Solar Panel Analyzer

DRS – Design Requirement Specification Document

Corso di **Software Engineering**
Unige - Università degli Studi di Genova
Dibris - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi

Cliente:
Wesii s.r.l.

Versione: 2.0

Data:
14/05/2018

Autore:
Jacopo De Luca

Revision History

Versione	Data	Autore	Note
1.0	03/05/2018	Jacopo De Luca	Prima stesura
1.1	05/05/2018	Jacopo De Luca	Correzioni
1.2	08/05/2018	Jacopo De Luca	Correzioni
2.0	14/05/2018	Jacopo De Luca	Seconda stesura
2.1	16/05/2018	Jacopo De Luca	Rifinitura

Sommario

1. Introduzione

1.1 Scopo del documento

Questo documento (DRS) è richiesto nel corso di Software Engineering (SE) e simula una pratica comune nei processi di Ingegneria del Software.
Dopo la stesura dell'URD, l'ingegnere informatico redige questo documento, nel quale viene descritta l'architettura del sistema per gli sviluppatori, progettisti e tester. Alcune parti, ad esempio l'interfaccia grafica, possono essere esposte agli utenti del sistema, non essendo necessarie competenze tecniche ed essendo questi fruenti del medesimo.
Come già esposto nel URD, il software sarà un tool di riconoscimento e visualizzazione di pannelli solari e verrà sviluppato per Wesii s.r.l.
Il presente DRS è redatto da Jacopo De Luca, studente di Laurea Magistrale in Ingegneria Informatica presso l'Università degli Studi di Genova, il quale progetterà e svilupperà il software in questione; la parte di testing, invece, verrà affrontata da un collega iscritto al medesimo corso di SE.

1.2 Descrizione generale del sistema

Il progetto si propone di fornire un'automatizzazione del processo di riconoscimento di ogni singolo pannello solare all'interno di impianti fotovoltaici. Il sistema quindi è composto da un'applicazione stand-alone che governa una procedura divisa in 3 fasi: inserimento input, processo di riconoscimento dei pannelli e visualizzazione degli output. Le immagini e i dati vengono salvati nel pc in uso, non è quindi previsto un database. L'utente deve fornire le immagini, compilare una maschera di settaggi e far partire l'elaborazione; quest'ultima, ovvero il cuore del software, segue un algoritmo di image processing ed estrae i pannelli solari riconosciuti utilizzando le librerie OpenCV (computer vision); infine gli output dovrebbero essere resi visibili in una sezione del software (priorità 'Desirable').
Il software è scritto in Python3 e deve essere utilizzabile in ambiente Linux.

1.3 Definizioni e acronimi

Acronimo - Nome	Definizione
SE	Software Engineering
JD	Jacopo De Luca
SPA	Solar Panel Analyzer
DRS	Design Requirement Specification Document
URD	User Requirements Document
SO	Sistema Operativo

1.4 Bibliografia

De Luca J., URD di Solar Panel Analyzer (SPA), Chiavari, 2018
Narizzano M., Slide del Corso di SE, Genova, 2018
Python 3.6 documentation (e successive)
OpenCV (Open Source Computer Vision Library) documentation

1.5 Overview del documento

Nella sezione due viene presentata una descrizione ad alto livello dell'Applicazione. Nella sezione tre vengono descritte le interfacce del sistema mentre nella sezione quattro viene descritto come vengono memorizzati i dati. La sezione cinque descrive la struttura del sistema, mentre nella sezione sei vengono descritti in dettaglio come i requisiti sono implementati.

2. Descrizione del progetto

2.1 Introduzione

In questo progetto verrà adottato un approccio object oriented con un'architettura desktop. Per disegnare i diagrammi UML è stato utilizzato il tool Dia.

2.2 Architettura del sistema

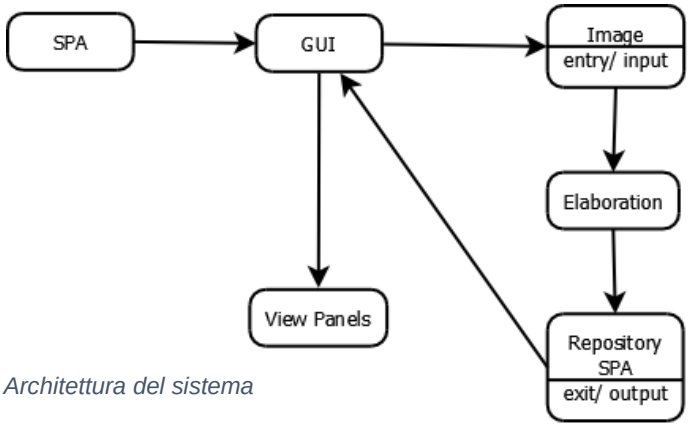

Figura 1. Architettura del sistema

In Figura 1 viene presentata una possibile architettura ad alto livello del sistema. L'applicazione è stand-alone e tutte le immagini e i dati sono salvati in locale nel pc in uso (non è previsto alcun database).

SPA è l'applicazione.
L'utente si interfaccia tramite una GUI che permette l'avvio del processo di riconoscimento dei pannelli solari, suddiviso in inserimento input, elaborazione e generazione output. I panelli ottenuti sono visualizzabili sempre tramite la GUI, attraverso la funzionalità "View panels".

Vincoli e assunzioni

Si assume che le immagini in input siano esclusivamente fornite dal volo di un drone di WeSii s.r.l, pertanto devono contenere impianti fotovoltaici. La prima assunzione è quindi che SPA si aspetta sempre immagini raffiguranti pannelli solari perché il software esegue un'analisi geometrica delle immagini e non utilizza algoritmi di object detection.
SPA potrebbe necessitare di requisiti minimi del SO (molto dipende dalla grandezza delle immagini), ma in questa fase di progettazione non è ancora possibile identificarli.
Per il corretto funzionamento, il SO deve inoltre essere dotato degli strumenti necessari, quali Python e OpenCV oltre ad alcune altre librerie.

