CSE 561: Modeling & Simulation Theory and Applications

SIMULATING TECHNICAL CUSTOMER SUPPORT CENTER

Shreyansh Khandelwal
M.S in Computer Science

Bala Sujith Potineni
M.S in Computer Science

School of Computing and Augmented Intelligence
Arizona State University, Tempe, AZ, USA, 85281

Date

Contents

Team Members Tasks (% effort is preliminary)

List of tasks	Name 1	% Effort	Name 2	% Effort
1. Introduction and background	Shreyansh	50	Sujith	50
2. System description	Shreyansh	50	Sujith	50
3. Modeling	Shreyansh	50	Sujith	50
3.1 Model description	Shreyansh	50	Sujith	50
3.2 Model specification	Shreyansh	50	Sujith	50
3.3 Model implementation	Shreyansh	50	Sujith	50
4. Simulation experiments	Shreyansh	50	Sujith	50
5. Evaluation	Shreyansh	50	Sujith	50

6. Conclusions	Shreyansh	50	Sujith	50
7. References	Shreyansh	50	Sujith	50
Appendices	Shreyansh	50	Sujith	50

ABSTRACT

This project report presents a comprehensive model for analyzing and optimizing a technical support center focused on key performance metrics like issue resolution rates and service levels. Multiple variables are investigated including volume of customer inquiries, complexity of issues, staffing levels and skills, knowledge base content, and historical issues. The goal is developing a robust understanding of support center dynamics during periods of varying workloads. Building on discrete event simulations of the routing system and service level agreements (SLAs), the model quantifies the interconnected impact of critical resourcing variables such as staff levels, training, and knowledge base completeness on service quality metrics. Comparisons are made to real-world support constraints and objectives at industry leaders like Amazon. Key contributions are framework for matching technician expertise to inquiry priority and complexity more precisely, dynamic updating of knowledge base content with resolved issues to improve productivity, and optimization of staffing models to achieve targeted resolution rates. With flexible inputs and evidence-based recommendations, the model provides customized guidance for efficiently aligning technical support operations to client domains and objectives.

1 INTRODUCTION AND BACKGROUND

The objective of this project is to develop a model for a technical support center. The aim of the model is to examine several metrics related to request resolution and procedure, considering different scenarios based on specific criteria such as the quantity of technical requests, complexity of requests, demography of expert workers, and issue history. The objective of our project is to analyze the operational features when faced with a high volume of tasks that have a consistent level of complexity and are distributed numerically over time. This analysis will be conducted as the product knowledge base is updated using previous resolutions as a basis. The findings effectively address key considerations pertaining to human resource management, product complexity, and documentation requirements.

In the world of technology, which is getting more complicated and linked, technical customer support centers are very important for keeping customers happy and making sure that goods and services keep working. These help centers are the first line of defense for answering customer questions, taking them to the next level, troubleshooting problems, setting up problems, and fixing a wide range of technical issues. Simulation modeling can be a useful way to improve the level of customer service and run a technical support center more efficiently.

DEVS (Discrete Event System Specification) simulation is a powerful approach for modeling and analyzing complex systems, particularly those involving discrete events and dynamic interactions. The DEVS formalism allows for the representation of system components, their behaviors, and their interactions in a structured and modular manner. By applying DEVS simulation to a technical customer support center, various aspects of its operations can be studies, optimized, and enhanced. This modeling approach can help in evaluating the center’s efficiency in handling customer’s escalations, and their response times.

Keys aspects to consider when modeling a technical customer support center using DEVS may include:

- **Customer Case Interaction Flow**: A customer case based on it's priority, severity and the product that it was registered against can be routed to an appropriate support center and to a specific team based on crucial the case is for the customer and the company.
- **Support Agent Behavior**: The simulation can represent the behaviors and decision making process of these agents, including their solutions provided to each case that was escalated to them.
- **System Performance Metrics**: Simulations can generate performance metrics like average response times, resolution rates, and customer satisfaction scores, allowing for continuous improvement of support center operations.

DEVS simulation to model a technical customer support center provides a systematic and data-driven approach to understanding, optimizing, and managing the complexities of customer support operations. Such modeling can contribute to better resource utilization, improved customer experiences, and ultimately, more efficient and effective support services for the end-users of technical products and services.

