Image Processing System - DEVS Model

Author: Saja Fawagreh

Student Number: 101217326

Course Name: SYSC 5104 - Methodologies for Discrete Event Modelling and Simulation

Instructor: Dr. Gabriel Wainer

Date Submitted: 2025-03-03

1. Introduction

This report presents the design and implementation of an Image Processing System using the Discrete Event System Specification (DEVS) framework. The system consists of three main components: **Loader**, **Filter**, and **Analyzer**, which work together to process a sequence of images.

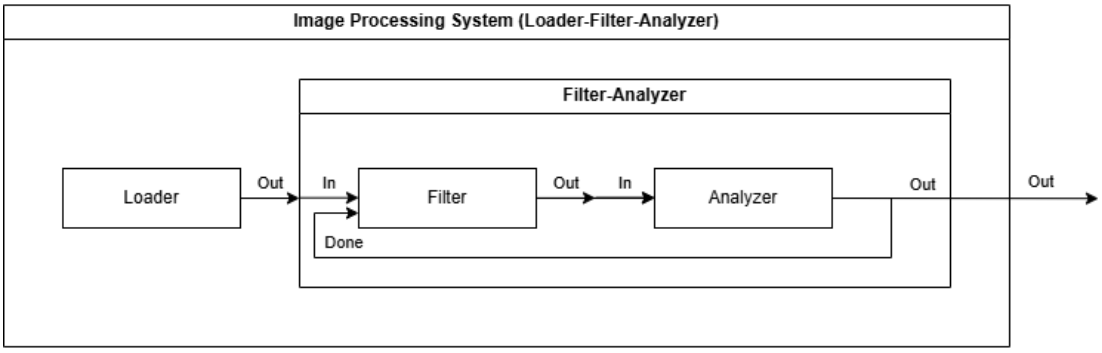
2. Conceptual Description of the Image Processing System

This section provides an overview of the system's purpose, components, and high-level functionality.

2.1 Problem Description

Machine vision is essential in fields like medical imaging, security surveillance, and industrial automation, where accurate image analysis supports decision-making. In medicine, doctors use processed MRI and CT scans to detect anomalies, while in security, enhanced surveillance helps identify threats. In manufacturing, automated defect detection ensures quality control. However, raw images often contain noise and distortion, which can obscure important details. Effective image processing improves accuracy, speed, and reliability in these applications. [1]

2.2 Model Structure



2.3 Component Descriptions

Loader

- Function: Accepts a list of images and sends it to the **Filter**.
- Behavior:
    1. Loads a list of .jpeg images at initialization.
    2. Converts each image to .png format before forwarding.
    3. Forwards one image at a time to the **Filter** every 5 seconds for processing.

Filter

- Function: Stores and processes images before sending them to the **Analyzer**.
- Behavior:
    1. Maintains a FIFO queue for images received from the **Loader**.
    2. Filters the first image in the queue (e.g., noise removal, sharpening).
    3. Sends the filtered image to the **Analyzer**.
    4. Waits for a completion signal from the **Analyzer** before sending the next image.

Analyzer

- Function: Analyzes the filtered image and signals the **Filter** when finished.
- Behavior:
    1. Accepts the image from the **Filter**.
    2. Performs analysis (e.g., detecting tumours, identifying defects).
    3. Generates an analysis report and sends the report as output.
    4. Sends a "done" signal to the **Filter** to enable processing of the next image.

3. Model Structure and Coupling Scheme

This section defines the structure of the atomic and coupled models in the Image Processing System, following the DEVS modelling framework. The system consists of three atomic models and two coupled model, which organizes their interactions.

3.1 Atomic Models

The system includes the following atomic models, each representing an independent component with defined behaviour:

3.1.1 Loader

- Description:

  The Loader initializes with a list of images in .jpeg format, converts each to .png, and sends them one at a time to the Filter every 5 seconds. Once all images are sent, it becomes inactive.

- Ports:

  image_out: Sends images (image1.png, image2.png, etc.) to the Filter.

3.1.2 Filter

- Description:

  The Filter receives images from the Loader, applies processing (e.g., enhancement, noise reduction), and sends the filtered images to the Analyzer. It waits for a "done" signal from the Analyzer before processing the next image.

- Ports:
    - **image_in**: Receives images (image1.png, image2.png, etc.) from the Loader.
    - **filtered_image_out**: Sends filtered images (filtered_image1.png, filtered_image2.png, etc.) to the Analyzer.
    - **done**: Receives "done" from the Analyzer to start the next image.

3.1.3 Analyzer

- Description:

  The Analyzer receives filtered images, performs analysis (e.g., detecting tumours, identifying defects), and generates a report. Once the image is processed, it sends a "done" signal to the Filter to allow the next image to be processed.

- Ports:
    - **filtered_image_in**: Receives filtered images (filtered_image1.png, filtered_image2.png, etc.) from the Filter.
    - **analysis_report_out**: Sends analysis reports (image1_analysis_report.txt, image2_analysis_report.txt, etc.).
    - **done**: Sends "done" to the Filter.

3.2 Coupled Model

The system includes the following coupled models, integrating different atomic models:

3.2.1 FilterAnalyzer

- Description:

  The FilterAnalyzer manages the interaction between the Filter and the Analyzer to ensure proper sequential image processing.

- Ports:
    - **in**: Receives images from the Loader (image1.png, image2.png, etc.).
    - **out**: Sends analysis reports (image1_analysis_report.txt, image2_analysis_report.txt, etc.).
- Coupling:
    - External Couplings:
        - Receives images from the Loader.
        - Sends analysis reports as output.