3. Interfacce

3.1 GUI di Solar Panel Analyzer (SPA)

Interfaccia generale del sistema.

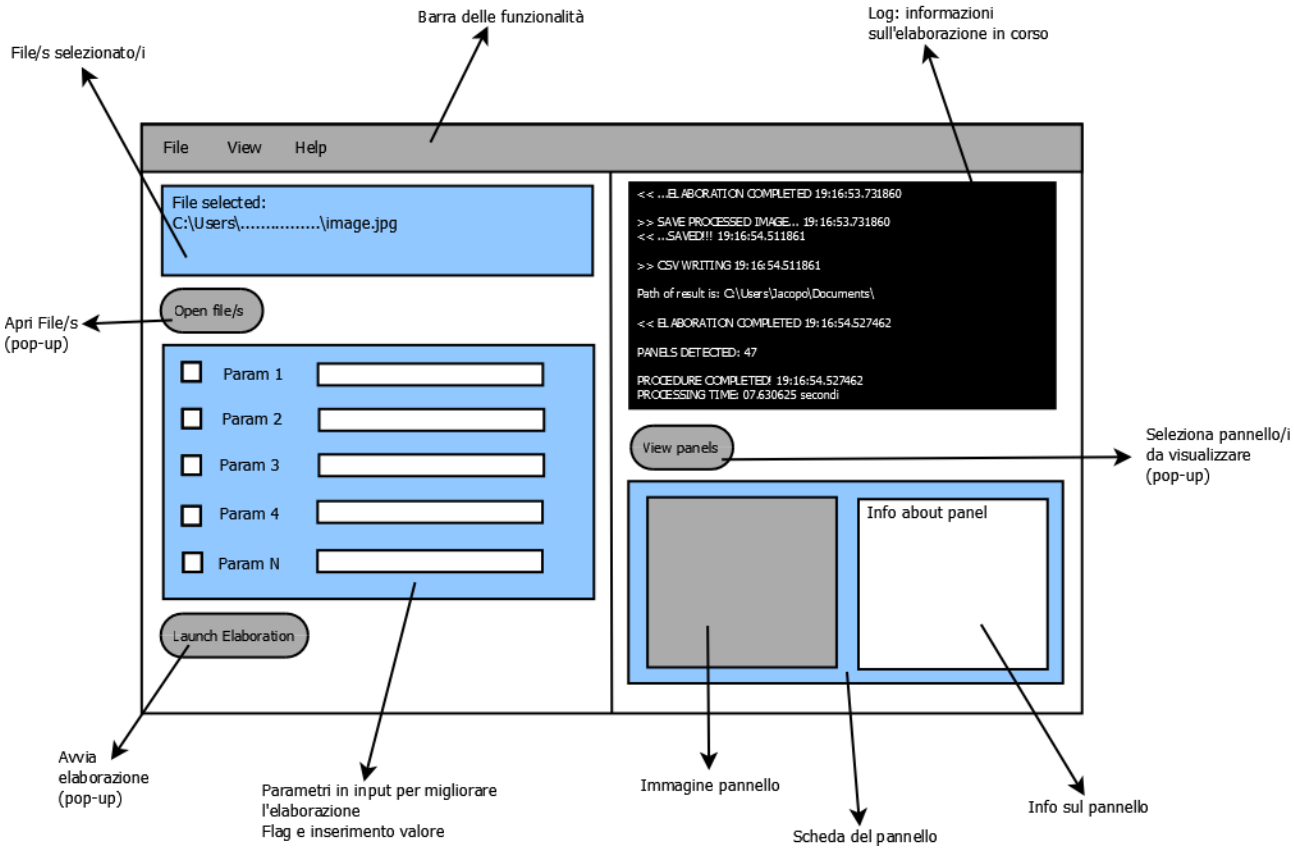

Figura 2. GUI di Solar Panel Analyzer (SPA)

L'interfaccia di SPA è essenziale.
Nella barra in alto si possono trovare tutte le caratteristiche del software: cliccando su qualsiasi voce si aprirà un menu a tendina tipico dei SO dal quale selezionare la funzionalità. In alternativa l'utente può servirsi dell'interfaccia molto semplice e immediata composta da pannelli informativi (file selezionati) e/o maschere (come il form per l'immissione dei parametri in input) e pulsanti.