Technical customer service centers play a crucial role in the operational framework of international technology corporations, including NetApp, Oracle, VMware, Amazon, Google, and Microsoft. These prominent multinational corporations in the technology sector cater to a vast customer base around the globe, and their technical customer support facilities play a pivotal role in guaranteeing a smooth and satisfactory customer journey. These centers are carefully crafted and managed to offer a comprehensive range of assistance and support for their wide array of products and services. This concise introduction provides an overview of the appearance and operations of technical customer support centers, highlighting their commitment to providing exceptional customer care.

- **NetApp:** It has a well-structured hub of expertise, focusing on data management solutions. The center is equipped with knowledgeable professionals who offer 24/7 support for NetApp's data storage and cloud services.
- **Oracle:** Its support center is a cornerstone of their global operations. It features an extensive network of support professionals, engineers, and experts who assist customers in optimizing their oracle software and hardware investments.
- **Amazon:** The amazon customer support ecosystem extends far beyond e-commerce. The technical support center is a central hub for customers using Amazon Web Services (AWS). It offers round-the-clock assistance for cloud computing and hosting services. The center's team of experts ensures that business can leverage AWS to scale and innovate their digital infrastructure.

These centers, equipped with highly skilled individuals and advanced infrastructure, have a significant impact on assisting businesses in maximizing the capabilities of advanced technologies and guaranteeing a seamless and problem-free client experience on a worldwide level.

The majority of these organizations operate on a significant scale, making it imperative for them to possess knowledge on effectively delivering prompt, dependable, and effective solutions to their end-users. However, an inquiry emerges regarding the methods employed to effectively handle the various categories

and degrees of support requests that are received. How do organizations strategically allocate and utilize their resources to effectively enhance customer satisfaction? One potential approach to consider is the implementation of a hierarchical technical support framework.

A tiered technical support model is a structured framework that categorizes support personnel into several levels or tiers, predicated on their expertise, tenure, and obligations. Each tier is responsible for addressing a distinct spectrum of support issues, ranging from rudimentary to intricate in nature. The concept revolves around the allocation of appropriate individuals to specific problems, with the option to elevate the matter to a higher level if deemed essential. In this manner, multinational corporations enhance the efficiency, quality, and cost-effectiveness of their support services.

In the multifaceted landscape of multinational corporations, the efficiency and effectiveness of customer support operations are paramount. To meet the diverse and often intricate demands of customers, a tiered support structure is commonly employed. This structure is designed to streamline issue resolution, capitalize on the expertise of support personnel, and ultimately deliver exceptional customer service.

Tier I: Technical Support Engineer

Tier I represents the first line of customer support, often referred to as the Technical Support Engineer (TSE). The professionals within this tier possess fundamental skills, enabling them to comprehend customer issues based on problem descriptions and error codes.

They rely on a centralized knowledge portal, which serves as a repository of historical cases and solutions. By comparing the presented issue with past cases, they can quickly diagnose and address routine problems. If they can provide a solution within the stipulated Service Level Agreement (SLA), which is typically one day, the case is marked as resolved and closed. This approach ensures that straightforward issues are promptly addressed.

Tier II: Escalation Engineers

Tier II, comprised of Escalation Engineers (EE), represents the second echelon of support. These individuals possess a higher degree of expertise and domain knowledge. Their role is to investigate cases that require a more in-depth analysis. They may delve into product documentation, knowledge base articles, and release notes to unearth potential solutions.

When a solution is identified, they either present it directly to the customer through online sessions or document it within the case notes, enhancing the repository of solutions for future reference.
However, in cases where a resolution remains elusive, Tier II personnel initiate the escalation process to Tier III, recognizing that deeper expertise is required.

Tier III: Premium Support Engineers or Development Team

Tier III, the final tier in the support structure, is staffed by Premium Support Engineers or members of the product's development team. These individuals possess the highest level of expertise, having contributed to the creation of the product.

They are equipped with a profound understanding of the product's architecture and underlying code base. This level of knowledge empowers them to explore the product's source code to uncover and rectify defects or bugs, ensuring a highly effective and tailored solution for the customer.

