasive and Mobile Computing*, vol. 15, pp. 228–241, 2014.

[18] Antonio Carzaniga, Alexander Wolf, and David Rosenblum, "Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service," in *PODC '00 Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing* , Portland, 2000, pp. 219-227.

[19] CASAS Project. (2009) Sito Web Progetto CASAS. [Online]. http://ailab.wsu.edu/casas/datasets.html

[20] Sharma Chakravarthy and Deepak Mishra, "Snoop: An Expressive Event Specification Language For Active Databases," 1993.

[21] Raphael Chand and Pascal Felber, "XNET: a reliable content-based publish/subscribe system," in *In Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, Ottobre 2004, pp. 264-273.

[22] Michael Coen, "Design principles for intelligent environments," in *AAAI Spring Symposium*, Standford, 1998, pp. 547-554.

[23] Commissione Europea. (2013, Maggio) Decisioni della Commissione Europea della qualità ecologica (Ecolabel EU) di rubinetteria. [Online]. http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri= OJ:L:2013:145:0006:0030:IT:PDF

[24] Commissione Europea. (2013, Novembre) Decisioni della Commissione Europea della qualità ecologica (Ecolabel EU) di vasi sanitari e scarico d'acqua. [Online]. http://www.isprambiente.gov.it/it/certificazioni/files/ecolabel/criteri/criteri-per-vasi-sanitari-2013-641-ue

[25] Diane Cook, "Learning Setting-Generalized Activity Models for Smart Spaces," *IEEE Intelligent Systems*, vol. 2010, no. 9, pp. 1-13, Settembre 2011.

[26] Diane Cook and Sajal Das, *Smart Environments: Technology, Protocols and Applications*.: John Wiley, 2004.

[27] CRAiS. CRAiS. [Online]. http://www.crais.eu/

[28] Gianpaolo Cugola and Alessandro Margara, "Complex event processing with T-Rex," *Journal of Systems and Software*, vol. 85, no. 8, pp. 1709–1728, Agosto 2012.

[29] Gianpaolo Cugola and Alessandro Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, 2012.

[30] Gianpaolo Cugola and Alessandro Margara, "Raced: an adaptive middleware for complex event detection," in *Proceedings of the 8th International Workshop on Adaptive and Reflective MIddleware*, 2009, p. 5.

[31] Gianpaolo Cugola and Alessandro Margara, "TESLA: a formally defined event specification language," in *DEBS '10 Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, 2010, pp. 50-61.

[32] Sajal Das and Diane Cook, "Designing and Modeling Smart Environments," *WOWMOM '06 Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks* , pp. 490-494, 2006.

[33] Sajal Das and Diane Cook, "The Role of Prediction Algorithms in the MavHome Smart Home Architecture," *IEEE Wireless Communications*, vol. 9, no. 6, pp. 77-84, Dicembre 2002.

[34] Paul Dekkers, M van Vliet, P Ligthart, H de Man, and G Ligtenberg, "Complex Event Processing," Nijmegen, 2007.

[35] George Demiris, Brian Hensel, Marjorie Scubic, and Marylin Rantz, "Senior residents' perceived need of and preferences for "smart home" sensor technologies," *International Journal of Technology Assessment in Health Care*, vol. 24, no. 1, pp. 120-124, 2008.

[36] Nathan Eagle, "Using mobile phones to model complex social systems," 2005.

[37] Nathan Eagle and Alex Pentland, "Reality mining: sensing complex social systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255-268, 2006.

[38] EsperTech Inc. (2008, Febbraio) Sito Web EsperTech Inc. [Online]. http://www.espertech.com/

[39] Partick Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114-131, Giugno 2003.

[40] J Flachsbart, D Franklin, and K Hammond, "Improving Human Computer Interaction in a Classroom Environment using Computer Vision," in *In Proceedings of the Conference on Intelligent User Interfaces*, New York, 2000.

[41] Jeff Foley and Gavin Churcher, "Applying Complex Event Processing and Extending Sensor Web Enablement to a Health Care Sensor Network Architecture," in *London Communications Symposium*, Londra, 2009, pp. 1-10.

[42] David Franklin, "Cooperating with People: The Intelligent Classroom.," in *In Proceeding of the Fifteenth National Conference in Artificial Intelligence*, Chicago, 1998.