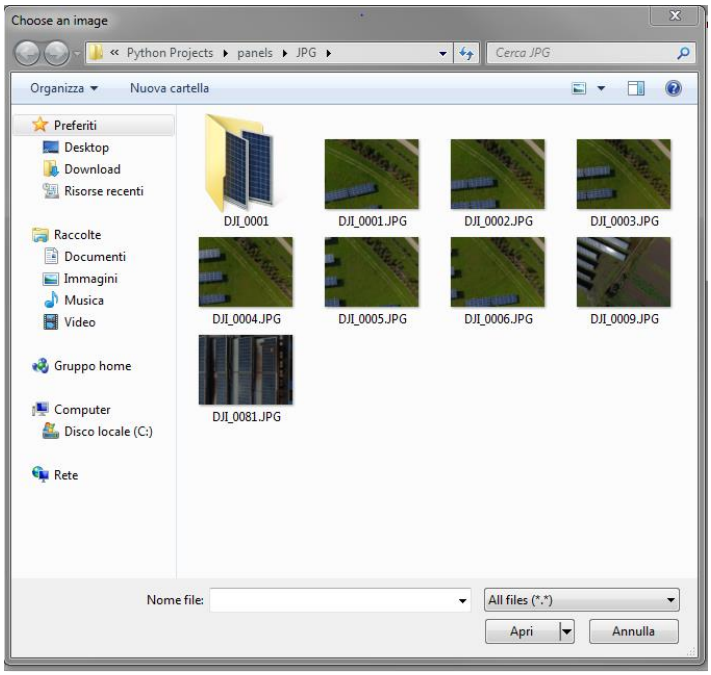

Figura 3. Esempio di pop-up per selezione immagine/i

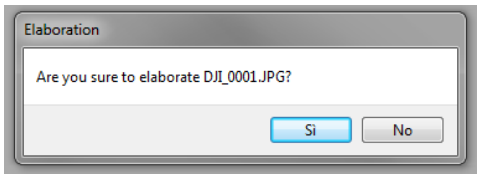

Figura 4. Esempio di pop-up per avvio elaborazione

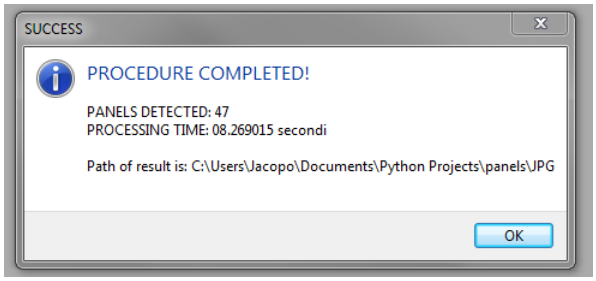

Figura 5. Esempio di pop-up di elaborazione completata

Alcune funzionalità del software prevedono l'utilizzo di finestre pop-up, ad esempio per la scelta delle immagini, per la conferma di avvio elaborazione o l'avviso di completamento della procedura.
Anche in questo caso si utilizzano le finestre di default del SO.
Di seguito saranno considerate in dettaglio le varie sezioni dell'interfaccia generale mostrata in Figura 2.

3.2 Sezione "File/s selezionato/i" e pulsante "Open file/s"

L'utente può inserire le immagini da elaborare direttamente con il pulsante "Open file/s", oppure utilizzando la voce "File" nella barra in alto e scegliendo la funzionalità corrispondente. In ogni caso si aprirà un pop-up per la scelta del file, o dei files, da processare.
Nella sezione dedicata all'interno GUI, l'utente può controllare quante e quali immagini sono state selezionate, quindi pronte all'elaborazione.

3.3 Sezione "Parametri" e pulsante "Launch Elaboration"

L'utente può inserire alcuni parametri per rendere più preciso l'algoritmo di riconoscimento dei pannelli solari. Nella sezione dedicata della GUI è possibile flaggare il parametro da considerare durante il processing e inserire il suo valore nella casella corrispondente. Se nessuno parametro viene flaggato, l'elaborazione sarà standard (e potrebbe essere imprecisa).
I parametri vengono letti e salvati alla pressione del tasto "Launch Elaboration" (oppure utilizzando la barra in alto); dopo aver confermato l'avvio della procedura con i valori indicati nel pop-up di controllo, il software procede con l'algoritmo di riconoscimento dei pannelli solari.

3.4 Sezione "Log"

Durante l'elaborazione SPA rende disponibile all'utente un pannello dei "Log" nel quale vengono stampati i progressi dell'elaborazione e alcuni valori chiave (come il numero dei pannelli solari identificati o la durata dei vari step): da questa l'utente può controllare l'andamento della procedura.

3.5 Sezione “Scheda del pannello” e pulsante “View panels”

Per visualizzare l’output della nostra elaborazione, l’utente può servirsi della sezione “Scheda del pannello”.
Attraverso il pulsante “View panels” (oppure dalla barra in alto), si può decidere quale impianto fotovoltaico processato visualizzare, ossia le cartelle nella directory di SPA: la GUI mostrerà un solo pannello alla volta di fianco a un elenco di sue caratteristiche. Ovviamente il software renderà possibile lo scorrimento delle altre immagini e dati corrispondenti dello stesso impianto.

4 Data stores

L’output dell’elaborazione, quindi le immagini e i dati (scritti in file descrittivi CSV e/o XML), sono salvati sul pc in uso all’interno di cartelle nella directory di default di SPA, la stessa in cui successivamente verranno ricercati per permettere la funzionalità “View panels” di cui prima.

5 Use cases

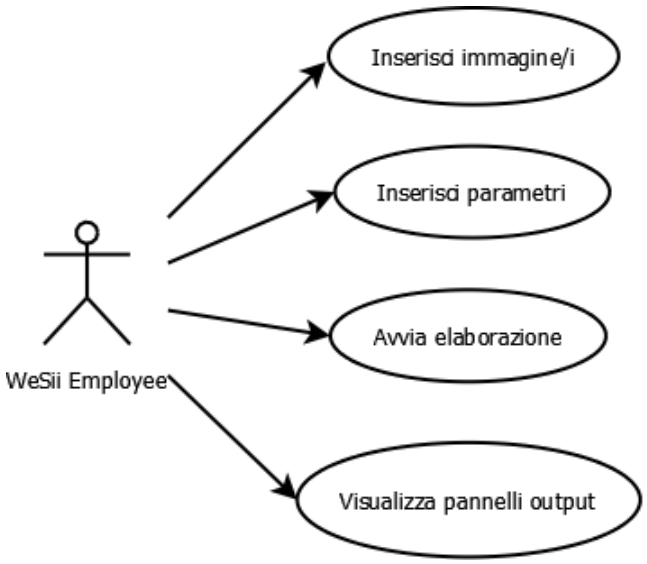

Figura 6. Diagramma Use Cases

In questa sezione si descrivono le funzionalità dell’applicazione e i ruoli dell’utente e del sistema stesso, definendo chi fa cosa.

Esiste solo un utente, ovvero l’operatore di WeSii, che utilizza il software sul proprio pc. In realtà all’interno dell’applicazione non esiste alcuna classe utente, ma l’operatore WeSii interagisce con il sistema SPA attraverso la GUI: nel diagramma si esprime la possibilità di inserire l’input, i parametri, avviare l’elaborazione e visualizzare l’output da parte dell’operatore, ma in realtà sono funzioni del sistema. Quindi il sistema governa tutte le funzionalità automatizzate: in primis avvia l’operazione richiesta dall’utente tramite GUI, riconosce il formato input ed eventualmente avvia una conversione, segue la procedura di riconoscimento dei pannelli, effettua continui controlli per migliorare l’identificazione e il conseguente output, che viene infine salvato.