    - Internal Couplings:
        - Filtered images are sent to the Analyzer.
        - The Analyzer signals the Filter to process the next image.

3.2.2 Top System

- Description:

  The Top System integrates the Loader and the FilterAnalyzer, ensuring images are processed sequentially from loading to filtering to analyzing.

- Ports:
    - **report_out**: Outputs the final analysis reports (image1_analysis_report.txt, image2_analysis_report.txt, etc.).

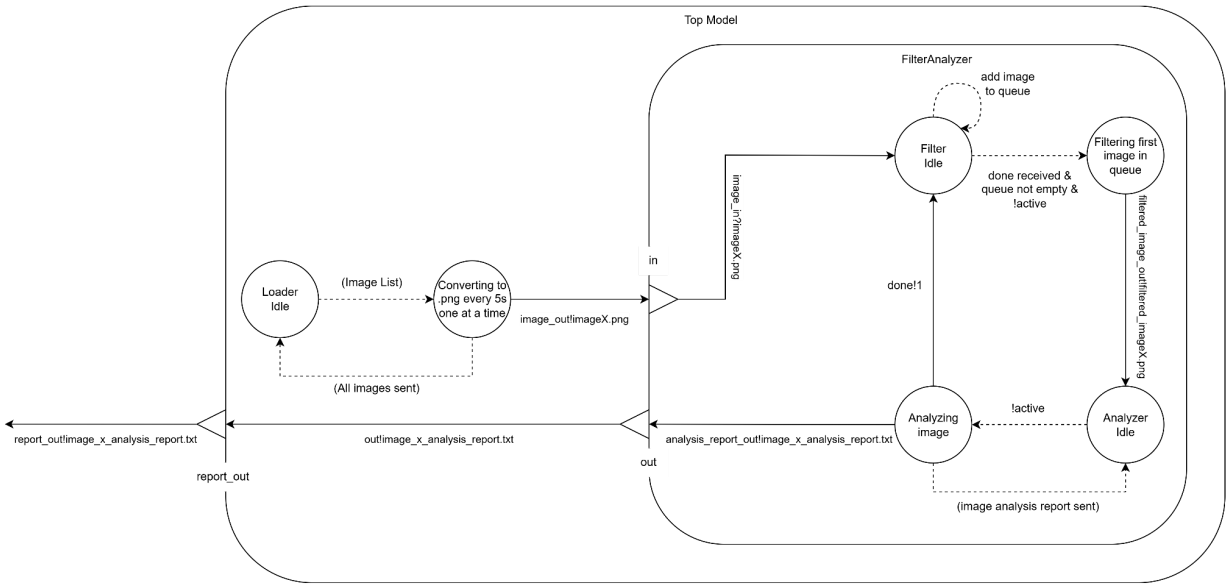
- Coupling:
    - External Couplings:
        - The Top System outputs the final analysis reports.
    - Internal Couplings:
        - The Loader sends images to the FilterAnalyzer System.
        - The FilterAnalyzer System outputs final reports.

4. DEVS Formal Specification of the Coupled Model

This section presents the DEVS formal specification for each **coupled model** in the Image Processing System, outlining their structure, input and output interactions, and internal & external couplings that define the system's overall behaviour.

4.1 Top System

The Top System is a coupled model that integrates the Loader and the FilterAnalyzer. It initializes with a list of images, converts them to .png, and sequentially sends them to the FilterAnalyzer for processing. The FilterAnalyzer processes and analyzes the images, generating reports that are sent as the final output of the system.