[43] Mathieu Gallissot, Jean Caelen, Nicolas Bonnefond, Brigitte Meillon, and Sylvie Pons, "Using the Multicom Domus Dataset," Grenoble, 2011.

[44] Leonid A. Gavrilov and Patrick Heuveline, "Ageing of population," *The encyclopedia of population*, vol. 1, no. 1, pp. 32-37, Dicembre 2003.

[45] Carola Graf, "Katz Index of Independence in Activities of Daily Living (ADL)," *try this: Best Practices in Nursing Care to Older Adults*, vol. 108, no. 4, 2008.

[46] Giancarlo Gualtieri and Sabrina Prati. (2015, Novembre) ISTAT. [Online]. http://www.istat.it/it/files/2015/11/Natalit%C3%A0_fecondita_2014.pdf?title=Natalit%C3%A0+e+fecondit%C3%A0+-+27%2Fnov%2F2015+-+Testo integrale.pdf

[47] Gurav Jain and Diane Cook, "Monitoring Health by Detecting Drifts and Outliers in Patterns of an Inhabitant in a Smart Home," Texas, 2006.

[48] Ashwin Jayaprakash. (2007) Sito Web StreamCruncher. [Online]. http://javaforu.com/

[49] Brad Johnson, Armando Fox, and Terry Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms," *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 67-74, Aprile 2002.

[50] N Khoussainova, M Balazinska, and D Suciu, "Probabilistic Event Extraction from RFID Data," in *ICDE 2008. IEEE 24th International Conference on Data Engineering*, 2008, pp. 1480-1482.

[51] R Kowalski and M Sergot, "A logic-based calculus of events," *New Generation Computing*, vol. 4, no. 1, pp. 67-95, 1986.

[52] Christophe Le Gal, Jerome Martin, Augustin Lux, and James Crowley, "SmartOffice: Design of an intelligent environment," *IEEE Intelligent Systems*, vol. 16, no. 4, pp. 60-66, 2001.

[53] Victor Lesser et al., "The Intelligent Home Testbed," in *Proceedings of the Autonomy Control Software Workshop*, Seattle, 1999.

[54] Guoli Li and Hans-Arno Jacobsen, "Composite subscriptions in content-based publish/subscribe systems," in *Middleware '05 Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, 2005, pp. 249-269.

[55] Shane Lowe and Gearóid ÓLaighin, "Monitoring human health behaviour in one's living environment: a technological review," *Medical Engineering & Physics*, vol. 36, pp. 147-168, 2014.

[56] David Luckham, "Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events," in *DIMACS Partial Order Methods Workshop IV*, 1996.

[57] David Luckham and James Vera, "An Event-Based Architecture Definition Language," *IEEE Transactions on Software Engineering*, vol. 21, no. 9, pp. 717-734, Settembre 1995.

[58] Simone Mangano et al., "BRIDGe: a Project for the Mutual Reassurance for Autonomous and Independent Living," *Intelligent Systems, IEEE*, vol. 30, no. 4, pp. 31-38, 2015.

[59] Masoud Mansouri-Samani and Morris Sloman, "GEM: A generalized event monitoring language for distributed systems," *Distributed Systems Engineering*, vol. 4, no. 2, p. 96, Giugno 1997.

[60] Masoud Mansouri-Samani and Morris Sloman, "Monitoring distributed systems," *Network, IEEE*, vol. 6, no. 7, pp. 20-30, Novembre 1993.

[61] Andrea Masciadri, Anna Trofimova, Matteo Matteucci, and Fabio Veronese, "Unsupervised Methods for Activities of a Daily Living Drift Modeling and Recognition," 2015.

[62] Michael Mozer, "The Neural Network House: An Environment that Adapts to its Inhabitants," in *Proceedings of the American Association for Artificial Intelligence Spring Symposium*, Melro Park, CA, 1998, pp. 110-114.

[63] Gero Mühl, Ludger Fiege, and Peter Pietzuch, *Distributed Event-Based Systems.*: Springer-Verlag Berlin Heidelberg, 2006.

[64] Nazioni Unite, "World population ageing," Affari economici e sociali, New York, 2015.