6 Structural Design

In questa sezione si descrive il sistema in generale attraverso grafici strutturali quali diagramma delle classi e diagramma degli oggetti.

6.1 Class Diagram

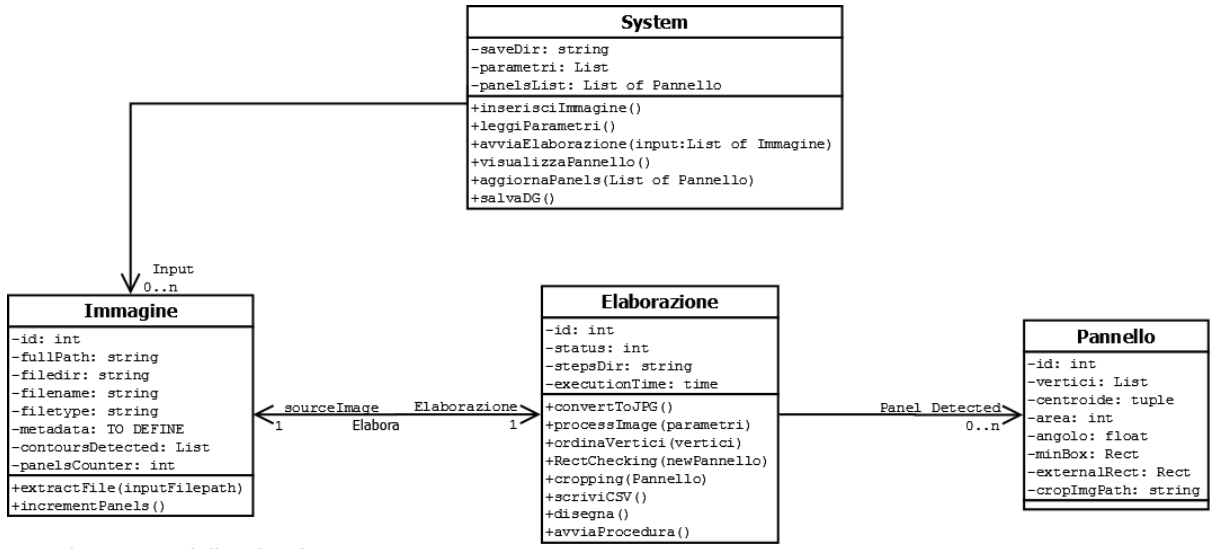

Figura 7. Diagramma delle Classi

6.1.1 System

System è il cuore di SPA. Rappresenta l'applicazione che governa i processi automatizzati.
L'utente fruisce delle funzionalità del software attraverso l'interfaccia di cui sopra.
La responsabilità di questa classe risiede nel ricevere le richieste dell'utente tramite GUI ed eseguirle.

6.1.1.1 Attributi

- *saveDir* di tipo String
  Contiene il percorso della directory di default dove verrà salvato l'output delle elaborazioni.

- *parametri* di tipo List of Array
  Questa lista di array contiene i valori dei parametri di elaborazione. La dimensione della lista è nota e fissa, valori sono inizializzati a None e si aggiornano alla pressione del bottone di avvio elaborazione.

- *panelsList* di tipo List of *Pannello*
  Lista che contiene i pannelli riconosciuti nell'impianto in elaborazione. Siccome un pannello può essere ripreso in due foto diverse dello stesso impianto, *panelsList* è l'attributo che, confrontando i pannelli riconosciuti ad ogni elaborazione, li contiene senza doppioni

- *input* di tipo List of *Immagine*
  List di oggetti di tipo *Immagine*, ovvero gli input delle elaborazioni, inizialmente a None e successivamente aggiornata. L'utente può scegliere la/e immagine/i da elaborare attraverso la GUI.

6.1.1.2 Metodi

- *inserisciImmagine()*
  Questa funzione è collegata al corrispondente pulsante dell'interfaccia. L'inserimento dell'immagine/i, tramite pop-up, crea un nuovo oggetto *Immagine* e aggiorna la List.

- *leggiParametri()*
  Questo metodo viene richiamato automaticamente alla pressione del pulsante di avvio elaborazione. L'utente può inserire i *parametri* direttamente all'interno della GUI che vengono letti e salvati quando si lancia l'elaborazione.

- *avviaElaborazione(input)*
  Alla pressione del tasto corrispondente si leggono i *parametri* tramite la funzione *leggiParametri()* e si apre un pop-up di conferma della richiesta. Se l'utente procede, viene creato un oggetto *Elaborazione* e si richiama la procedura di riconoscimento dei pannelli nella *Immagine input*, considerando i *parametri* inseriti.

- *visualizzaPannello()*
  Questo metodo è collegato al bottone "View panels". L'utente può scegliere quale pannello o quale impianto visualizzare: SPA andrà a leggere il contenuto della cartella selezionata e renderà visibili i pannelli, ovvero le immagini croppate, con i corrispondenti dati, ottenuti dai file descrittivi.

- *aggiornaPanels(output Elaboration)*
  Per ogni immagine in input viene eseguita un'elaborazione, il cui output è una lista di oggetti di tipo *Pannello*, ossia i pannelli identificati nella stessa. Poiché un pannello può essere immortalato su due (o più) immagini diverse, questo metodo è necessario per confrontare ogni volta l'output con *panelsList*, tenendo aggiornato quindi l'elenco dei pannelli dell'impianto, evitando doppioni. Il funzionamento del confronto è ancora da definire.

- *salvaDG()*
  Crea un file descrittivo DG (vedi URD), da definire ancora il formato, con le informazioni dei pannelli dell'impianto (ovvero *panelsList)*.

6.1.2 Immagine

Immagine rappresenta un'immagine in input, ovvero un classe-contenitore delle informazioni riguardanti l'immagine selezionata. SPA organizza le immagini in liste.

6.1.2.1 Attributi

- *id* di tipo int
  Identificativo univoco.

- *fullPath* di tipo String
Contiene il percorso dell'immagine

- *filename* di tipo String
  Contiene il nome del file, estratto da *fullPath* attraverso il metodo *extractFile()*.