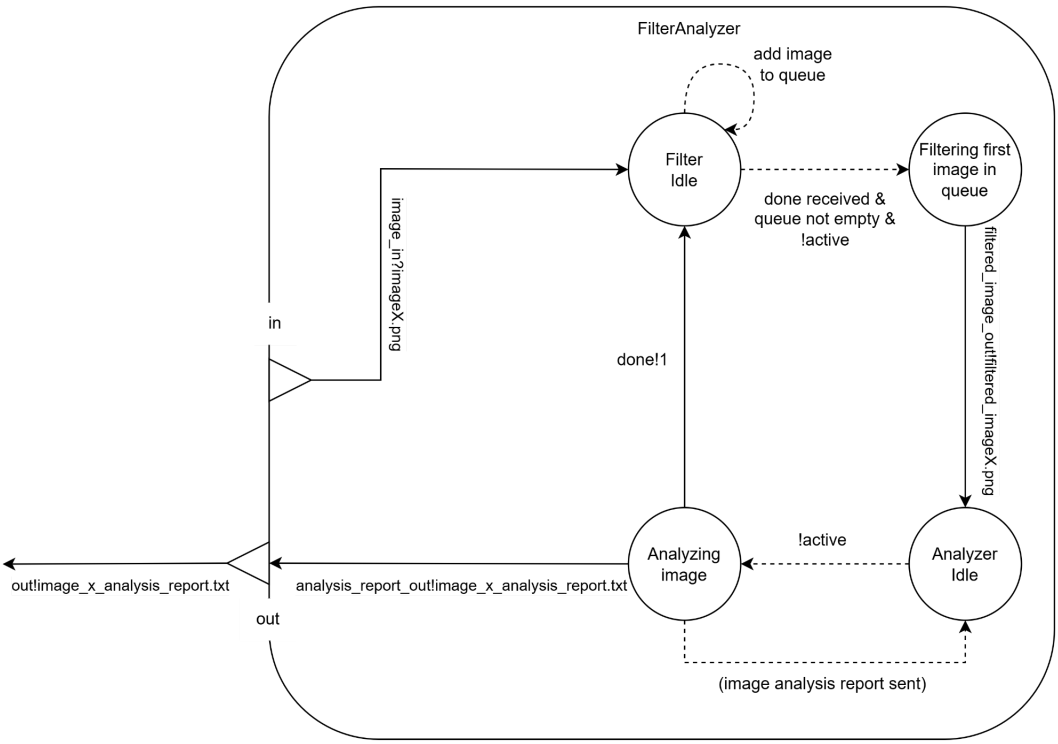


$$TopSystem = \langle X,\ Y,\ M,\ EIC,\ EOC,\ IC,\ Select \rangle$$

- X (Input set): $\emptyset$ $(Top\ system\ does\ not\ receive\ external\ input)$
- Y (Output set): $\{\ report\_out\ \}$ $(Final\ analysis\ reports\ generated)$
- M (Components): $\{\ Loader,\ FilterAnalyzer\ \}$
- EIC (External Input Coupling): $\emptyset$ $(No\ external\ input\ couplings)$
- EOC (External Output Coupling):
  - $\{\ FilterAnalyzer.out \rightarrow report\_out\ \}$
  - The FilterAnalyzer generates a report, which becomes the system's output.
- IC (Internal Coupling):
  - $\{\ Loader.image\_out \rightarrow FilterAnalyzer.in\ \}$
  - The Loader sends .png images to the FilterAnalyzer.
- Select: Not required since no simultaneous events need resolution.

4.2 FilterAnalyzer

The FilterAnalyzer is a coupled model that integrates the Filter and Analyzer components. It receives .png images, processes them in the Filter, and sends the filtered images to the Analyzer for analysis. The Analyzer generates a report and sends a done signal back to the Filter, allowing it to process the next image in the queue.



$$FilterAnalyzer = \langle X,\ Y,\ M,\ EIC,\ EOC,\ IC,\ Select \rangle$$

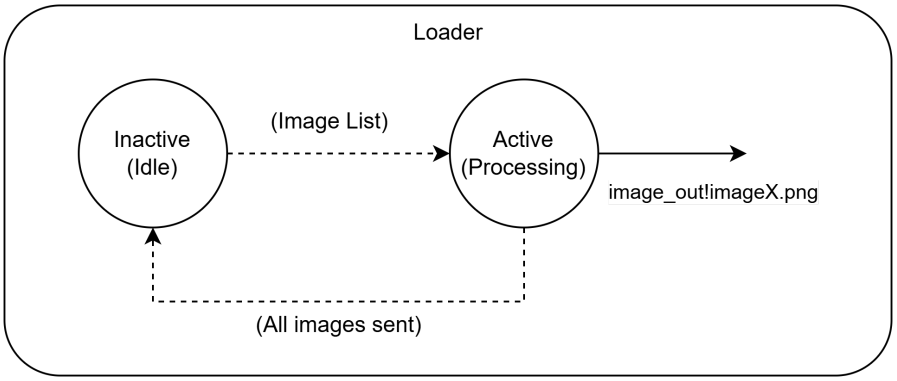
- X (Input set): $\{\ in\ \}$ $(Receives\ .png\ images)$
- Y (Output set): $\{\ out\ \}$ $(Outputs\ analysis\ reports)$
- M (Components): $\{\ Filter,\ Analyzer\ \}$
- EIC (External Input Coupling):
  - $\{\ in \rightarrow Filter.image\_in\ \}$
  - The FilterAnalyzer receives .png images and forwards them to the Filter for processing.
- EOC (External Output Coupling):
  - $\{\ Analyzer.analysis\_report\_out \rightarrow out\ \}$
  - The Analyzer generates a report, which becomes the system's output.
- IC (Internal Coupling):
  - $\{\ Filter.filtered\_image\_out, Analyzer.filtered\_image\_in\ \}$
  - $\{\ Analyzer.done, Filter.done\ \}$
  - The Filter sends the processed image to the Analyzer for analysis.
  - Once the Analyzer completes processing, it sends a "done" signal back to the Filter, indicating readiness for the next image.
- Select: Not required since no simultaneous events need resolution

5. DEVS Formal Specification of the Atomic Models

This section presents the DEVS formal specification for each atomic model in the Image Processing System, describing their states, input and output events, and transition behaviours.

5.1 Loader

The Loader initializes with a list of images, converts them to .png, and sequentially sends one image every 5 seconds to the Filter. Once all images are sent, it transitions to an Inactive state.



- S (State Set):

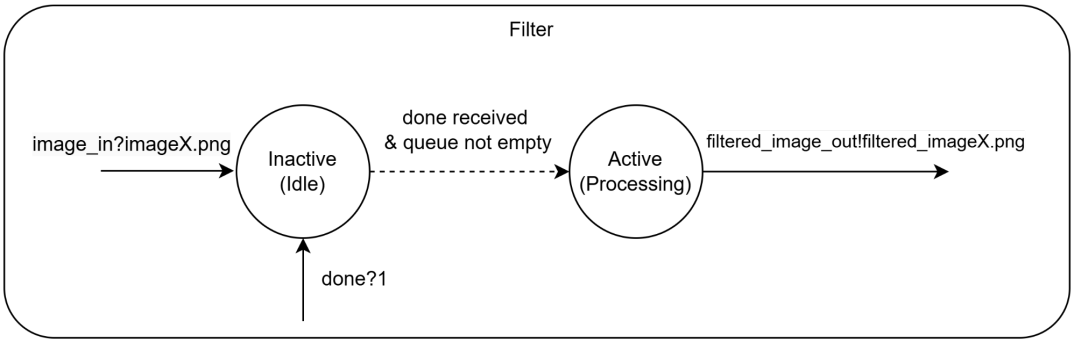
  $S = \{active, image\_queue, sigma\}$
- X (Input Set):