[65] Robert Orr and Gregory Abowd, "The smart floor: a mechanism for natural user identification and tracking," in *CHI'00 extended abstracts on Human factors in computing systems*, New York, 2000, pp. 275-276.

[66] Alex Pentland, "Automatic mapping and modeling of human networks," *Physica A: Statistical Mechanics and its Applications*, vol. 378, no. 1, pp. 59-67, 2007.

[67] Peter Pietzuch, Brian Shand, and Jean Bacon, "A framework for event composition in distributed systems," in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, 2003, pp. 62-82.

[68] Daniele Proserpio, Fabio Salice, and Fabio Veronese, "SHARON: Simulator of Human Activities, ROutines and Needs: Analysis, Implementation and Validation," 2014.

[69] Daniel Romero et al., "RESTful Integration of Heterogeneous Devices in Pervasive Environments," in *Distributed Applications and Interoperable Systems*. Amsterdam: Springer Berlin Heidelberg, 2010, pp. 1-14.

[70] Shirley Ruder, "The challanges of a family member caregiver: how the home health and hospice can help at the end of life," *Home Healthcare Nurse*, no. 26, pp. 131-136, Marzo 2008.

[71] Nicholas Schultz-Møller, Matteo Migliavacca, and Peter Pietzuch, "Distributed complex event processing with query rewriting," in *In Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, 2009, pp. 1-12.

[72] Marco Seiriö. (2013) Sito Web RuleCore. [Online]. http://sourceforge.net/projects/rulecore/

[73] Robert Strom et al., "Gryphon: An information flow based approach to message brokering," *arXiv preprint cs/9810019*, Ottobre 1998.

[74] Nagender Kumar Suryadevara and Subhas Chandra Mukhopadhyay, "Wireless Sensor Network Based Home Monitoring System for Wellness Determination of Elderly," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1965-1972, Giugno 2012.

[75] Nagender Kumar Suryadevara, Subhas Chandra Mukhopadhyay, Ruili Wang, and Ramesh Rayudu, "Forecasting the behavior of an elderly using wireless sensors data in a smart home," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2641-2652, Novembre 2013.

[76] Sándor Szalai, *The Use of Time: Daily Activities of Urban and Suburban Populations in Twelve Countries*. The Hague: Mouton, 1975.

[77] Ali Maleki Tabar, Arezou Keshavarz , and Hamid Aghajan, "Smart home care network using sensor fusion and distributed vision-based reasoning," in *VSSN '06 Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks* , New York, 2006, pp. 145-154.

[78] Emmaniel Tapia. (2015) Sito Web progetto MIT. [Online]. http://courses.media.mit.edu/2004fall/mas622j/04.projects/home/

[79] Emmanuel Tapia, Stephen Intille, and Kent Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," *Pervasive Computing*, pp. 158-175, 2008.

[80] Can Tunca, Hande Alemdar, Halil Ertan, Dumaz Incel Ozlem, and Cem Ersoy, "Multimodal Wireless Sensor Network-Based Ambient Assisted Living in Real Homes with Multiple Residents," *Sensors*, vol. 14, no. 6, pp. 9692-9719, 2014.

[81] Université de Sherbrooke. (2011) Sito Web Progetto Domus.

[82] Tim van Kasteren. Sito Web progetto Kasteren. [Online]. https://sites.google.com/site/tim0306/datasets

[83] Tim van Kasteren, Athanasios Nolas, and Gwenn Englebie, "Human activity recognition from wireless sensor network data: Benchmark and software.," *Activity Recognition in Pervasive Intelligent Environments*, pp. 165–186, 2011.

[84] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91-102, 1992.

[85] Eugene Wu, Yanlei Diao, and Shariq Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006, pp. 407-418.

[86] Che-Chang Yang and Yeh-Liang Hsu, "Remote monitoring and assessment of daily activities in the home environment," *Journal of Clinical Gerontology & Geriatrics*, vol. 3, no. 2, pp. 1-8, 2012.

[87] Michael Youngblood, Diane Cook, and Lawrence Holder, "Managing adaptive versatile environments," *Pervasive and Mibile Computing*, vol. 1, no. 4, pp. 373-403, 2005.