- *filedir* di tipo String
  Contiene la directory in cui il file è contenuto, estratta da *fullPath* attraverso il metodo *extractFile()*.

- *filetype* di tipo String
  Contiene l'estensione del file, estratta da *fullPath* attraverso il metodo *extractFile()*.

- *metadata*
  Contiene i metadati del file. Alcuni di essi forniscono informazioni importanti, se non addirittura fondamentali. Per il momento non è ancora chiaro come questi metadati siano scritti sul file input e come estrarli, di conseguenza le indicazioni riguardo l'utilizzo di questo attributo e di un metodo per estrarre il tutto sono incomplete.

- *contoursDetected* di tipo List
  Inizialmente a None, questo attributo si aggiorna durante l'elaborazione e contiene i contorni dei pannelli identificati nell'immagine.

- *panelsCounter* di tipo int
  È un contatore dei pannelli identificati.

- Questa classe ha inoltre bisogno di un riferimento all'*Elaborazione*. Essendo in rapporto 1:1, è necessario un attributo contenente l'id dell'elaborazione corrispondente.

6.1.2.2 Metodi

- *extractFile(inutFilepath): filedir, filename, filetype*
  Questo metodo prende in input il percorso del file da elaborare e restituisce i valori di *filedir*, *filename* e *filetype*.

- *incrementPanels()*
  Ogni volta che durante l'elaborazione dell'immagine viene identificato un nuovo pannello solare, viene incrementato il contatore.

- *Metadata()*
  Molto probabilmente sarà necessario almeno un metodo per gestire i metadati.
  Per il momento non è possibile fornire ulteriori informazioni, per quanto detto sopra.

6.1.3 Elaborazione

Questa classe rappresenta la procedura di riconoscimento dei pannelli solari all'interno delle immagini in input. Comprende un'eventuale conversione in JGP del file in ingresso, processi di image processing con valori fissati e i parametri inseriti dall'utente, la "panel detection" (basata sulla geometria e non su algoritmi di object detection), controlli sulla correttezza dell'output che si sta generando (possibile anche eventuale algoritmo di machine learning per migliorare l'elaborazione), cropping, scrittura CSV ed altri eventuali file utili, finalizzazione output e salvataggio nel pc.

6.1.3.1 Attributi

- *id* di tipo int
  Identificativo univoco.

- *sourceImage* di tipo *Immagine*
  Immagine in input all'elaborazione.

- *status* di tipo int
  Contiene lo stato attuale dell'elaborazione e può assumere i valori 0 (inizializzazione), 1 (image processing), 2 (panels detection), 3 (completing) e 4 (completed).

- *stepsDir* di tipo String
  Contiene il percorso della directory dove vengono salvati i file utili in corso di elaborazione (es. le immagini generate durante la image processing) e che potrebbero servire all'utente o al programmatore per controllare il corretto funzionamento della procedura.

- *executionTime* di tipo Time
  Valore del tempo trascorso. Non viene aggiornato costantemente, ma solo in alcuni momenti critici (ad esempio al cambio di stato dell'elaborazione).

- *panelsDetected* di tipo List of *Pannello*
  Lista dei pannelli riconosciuti durante l'elaborazione di un'immagine. Inizializzata a None, si aggiorna durante l'elaborazione.

6.1.3.2 Metodi

- *convertToJPG()*
  Questo metodo converte *sourceImage* in formato JPG. L' image processing è più difficile e pesante per alcuni formati, quindi si è deciso di utilizzare sempre un JPG nelle elaborazioni.

- *processImage(parametri): contours*
  Questa funzione raccoglie tutto il procedimento di processing dall'immagine JPG in input fino al riconoscimento dei contorni in essa, ovvero ciò che restituisce. I contorni forniscono la base su cui poter attuare il riconoscimento geometrico dei pannelli, ovvero rettangoli di determinate dimensioni e rapporti.

- *ordinaVertici(vertici)*
  Sorting dell'array di vertici, valore temporaneo in *Elaborazione*, passato come parametro.

- *RectChecking(newPannello): boolean*
  Controllo sui poligoni rettangolari riconosciuti all'interno di un'immagine, quindi i possibili pannelli solari. Ogni volta che viene riconosciuto un nuovo rettangolo candidato ad essere un nuovo pannello (*newPanello*), si controlla che non sia già stato inserito o che non coincida con un altro rettangolo (dalla *panelsDetected*). Nel caso in cui si trovi una coincidenza, si tiene il rettangolo più piccolo, quindi quello più preciso (poiché la procedura è governata da limiti di grandezza e di rapporti fisici inseriti dall'utente o di default). Ritorna una variabile booleana che indica l'eventuale aggiornamento della lista.

- *cropping(Pannello)*
  Funzione di ritaglio di *Pannello* in *sourceImage* e salvataggio in *saveDir*.

- *scriviCSV()*
  Generazione del file CSV contenente i dati importanti dei pannelli identificati.

- *disegna()*
  Traccia sull'immagine sorgente i contorni che hanno superato con successo tutti gli step e controlli della procedura, ossia i pannelli solari (*panelsDetected*).

- *avviaProcedura(parametri)*
  Metodo principale di *Elaborazione*. Contiene tutte le funzioni sopra descritte e le richiama nell'ordine definito dall'algoritmo ideato per il riconoscimento dei pannelli solari nell'immagine in input, considerando i *parametri*.

Per il momento sono stati identificati i metodi appena elencati, ma il software potrebbe necessitarne altri per migliorare l'elaborazione. Di seguito alcune funzioni ideate per migliorare la procedura di riconoscimento, non ancora considerate all'interno dell'algoritmo e nella Figura 6.

- *gestioneMetadata()*
  Probabilmente servirà una funzione per gestire i metadati immagazzinati nell'immagine in input: tra questi sono di fondamentale importanza le coordinate geografiche.
  Come già detto non è ancora nota la forma di questi metadati e quindi come gestirli, di conseguenza il metodo è ancora vago.

- *limits()*
  Il confronto dei poligoni con i parametri limite potrebbe essere effettuata da un metodo dedicato da richiamare durante la procedura.