  $X = \emptyset$ $(Loader\ does\ not\ receive\ external\ input)$
- Y (Output Set):

  $Y = \{image\_out\}$ $(Sends\ images\ as\ output)$
- δ_{int} (Internal Transition Function):
  - If Active and images remain → continue sending the next image.
  - If all images are sent → transition to Inactive.
- δ_{ext} (External Transition Function):
  - Not applicable (Loader does not process external events).
- λ (Output Function)
  - When Active, converts and outputs an image (e.g., image1.png).
- ta (Time Advance Function):
  - ta(Active) = 5.0s (Processes images every 5 seconds).
  - ta(Inactive) = ∞ (Waits indefinitely when no images are available).

5.2 Filter

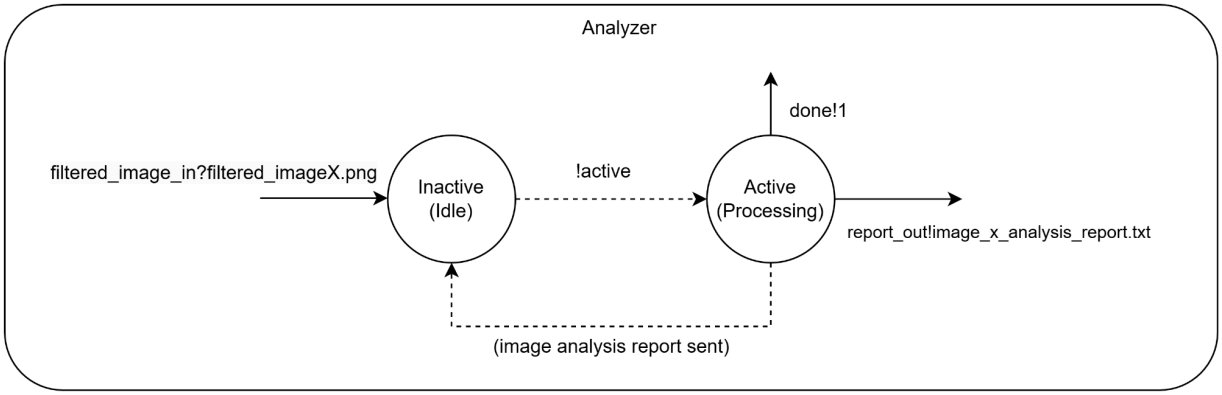
The Filter receives images from the Loader, processes them, and sends them to the Analyzer. It stores incoming images in a queue and waits for a "done" signal from the Analyzer before processing the next image. Once all images are processed, it transitions to an Inactive state.



- S (State Set):
  $S = \{active, image\_queue, done\_received, sigma\}$
- X (Input Set):
  $X = \{image\_in, done\}$
- Y (Output Set):
  $Y = \{filtered\_image\_out\}$ $(Sends\ filtered\ images\ as\ output)$
- δ_{int} (Internal Transition Function):
  - Removes the processed image from the queue.
  - Deactivates and waits for new images or done signal.
- δ_{ext} (External Transition Function):
  - Stores incoming images in the queue.
  - If a "done" signal is received, marks the filter as ready for the next image.
  - If the queue is not empty, "done" is received, and the filter is not active, it activates immediately to process the next image.
- λ (Output Function)
  - When Active, it filters and outputs the image (e.g., filtered_image1.png).
- ta (Time Advance Function):
  - ta(Active) = 0.1s (Schedules processing after 0.1s.).
  - ta(Inactive) = ∞ (Waits indefinitely when no images are waiting or done signal is not received).

5.2 Analyzer

The Analyzer receives filtered images, processes them, and generates an analysis report. Upon receiving an image, if it is not active, it becomes active, processes the image, and outputs a report. Once processing is complete, it sends a "done" signal to indicate readiness for the next image. It then transitions to an inactive state, waiting indefinitely for the next image.



- S (State Set):

  $S = \{active, image, sigma\}$
- X (Input Set):

  $X = \{filtered\_image\_in\}$
- Y (Output Set):

  $Y = \{report\_out,\ done\}$ $(Outputs\ the\ image\ analysis\ report\ and\ a\ "done"\ signal)$
- δ_{int} (Internal Transition Function):
  - Deactivates, clears the image, and waits for a new image.
- δ_{ext} (External Transition Function):
  - If an image is received while inactive, it stores the image and immediately activates to process it.
- λ (Output Function)
  - When active, it analyzes the image and outputs the report (e.g., imageX_analysis_report.txt).
- ta (Time Advance Function):
  - ta(Active) = 0.1s (Schedules processing after 0.1s.).
  - ta(Inactive) = ∞ (Waits indefinitely for the next image).

6. Model Experimentation and Validation

This section outlines the experimentation strategy, which involves testing each atomic model separately before incrementally testing the coupled system. The objective is to validate the correctness of the Loader, Filter, and Analyzer by designing two test cases per model and analyzing their behaviour under different scenarios.

6.1 Loader Experimentation

Objective:

To verify that the Loader correctly converts .jpeg images to .png and sends them sequentially every 5 seconds until all images are processed.