The probability of case resolution within Tier III is nearly 100%, as their ultimate responsibility is to resolve cases comprehensively, regardless of their complexity.

The tiered support structure optimizes the allocation of resources, guarantees that customer inquiries are addressed with the appropriate level of expertise, and enhances overall customer satisfaction. This approach is emblematic of multinational companies' commitment to delivering exceptional customer service while harnessing the full spectrum of technical expertise within their support organizations.

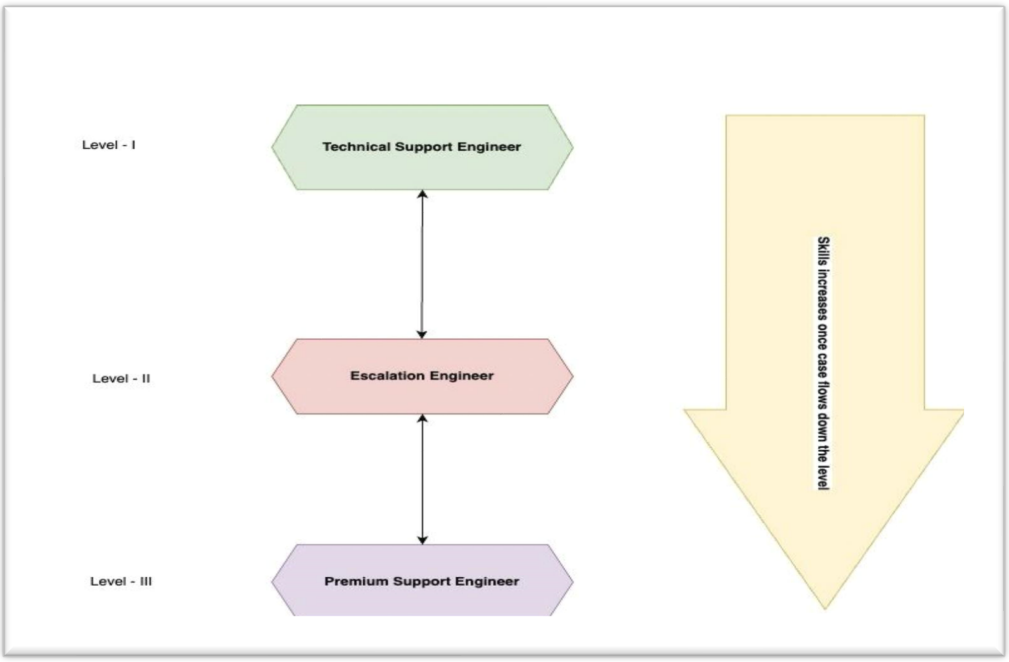


Figure 1: Support Level Tiers

In the figure I, we have defined three major buckets in a technical support center. They are arranged in increasing level of skills. The Premium Support Engineer sits at the end where as the Technical Support Engineer sits at the top or is the first line of defense.

Figure 2: Customer Report Issue in the Product

A customer issue resolution is the steps that need to be taken to successfully solve a customer complaint. It's critical to get issue resolution right, otherwise, customers can become frustrated. If you get it wrong, the customer's will start giving negative feedback on their overall experience and that hampers the organization businesses and goals. A basic idea is that customer reports the issue, a ticket is generated, and severity level/priority level is assigned. A case is assigned to a particular agent belonging to either of those three mentioned categories in figure I.

The technical customer service industry size was approximately $420 billion in 2022 as per Statistica report.

- Salesforce – the leader in customer relationship management(crm) software generated over $26 billion in revenue in 2022, with support services a key element.
- IBM – generates revenue of around $60 billion in technical support & services.
- Oracle - support revenues were around $7.40 billion in 2022.

Figure 3: Global Industry Size

The Figure 4 gives us a clear idea on why simulation is required for support centers.

Figure 4: Simulation necessary for Technical Support

As per stat, on recent volume of customer cases that are registered against Amazon customer center. The following information is good demonstration of how many customers does a tech firm Amazon holds:

- Amazon has over 300 million active customer accounts worldwide.
- An estimated value of receiving 3.8 million customer support contacts per day on average, which is 114 million per month.