- *Machinelearning()*

Per migliorare l'efficienza dell'algoritmo di riconoscimento dei pannelli, potrebbe essere usato qualche tecnica di Machine Learning, oltre all'uso di limitazioni fisse e rigide: la Computer Vision non è perfetta, quindi un buon algoritmo di Machine Learning potrebbe essere d'aiuto.

6.1.4 Pannello

Classe per la definizione dei pannelli. Gli oggetti istanziati da *Pannello* sono i pannelli identificati durante l'elaborazione dell'immagine.

6.1.4.1 Attributi

- *id* di tipo int
  Identificativo univoco.

- *vertici* di tipo List of tuple
  Lista di dimensione 4, contenente le coordinate immagine dei vertici del pannello nell'immagine in cui è stato individuato.

- *Centroide* di tipo tuple
  Coordinate del centro del pannello riferite all'immagine in cui questo è stato individuato.

- *area* di tipo int
  Valore dell'area del pannello in pixel.

- *angolo* di tipo int

Quantità dell'angolazione del pannello rispetto al sistema di coordinate dell'immagine in cui è stato individuato.

- *minBox* di tipo Rect

Rettangolo minimo che individua il pannello.

- *externalRect* di tipo Rect

Rettangolo i cui lati sono paralleli agli assi x,y del sistema cartesiano dell'immagine, quindi circoscritto al pannello, che può essere inclinato.

- *cropImgPath* di tipo string

Attributo che tiene traccia del percorso dell'immagine in cui è stato individuato ed estratto il pannello.

6.2 Object Diagram

Il diagramma degli oggetti descrive una situazione tipica in cui può trovarsi il sistema.
Nella Figura 8 in particolare è evidente che:

1. SPA ha un set di 4 immagini in input.
2. 2 elaborazioni sono state completate (status 4), 1 elaborazione è in corso (status 2), mentre l'ultima elaborazione non è stata ancora inizializzata (si attende la fine di quella precedente).
3. El1 ha identificato 3 pannelli in circa 31 secondi, El2 ne ha identificati 2 in circa 30 secondi.
4. El3 è nella fase di "panel detection" e per il momento ha identificato un pannello dopo circa 7 secondi di elaborazione.
5. L'oggetto corrispondente alla quarta immagine è stato inizializzato e salvato nella lista input di SPA, per procedere all'elaborazione si attende il completamento di quella precedente.

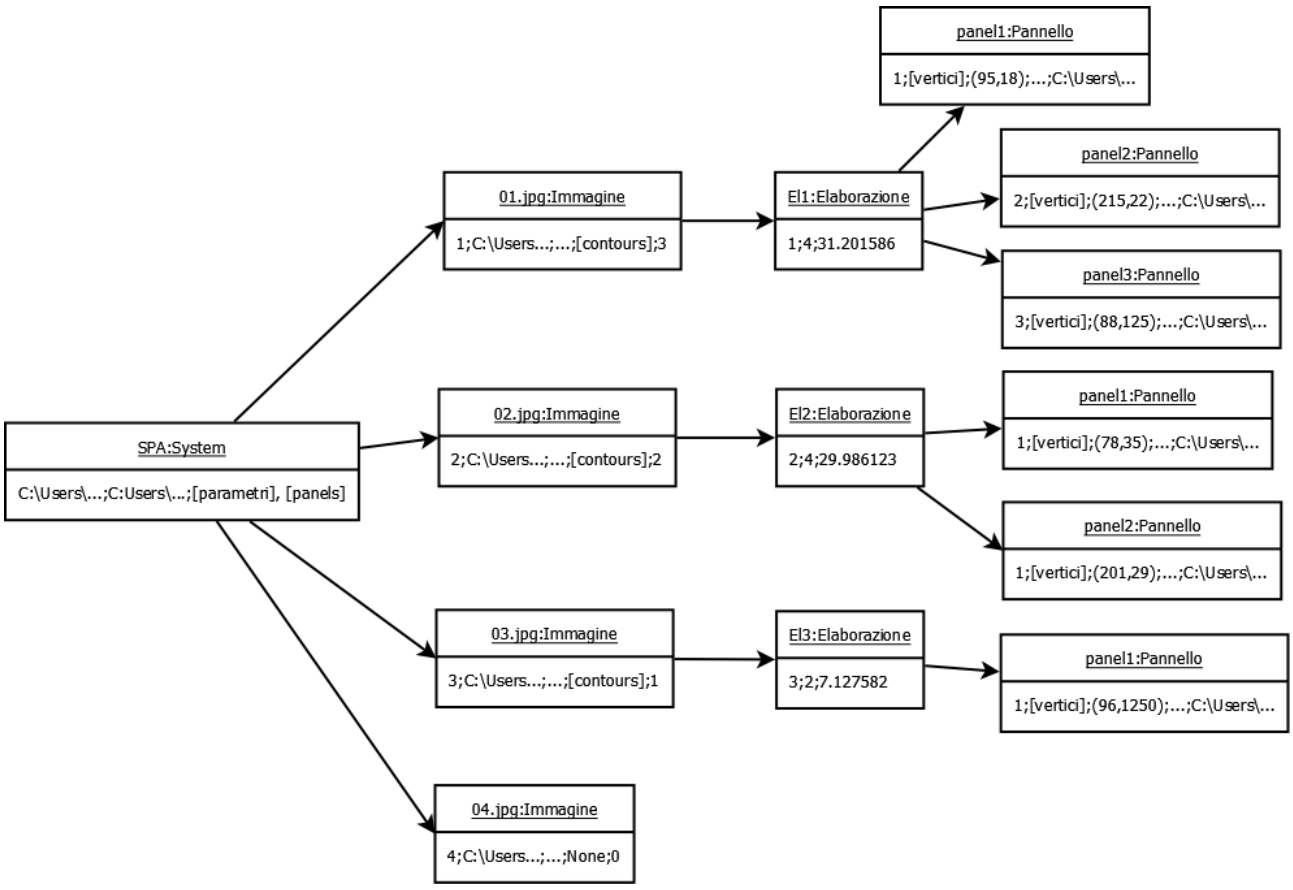

Figura 8. Object Diagram

7 Dynamic model

In questa sezione vengono descritti i comportamenti dinamici del sistema, facendo riferimento ai requisiti dell'URD.