Test 1: Basic Image Loading Test

- Input: A list of two images:
  ["image1.jpeg", "image2.jpeg"]
- Expected Behavior:
  - State: Active when the list of images is received.
  - Converts image1.jpeg → image1.png and sends it at t = 5s
  - Converts image2.jpeg → image2.png and sends it at t = 10s
  - State: Deactivated after sending all images.

Test 2: Empty List Test

- Input: An empty list of images: []
- Expected Behavior:
  - The Loader should remain inactive (state = 0).
  - No images should be sent.

6.2 Filter Experimentation

Objective:

To verify that the Filter correctly processes images and waits for the "done" signal from the Analyzer before processing the next image.

Test 1: Basic Image Filtering Test

- Input:
    - The Loader sends two images (image1.png, image2.png) to the Filter at:
        - t = 5s → image1.png
        - t = 10s → image2.png
    - The Analyzer processes each filtered image and sends a "done" signal back to the Filter at:
        - t = 6s → "done" signal for image1.png
        - t = 11s → "done" signal for image2.png
- Expected Behavior:
    - Image 1 Processing:
        - State: Active (Filter starts processing image1.png at t = 5).
        - image1.png is received.
        - Filter converts it to filtered_image1.png and sends it at t = 5.1s.
        - State: Deactivated (Filter stops processing at t = 5.1s).
        - Analyzer sends "done" at t = 6s.
    - Image 2 Processing:
        - State: Active (Filter starts processing image2.png at t = 10).
        - image2.png is received.
        - Filter converts it to filtered_image2.png and sends it at t = 10.1s.
        - State: Deactivated (Filter stops processing at t = 10.1s).
        - Analyzer sends "done" at t = 11s.

Test 2: No "Done" Signal

- Input:
    - The Loader sends two images (image1.png, image2.png) to the Filter at:
        - t = 5s → image1.png
        - t = 10s → image2.png
    - The Analyzer never sends a "done" signal.
- Expected Behavior:
    - Image 1 Processing:
        - State: Active (Filter starts processing image1.png at t = 5).
        - Filter converts it to filtered_image1.png and sends it at t = 5.1s.
        - State: Deactivated (Filter stops processing at t = 5.1s).

    - Image 2 Processing:
        - image2.png remains in the queue, waiting indefinitely.

6.3 Analyzer Experimentation

Objective:

To verify that the Analyzer correctly processes filtered images and generates a corresponding report.

Test 1: Basic Image Analyzing Test

- Input:
    - The Filter sends two filtered images (filtered_image1.png, filtered_image2.png) to the Analyzer at:
        - t = 5s → filtered_image1.png
        - t = 10s → filtered_image2.png
- Expected Behavior:
    - Filtered Image 1 Processing:
        - Analyzer State: Active (Analyzer starts processing filtered_image1.png at t = 5s).
        - Generates image1_analysis_report.txt and sends "done" at t = 5.1s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 5.1s).
    - Filtered Image 2 Processing:
        - Analyzer State: Active (Analyzer starts processing filtered_image2.png at t = 10s).
        - Generates image2_analysis_report.txt and sends "done" at t = 10.1s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 10.1s).

Test 2: Simultaneous Filtered Image Inputs to Analyzer

- Input:
    - The Filter sends two filtered images (filtered_image1.png, filtered_image2.png) to the Analyzer at:
        - t = 5s → filtered_image1.png
        - t = 5s → filtered_image2.png

- Expected Behavior:
    - t = 5s: The Analyzer receives filtered_image1.png first and starts processing.
        - Since the Analyzer has no queue, filtered_image2.png is ignored.
    - t = 5.1s: The Analyzer completes processing filtered_image1.png and sends:
        - Image1_analysis_report.txt
        - "done" signal

6.4 Coupled Model (FilterAnalyzer System) Experimentation

Objective:

To verify that the FilterAnalyzer system works as a unit, ensuring proper handshaking between components.

Test 1: Sequential Image Processing Pipeline

- Input:
    - The Loader sends two images (image1.png, image2.png) to the FilterAnalyzer at:
        - t = 5s → image1.png
        - t = 10s → image2.png
- Expected Behavior:
    - Image 1 Processing:
        - Filter State: Active (Filter starts processing image1.png at t = 5).
        - Filter converts it to filtered_image1.png and sends it at t = 5.1s.
        - Filter State: Deactivated (Filter stops processing at t = 5.1s).
        - Analyzer State: Active (Analyzer starts processing filtered_image1.png at t = 5.1s).
        - Analyzer generates image1_analysis_report.txt and sends "done" at t = 5.2s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 5.2s).
    - Image 2 Processing:
        - Filter State: Active (Filter starts processing image2.png at t = 10).
        - Filter converts it to filtered_image2.png and sends it at t = 10.1s.
        - Filter State: Deactivated (Filter stops processing at t = 10.1s).
        - Analyzer State: Active (Analyzer starts processing filtered_image2.png at t = 10.1s).

        - Analyzer generates image2_analysis_report.txt and sends "done" at t = 10.2s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 10.2s).