Figure 5: Amazon Customer Center

2 SYSTEM DESCRIPTION

a. Introduction to System:

In the world of technology, which is getting more complicated and linked, technical customer support centers are very important for keeping customers happy and making sure that goods and services keep working. These help centers are the first line of defense for answering customer questions, taking them to the next level, troubleshooting problems, setting up problems, and fixing a wide range of technical issues. Simulation modeling can be a useful way to improve the level of customer service and run a technical support center more efficiently.

b. System Components and Architecture:

The system is comprised of a customer case generator that produces cases with varying attribute combinations, including case priority, case severity, the associated product type, error codes, and customer names.

Within the system, there exists a routing mechanism responsible for directing cases to the appropriate support centers. This routing is contingent on the priority and severity of the issues, with high-priority and high-severity cases being directed to the highest skill level or organizational tier within the support framework.

The initial level of support is provided by technical support engineers, representing the foundational tier within the support structure. Their expertise is primarily focused on addressing pre-existing faults or bugs, and their knowledge is limited in scope.

Elevating the support hierarchy, we encounter the Escalation Engineers, a group of support professionals delivering Tier II support. They possess advanced technical acumen and a deeper understanding of the product. When a case is escalated from Level 1, an Escalation Engineer conducts a more comprehensive investigation.

At the apex of the support structure are the Premium Support Engineers, armed with an intricate understanding of the product's architecture and underlying code base. This elevated knowledge empowers them to delve into the product's source code to identify and rectify defects or bugs, ensuring tailored and highly effective solutions for customers.

Each support center comprises these three organizational levels, and cases are directed to them based on the nature of the case.

Furthermore, a transducer and performance analyzer are integrated into the system, capturing crucial metrics that facilitate improvement at each support level. This system component aids in identifying where training is required to enhance the handling of customer cases.

c. System Behavior:

The System behavior is captured in figure III that depicts the process workflow. The case transitions from Level-I to Level-II if the lowest level is not able to resolve a customer case. Similarly, if Level-II is unequipped to resolve a case then it is directed towards the last level that comprises of Development team and Engineers.

d. Abstraction Levels:

The higher-level abstraction is provided in the modeling description section 3.1 and low-level abstraction with DEVS atomic specification is specified in modeling specifications section 3.2.