7.1 Funzionalità di SPA: macchina a stati

Il grafico seguente rappresenta la macchina a stati di SPA, ovvero come il sistema risponde alla pressione dei pulsanti della propria interfaccia.

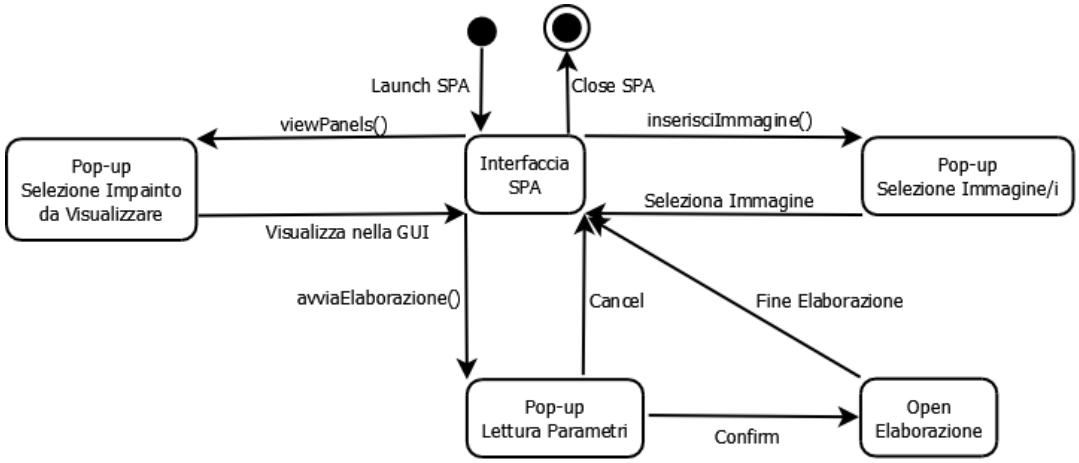

Figura 9. Diagramma di stato

Le funzionalità di SPA sono 3:

1) Selezione Input, ovvero la scelta dell'immagine o dell'impianto (set di immagini) da elaborare. Alla pressione del pulsante corrispondente si apre un pop-up del SO con l'albero delle cartelle del pc, dal quale l'utente può selezionare l'immagine/i.

2) Avvia Elaborazione.
   Il tasto di avvio elaborazione richiama in primis la funzione di lettura dei parametri inseriti dall'utente e apre un pop-up di conferma; nel caso in cui l'utente decidesse di continuare con il setup descritto nella finestra, si passa alla procedura di riconoscimento.

3) Visualizza Pannelli (priorità "Desirable" – vedi URD)
   L'interazione con l'utente è analoga a quella del punto 1 – input, con la differenza che l'utente può selezionare una cartella che rappresenta l'output di un impianto processato: SPA dovrebbe essere in grado di leggerne il contenuto e rendere visibili le immagini dei pannelli e delle schede (estraendo i dati dai file descrittivi) nella GUI.