Test 2: Simultaneous Image Processing

- Input:
    - The Loader sends two images (image1.png, image2.png) at t = 5s.
        - t = 5s → image1.png, image2.png (sent together).
- Expected Behavior:
    - Image 1 Processing:
        - Filter State: Active (Filter starts processing image1.png at t = 5).
        - Filter converts it to filtered_image1.png and sends it at t = 5.1s.
        - Filter State: Deactivated (Filter stops processing at t = 5.1s).
        - Analyzer State: Active (Analyzer starts processing filtered_image1.png at t = 5.1s).
        - Analyzer generates image1_analysis_report.txt and sends "done" at t = 5.2s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 5.2s).
    - Image 2 Processing:
        - Filter State: Active (Filter starts processing image2.png at t = 5.2s after receiving "done").
        - Filter converts it to filtered_image2.png and sends it at t = 5.3s.
        - Filter State: Deactivated (Filter stops processing at t = 5.3s).
        - Analyzer State: Active (Analyzer starts processing filtered_image2.png at t = 5.3s).
        - Analyzer generates image2_analysis_report.txt and sends "done" at t = 5.4s.
        - Analyzer State: Deactivated (Analyzer stops processing at t = 5.4s).

6.4 Top System Experimentation

Objective:

To verify that the Top System, which integrates the Loader and FilterAnalyzer, functions correctly as a complete image processing pipeline, ensuring images are loaded, filtered, and analyzed sequentially.

Test 1: Standard Processing Test

- Input: A list of two images:
  ["image1.jpeg", "image2.jpeg"]
- Expected Behavior:
  - Image 1 Processing:
    - Loader State: Active (Receives image list at initialization).
    - Loader converts image1.jpeg → image1.png and sends it at t = 5s.
    - Filter State: Active (Receives image1.png at t = 5s).
    - Filter converts image1.png → filtered_image1.png and sends it at t = 5.1s.
    - Filter State: Deactivated (Filter stops processing at t = 5.1s).
    - Analyzer State: Active (Receives filtered_image1.png at t = 5.1s).
    - Analyzer generates image1_analysis_report.txt and sends "done" at t = 5.2s.
    - Analyzer State: Deactivated (Analyzer stops processing at t = 5.2s).
  - Image 2 Processing:
    - Loader converts image2.jpeg → image2.png and sends it at t = 10s.
    - Loader State: Deactivated (Loader stops processing at t = 10s)
    - Filter State: Active (Receives image2.png at t = 10s).
    - Filter converts image2.png → filtered_image2.png and sends it at t = 10.1s.
    - Filter State: Deactivated (Filter stops processing at t = 10.1s).
    - Analyzer State: Active (Receives filtered_image2.png at t = 10.1s).
    - Analyzer generates image2_analysis_report.txt and sends "done" at t = 10.2s.
    - Analyzer State: Deactivated (Analyzer stops processing at t = 10.2s).

Test 2: No Image Test

- Input: An empty list of images: []
- Expected Behavior:

- Loader State: Inactive (No images to process).
- Filter and Analyzer: Never activate, as no images are received.

7. Model Experimentation and Validation

This section presents the execution results of the Image Processing System based on the experimentation strategy outlined in Section 6. Each atomic and coupled model is tested independently, and their behaviour is compared against expected outputs to validate correctness. The logs from the simulation runs are provided, followed by an analysis of whether the observed behaviour aligns with expectations.

7.1 Loader Execution

Test 1: Basic Image Loading Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;loader model;;State = 1
5;1;loader model;image_out;image1.png
5;1;loader model;;State = 1
10;1;loader model;image_out;image2.png
10;1;loader model;;State = 0
10;1;loader model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 0s, the Loader is active (State = 1).
- At t = 5s, it sends image1.png, remains active (State = 1), and schedules the next image.
- At t = 10s, it sends image2.png and deactivates after sending all images (State = 0).

The Loader does not send any further images, confirming the expected behaviour.

Test 2: Empty List Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;loader model;;State = 0
0;1;loader model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

  - At t = 0s, the Loader is inactive (State = 0) since the image list is empty.

  The state remains inactive throughout the simulation, confirming no images are processed or sent.

  This matches the expected behaviour where the Loader should remain idle when given an empty list.

7.2 Filter Execution

Test 1: Basic Image Filtering Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;filter done signal file;;6
0;2;filter input file;;5
0;3;filter model;;State = 0
5;2;filter input file;out;image1.png
5;2;filter input file;;5
5;3;filter model;;State = 1
5.1;3;filter model;out;filtered_image1.png
5.1;3;filter model;;State = 0
6;1;filter done signal file;out;1
6;1;filter done signal file;;5
6;3;filter model;;State = 0
10;2;filter input file;out;image2.png
10;2;filter input file;;inf
10;3;filter model;;State = 1
10.1;3;filter model;out;filtered_image2.png
10.1;3;filter model;;State = 0
11;1;filter done signal file;out;1
11;1;filter done signal file;;inf
11;3;filter model;;State = 0
11;1;filter done signal file;;inf
11;2;filter input file;;inf
11;3;filter model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 5s, the Filter receives image1.png and transitions to an active state (State = 1).
- At t = 5.1s, the Filter processes and outputs filtered_image1.png, then deactivates (State = 0).
- At t = 6s, the Analyzer sends a "done" signal, allowing the Filter to process the next image.
- At t = 10s, the Filter receives image2.png and activates again (State = 1).
- At t = 10.1s, the Filter processes and outputs filtered_image2.png, then deactivates (State = 0).
- At t = 11s, the Analyzer sends another "done" signal, confirming synchronization between the components.

This confirms that the Filter correctly processes images sequentially and waits for the "done" signal before processing the next image, matching the expected behaviour.