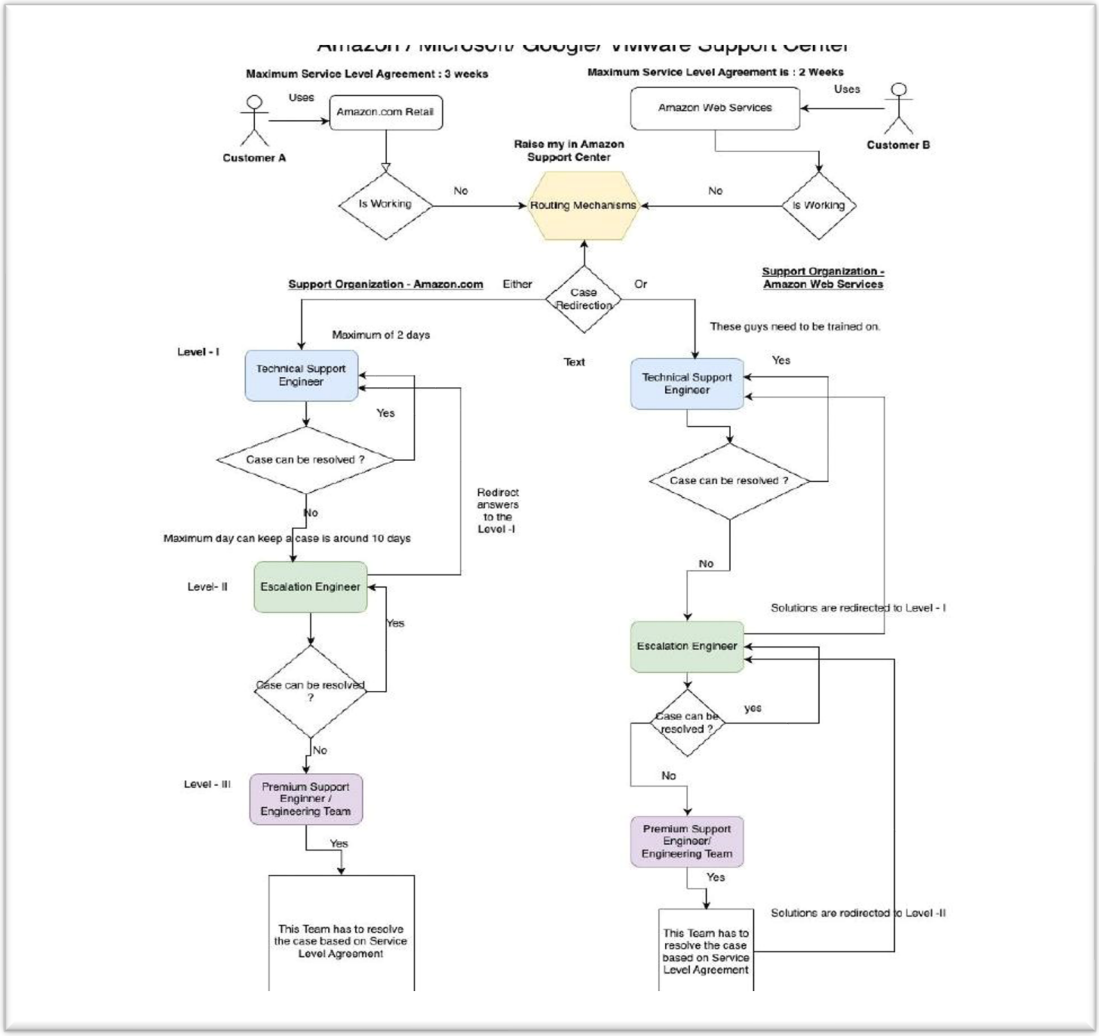


Figure 6: Generic Process Workflow

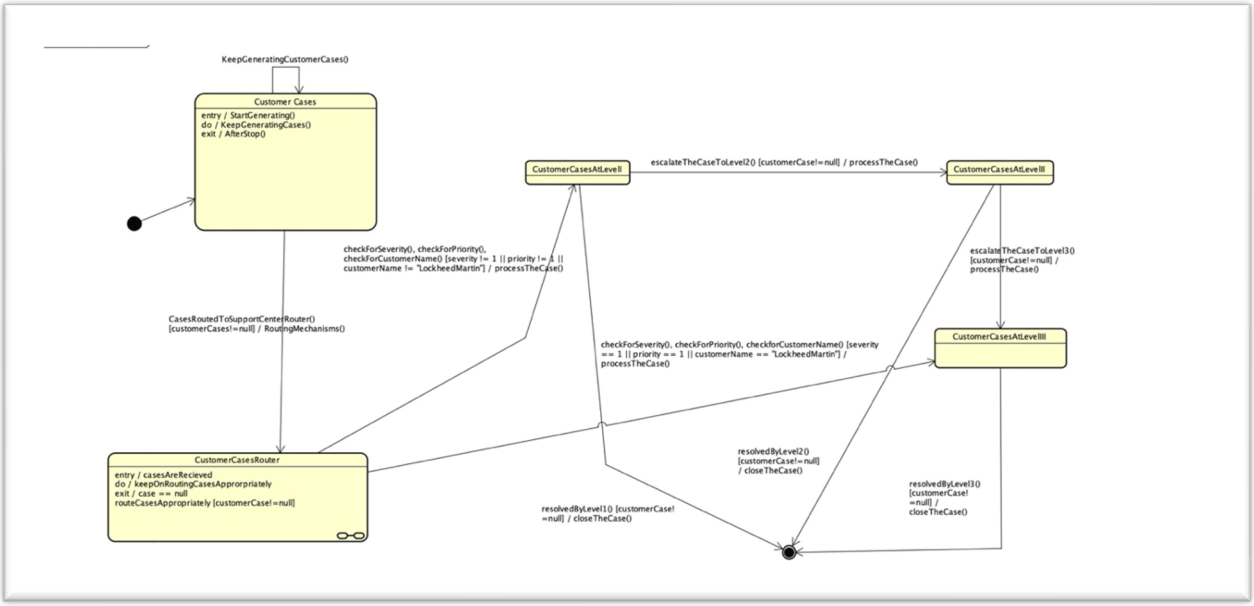


Figure 7: CustomerCase State Machine

In the provided state machine diagram (refer to Figure 7), the lifecycle of customer cases is represented through distinct states:

- **Initial State**
- **CustomerCasesGenerator**
- **CustomerCasesAtRouter**
- **CustomerCasesAtLevel1**
- **CustomerCasesAtLevel2**
- **CustomerCasesAtLevel3**
- **Final State**

1. **CustomerCasesGenerator:** Customer cases originate in the state of CustomerCasesGenerator, where various parameters such as severity, priority, customer name, problem description, error code, and product type contribute to their generation.

2. **CustomerCasesAtRouter:** Following generation, the cases transition to the state of CustomerCasesAtRouter. At this juncture, a pivotal decision is made regarding the subsequent state. The routing mechanisms, governed by criteria such as severity, priority, and customer names, determine whether a case moves to CustomerCasesAtLevel1 or CustomerCasesAtLevel3.

   a. *Routing Mechanism*:
      - *Category 1*: CustomerCases meeting conditions $severity == 1 \,||\, priority == 1 \,||$ customer name is "Lockheed Martin" are directed to CustomerCasesAtLevel3.

- *Category 2*: CustomerCases not meeting Category 1 criteria are directed to CustomerCasesAtLevel1.

3. **CustomerCasesAtLevel1:** In the state of CustomerCasesAtLevel1, cases can follow diverse paths. They may transition to the resolved state, proceed to the final state, or escalate to a higher-level state, namely CustomerCasesAtLevel2.

4. **CustomerCasesAtLevel2:** Cases that escalate to CustomerCasesAtLevel2 present the opportunity for further resolution or escalation to the final state (CustomerCasesAtLevel3).

5. **CustomerCasesAtLevel3:** When cases reach the state of CustomerCasesAtLevel3, they are poised to conclude their journey. Transitioning exclusively to the final state, cases in this state indicate successful resolution.

6. **Final State:** The final state marks the conclusion of the customer case lifecycle, signifying successful resolution. This comprehensive state machine captures the dynamic progression of customer cases through various stages, allowing for efficient tracking and management.