7.2 Inserimento input e avvio elaborazione (Requirements ID 1 e 2)

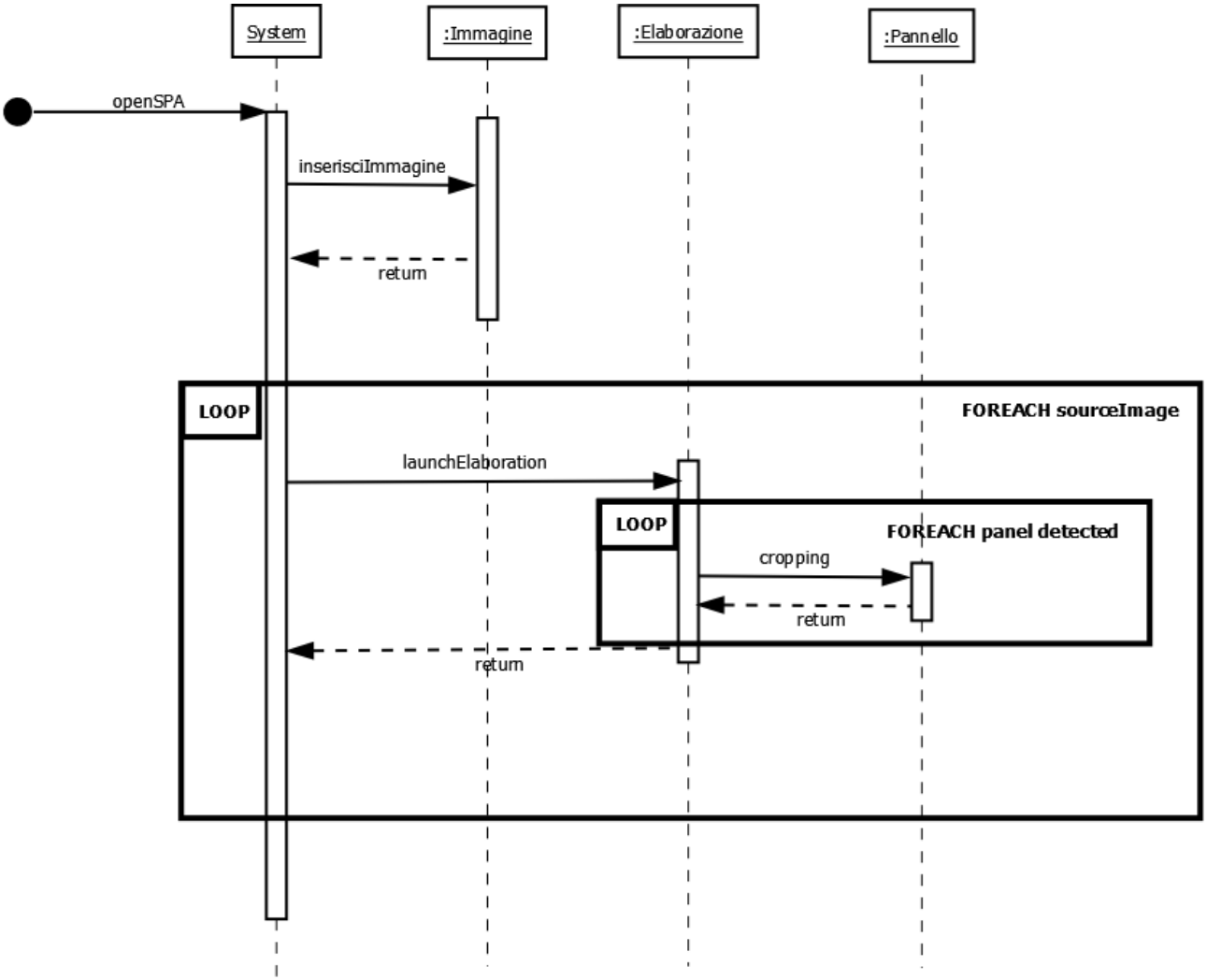

Figura 10. Diagramma di sequenza del flusso

Il diagramma raffigurato descrive il comportamento del sistema e dei vari oggetti in gioco nell'algoritmo. Innanzi tutto l'utente deve scegliere l'input (una o più immagini) e solo successivamente può partire l'elaborazione: una per ogni immagine in input. Ogni volta che durante la procedura viene riconosciuto un pannello, questo viene croppato e salvato.

7.3 Flusso dell'algoritmo (Requirements ID 4 e 6)

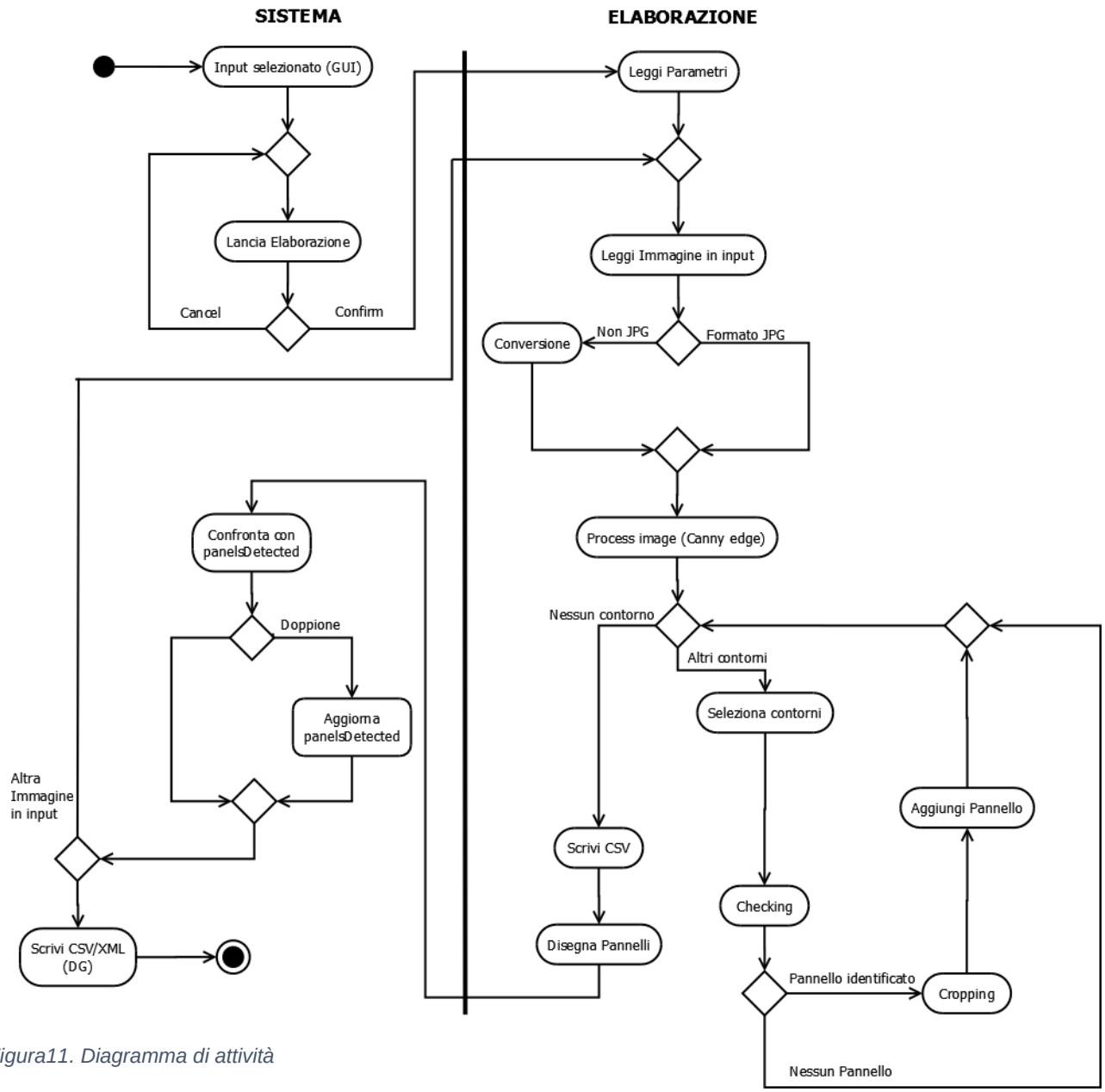

Figura11. Diagramma di attività

Il diagramma in Figura 11 descrive l'algoritmo di riconoscimento dei pannelli solari.
È evidenziato il ruolo del sistema e dell'elaborazione. Nel grafico è inoltre possibile apprendere come SPA decide di processare un'immagine, la sequenza delle azioni che compie e i momenti in cui procedere con l'algoritmo, uscire da un loop (attraverso i blocchi decisionali) e la produzione dei dati.