Test 2: No "Done" Signal

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;filter done signal file;;inf
0;2;filter input file;;5
0;3;filter model;;State = 0
5;2;filter input file;out;image1.png
5;2;filter input file;;5
5;3;filter model;;State = 1
5.1;3;filter model;out;filtered_image1.png
5.1;3;filter model;;State = 0
10;2;filter input file;out;image2.png
10;2;filter input file;;inf
10;3;filter model;;State = 0
10;1;filter done signal file;;inf
10;2;filter input file;;inf
10;3;filter model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 5s, the Filter receives image1.png and transitions to an active state (State = 1).
- At t = 5.1s, the Filter successfully processes and outputs filtered_image1.png, then deactivates (State = 0).

- At t = 10s, the Filter receives image2.png, but since no "done" signal is received from the Analyzer, it remains inactive (State = 0) and does not process the second image.

This matches the expected behaviour where the Filter waits indefinitely for the "done" signal before proceeding to the next image.

7.3 Analyzer Execution

Test 1: Basic Image Analyzing Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;analyzer file;;5
0;2;analyzer model;;State = 0
5;1;analyzer file;out;filtered_image1.png
5;1;analyzer file;;5
5;2;analyzer model;;State = 1
5.1;2;analyzer model;analysis_report_out;image1_analysis_report.txt
5.1;2;analyzer model;done;1
5.1;2;analyzer model;;State = 0
10;1;analyzer file;out;filtered_image2.png
10;1;analyzer file;;inf
10;2;analyzer model;;State = 1
10.1;2;analyzer model;analysis_report_out;image2_analysis_report.txt
10.1;2;analyzer model;done;1
10.1;2;analyzer model;;State = 0
10.1;1;analyzer file;;inf
10.1;2;analyzer model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 5s, the Analyzer receives filtered_image1.png and transitions to active state (State = 1).
- At t = 5.1s, the Analyzer successfully processes the image and outputs image1_analysis_report.txt, then sends a "done" signal (done = 1) before deactivating (State = 0).
- At t = 10s, the Analyzer receives filtered_image2.png and again transitions to active state (State = 1).

- At t = 10.1s, the Analyzer correctly processes the image and outputs image2_analysis_report.txt, then sends a "done" signal (done = 1) before deactivating.

This confirms that the Analyzer correctly processes filtered images sequentially, generates reports, and sends the "done" signal after completing each image.

Test 2: Simultaneous Filtered Image Inputs to Analyzer

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;analyzer file;;5
0;2;analyzer model;;State = 0
5;1;analyzer file;out;filtered_image1.png
5;1;analyzer file;;0
5;2;analyzer model;;State = 1
5;1;analyzer file;out;filtered_image2.png
5;1;analyzer file;;inf
5;2;analyzer model;;State = 1
5.1;2;analyzer model;analysis_report_out;image1_analysis_report.txt
5.1;2;analyzer model;done;1
5.1;2;analyzer model;;State = 0
5.1;1;analyzer file;;inf
5.1;2;analyzer model;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 5s, the Analyzer receives filtered_image1.png and transitions to active state (State = 1).
- At t = 5s, before processing the first image, it also receives filtered_image2.png, but since the Analyzer does not queue images, filtered_image2.png is ignored.
- At t = 5.1s, the Analyzer successfully processes filtered_image1.png, outputs image1_analysis_report.txt, and sends a "done" signal (done = 1) before deactivating.

This matches the expected behaviour for where the Analyzer processes only the first received image and ignores subsequent images until it is ready to receive a new one.

7.4 FilterAnalyzer Execution

Test 1: Sequential Image Processing Pipeline

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;filterAnalyzer file;;5
0;3;analyzer;;State = 0
0;4;filter;;State = 0
5;1;filterAnalyzer file;out;image1.png
5;1;filterAnalyzer file;;5
5;4;filter;;State = 1
5.1;3;analyzer;;State = 1
5.1;4;filter;out;filtered_image1.png
5.1;4;filter;;State = 0
5.2;3;analyzer;analysis_report_out;image1_analysis_report.txt
5.2;3;analyzer;done;1
5.2;3;analyzer;;State = 0
5.2;4;filter;;State = 0
10;1;filterAnalyzer file;out;image2.png
10;1;filterAnalyzer file;;inf
10;4;filter;;State = 1
10.1;3;analyzer;;State = 1
10.1;4;filter;out;filtered_image2.png
10.1;4;filter;;State = 0
10.2;3;analyzer;analysis_report_out;image2_analysis_report.txt
10.2;3;analyzer;done;1
10.2;3;analyzer;;State = 0
10.2;4;filter;;State = 0
10.2;1;filterAnalyzer file;;inf
10.2;3;analyzer;;State = 0
10.2;4;filter;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

  - At t = 5s, the FilterAnalyzer system receives image1.png.
  - At t = 5s, the Filter activates (State = 1) and begins processing image1.png.
  - At t = 5.1s, the Filter outputs filtered_image1.png, then deactivates (State = 0).
  - At t = 5.1s, the Analyzer activates (State = 1) upon receiving filtered_image1.png.
  - At t = 5.2s, the Analyzer outputs image1_analysis_report.txt, sends a "done" signal (done = 1), and deactivates (State = 0).
  - The same sequence repeats for image2.png at t = 10s, confirming the consistent operation.

This confirms that the FilterAnalyzer correctly processes images in sequence and synchronizes with the "done" signal before proceeding to the next image.