3 MODELING

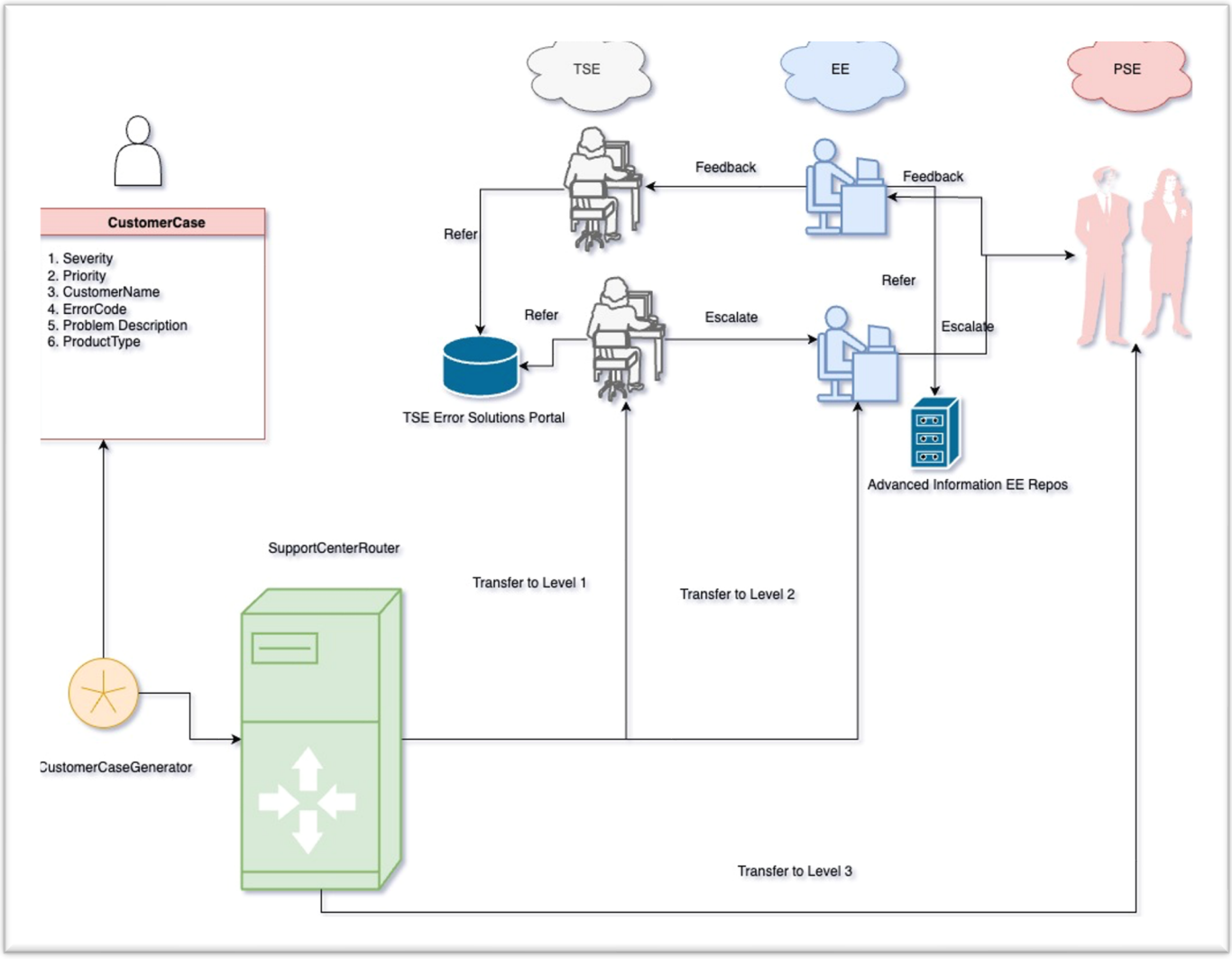


Figure 8: Modeling and Simulation

3.1 Model description

For our modeling and experimentation, our system comprises of the mentioned entities in the figure III. It comprises of the following components:

a. Customer Case Generator:

The customer case generator is responsible for creating customer support cases with randomized attributes to simulate incoming requests. It has a single input port that accepts a {Start} signal along with a number N representing the number of cases to generate.

Upon receiving the {$Start$} input, the generator will create N customer case entities sequentially. Each case is assigned a unique incremental $caseId$ starting from 1. The other attributes are assigned randomly based on the following distributions:

- **Severity**: Uniformly distributed integer from 1 to 5.

```
public int generateRandomSeverity() {
    int randomPriority = ThreadLocalRandom.current().nextInt(1, 6);
    return randomPriority;
}
```

Figure 9: Random Severity

- **Priority**: Uniformly distributed integer from 1 to 5.

```
public int generateRandomPriority() {
    int randomPriority = ThreadLocalRandom.current().nextInt(1, 6);
    return randomPriority;
}
```

Figure 10: Random Priority

- **Customer**: A set of customers using the products and services of Amazon.

```
    ArrayList<String> customerNameList = new ArrayList<>();
    customerNameList.add("Shreyansh");
    customerNameList.add("LockheedMartin");
    customerNameList.add("JohnSmith");
    customerNameList.add("USDeptofAgriculture");
    customerNameList.add("Sujith");
    customerNameList.add("Nalini");
    customerNameList.add("BobWilson");
    customerNameList.add("JaneDoe");
    customerNameList.add("AxelLopez");
    customerNameList.add("Michael");

    Random rand = new Random();

    String randomCustomerName = customerNameList.get(rand.nextInt(customerNameList.size()));

    return randomCustomerName;
}
```