Test 2: Simultaneous Image Processing

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;1;filterAnalyzer file;;5
0;3;analyzer;;State = 0
0;4;filter;;State = 0
5;1;filterAnalyzer file;out;image1.png
5;1;filterAnalyzer file;;0
5;4;filter;;State = 1
5;1;filterAnalyzer file;out;image2.png
5;1;filterAnalyzer file;;inf
5;4;filter;;State = 1
5.1;3;analyzer;;State = 1
5.1;4;filter;out;filtered_image1.png
5.1;4;filter;;State = 0
5.2;3;analyzer;analysis_report_out;image1_analysis_report.txt
5.2;3;analyzer;done;1
5.2;3;analyzer;;State = 0
5.2;4;filter;;State = 1
5.3;3;analyzer;;State = 1
5.3;4;filter;out;filtered_image2.png
5.3;4;filter;;State = 0
5.4;3;analyzer;analysis_report_out;image2_analysis_report.txt
5.4;3;analyzer;done;1
5.4;3;analyzer;;State = 0
5.4;4;filter;;State = 0
5.4;1;filterAnalyzer file;;inf
5.4;3;analyzer;;State = 0
5.4;4;filter;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

  - At t = 5s, the FilterAnalyzer system receives image1.png and image2.png simultaneously.
  - The Filter activates (State = 1) and starts processing image1.png first.
  - At t = 5.1s, the Filter outputs filtered_image1.png and deactivates (State = 0).
  - At t = 5.1s, the Analyzer activates (State = 1) upon receiving filtered_image1.png.
  - At t = 5.2s, the Analyzer outputs image1_analysis_report.txt, sends a "done" signal (done = 1), and deactivates (State = 0).
  - Upon receiving "done", the Filter reactivates (State = 1) and begins processing image2.png.

- The same sequence repeats for image2.png at t = 5.3s, confirming the consistent operation.

This matches the expected behaviour where the Filter handles multiple incoming images but processes them sequentially. The system correctly synchronizes using the "done" signal to prevent simultaneous processing.

7.5 Top System Execution

Test 1: Standard Processing Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;3;analyzer;;State = 0
0;4;filter;;State = 0
0;5;loader;;State = 1
5;4;filter;;State = 1
5;5;loader;image_out;image1.png
5;5;loader;;State = 1
5.1;3;analyzer;;State = 1
5.1;4;filter;out;filtered_image1.png
5.1;4;filter;;State = 0
5.2;3;analyzer;analysis_report_out;image1_analysis_report.txt
5.2;3;analyzer;done;1
5.2;3;analyzer;;State = 0
5.2;4;filter;;State = 0
10;4;filter;;State = 1
10;5;loader;image_out;image2.png
10;5;loader;;State = 0
10.1;3;analyzer;;State = 1
10.1;4;filter;out;filtered_image2.png
10.1;4;filter;;State = 0
10.2;3;analyzer;analysis_report_out;image2_analysis_report.txt
10.2;3;analyzer;done;1
10.2;3;analyzer;;State = 0
10.2;4;filter;;State = 0
10.2;3;analyzer;;State = 0
10.2;4;filter;;State = 0
10.2;5;loader;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 0s, the Loader is active (State = 1).
- t = 5s, the Loader sends image1.png to the FilterAnalyzer system.

- t = 5s, the Filter activates (State = 1) and begins processing image1.png.
- t = 5.1s, the Filter converts image1.png → filtered_image1.png, sends it to the Analyzer, and deactivates (State = 0).
- t = 5.1s, the Analyzer activates (State = 1) upon receiving filtered_image1.png.
- t = 5.2s, the Analyzer generates image1_analysis_report.txt, sends a "done" signal (done = 1), and deactivates (State = 0).
- t = 5.2s, the Filter receives "done", confirming that it can now process the next image.
- The same sequence repeats for image2.png at t = 10s (Loader deactivates (State = 0) after sending the image), confirming the consistent operation.

This confirms that the Top System functions correctly, integrating the Loader and the FilterAnalyzer properly. Images are processed sequentially, and proper synchronization is maintained through the "done" signal.

Test 2: No Images Test

- Simulation Log Output:

```
time;model_id;model_name;port_name;data
0;3;analyzer;;State = 0
0;4;filter;;State = 0
0;5;loader;;State = 0
0;3;analyzer;;State = 0
0;4;filter;;State = 0
0;5;loader;;State = 0
```

- Analysis:

✅ Correct Behavior Observed

- At t = 0s, the Loader starts in an inactive state (State = 0) as there are no images to process.
- The Filter remains inactive (State = 0) since no images are received.
- The Analyzer remains inactive (State = 0) since no filtered images are received.

This matches the expected behaviour, where the system correctly handles an empty input list without initiating any processing or state transitions.

8. Conclusion

The Image Processing System was successfully implemented and tested using the DEVS modelling framework. The Loader, Filter, and Analyzer atomic models were developed and validated individually to ensure they functioned correctly. The FilterAnalyzer coupled model was then tested to verify proper synchronization between components.

We confirmed through a structured experimentation process that each model behaves as expected under different test scenarios. The Loader correctly converted and sequentially sent images, the Filter processed images while waiting for a "done" signal, and the Analyzer generated reports and signalled completion. The FilterAnalyzer system demonstrated correct end-to-end image processing, confirming its ability to appropriately handle sequential and simultaneous image inputs.

These results validate the correctness and efficiency of the Image Processing System, showcasing the effectiveness of the DEVS framework in modelling event-driven systems.

9. References

[1] J. Gubbi, B. P, “Machine Vision and Industrial Automation Accelerating Adoption in Industrial Automation”, Tata Consultancy Services, 2018 Accessed: Feb. 06, 2025. [Online]. Available: https://www.tcs.com/content/dam/global-tcs/en/pdfs/what-we-do/research/reimagining-research/vol-1/machine-vision-and-industrial-automation.pdf