Figure 11: Random CustomerName

- **problemDescription**: Randomly generated string of characters.

```
public String generateRandomProblemDescription() {

    ArrayList<String> problemDescriptionList = new ArrayList<>();
    problemDescriptionList.add("UI-Not-showing-up");
    problemDescriptionList.add("Page-Load-Errors");
    problemDescriptionList.add("Search-Issues");
    problemDescriptionList.add("Shopping-Cart-Errors");
    problemDescriptionList.add("CheckOut-failures");
    problemDescriptionList.add("Account-management-issues");
    problemDescriptionList.add("Recommendation-failures");
    problemDescriptionList.add("UI-too-slow");
    problemDescriptionList.add("Performance-problems");
    problemDescriptionList.add("Cross-scripting-security-issues");
    problemDescriptionList.add("Browser-Compatibility-issues");

    Random rand = new Random();

    String randomProblemDescription = problemDescriptionList.get(rand.nextInt(problemDescriptionList.size()));

    return randomProblemDescription;
}
```

Figure 12: Random Problem Description

- **errorCode**: Uniformly distributed integer from 100 to 150.

```
    public int generateRandomErrorCode() {
        int randomErrorCode = ThreadLocalRandom.current().nextInt(100, 150);
        return randomErrorCode;
    }
```

Figure 13: Random Error Code

- **productType**: Product can fall into categories. It is a set of two products: {“Amazon.com”, “Amazon Web Services (AWS)”}.

```
    ArrayList<String> productTypeList = new ArrayList<>();
    productTypeList.add("Amazon.com");
    productTypeList.add("AWS");

    Random rand = new Random();

    String randomProductType = productTypeList.get(rand.nextInt(productTypeList.size()));
```

Figure 14: Random Product Type

- **Solutions**: Initialized to none. As soon as cases gets resolved, each case is appended with the solution provided by the support engineers. A solution can be provided by any of these three-support personnel TSE, EE, and PSE.

After defining these attributes, the customer case entity is sent through the output port. This allows other components in the simulation to receive a randomized stream of customer cases for processing. The generator provides a flexible way to simulate different volumes and combinations of incoming support requests.

An Example of a Customer case could like this:

Case1: $\{case_{id}, severity, priority, custome, problem_{description}, error_{code}, producttype\}$

$\{1,1, 2, Lockheed\ Martin, \text{“}The\ UI\ is\ not\ visible\text{”}, 404, \text{“}Amazon.com\}$

b. Support Route Center:

The support route center receives customer cases from the case generator and is responsible for routing them to the appropriate downstream support center based on the product type associated with each case. It implements a skill-based routing algorithm to match each incoming case to the correct support team.

The logic works as follows:

- Receive the incoming customer case entity.
- Extract the productType attribute from that same case.
- Extract the Priority and Severity associated with that case.
- Check the productType value:
    - If productType = "Amazon.com", route case to Amazon.com support queue ○ Else if productType = “AWS”, route case to AWS support queue.
- Route the case by sending to appropriate output port:
    - If Amazon.com, send case to output port 1.
    - If AWS, send case to output port 2.
- Check Priority and Severity:
    - If priority = 1 or severity = 1, set escalate to directly to premium support engineer (Tier III).
    - Else, escalate to Technical Support Engineer (Tier I).

The pseudo code for the routing logic of the Support Router Center component is as follows:

```
Initialize output ports:
port1 = Amazon support queue
port2 = AWS support queue While
true:
    Receive incoming customer case
caseType = get case product type
casePriority = get case priority
    caseSeverity = get case severity

        If   caseType   ==   "Amazon.com":
send case to port1
                If casePriority == “1” || caseSeverity == “1”:
                  send case directly to Premium Support Engineer (Amazon.com)
          Else if casePriority == “2” || caseSeverity == “2”|| casePriority == “3” ||
caseSeverity == “3”:
                  send case directly to Escalation Engineer (Amazon.com)
              Else:
      send case to the Technical Support Engineer (Amazon.com)          Else If
caseType == "AWS":
            send case to port2
                If casePriority == 1 or caseSeverity == 1:
                  send case directly to Premium Support Engineer (AWS)
          Else if casePriority == “2” || caseSeverity == “2”|| casePriority == “3” ||
caseSeverity == “3”:
                  send case directly to Escalation Engineer (AWS)
           Else
      send case directly to Technical Support Engineer (AWS )   Release case
End While
```

The route center now accounts for both the product type (skill-based routing) as well as priority/severity (priority-based escalation) when sending each case to the appropriate downstream queue.

c. Support Center (Amazon.com)

1. ***Technical Support Engineer component:***

This is the lowest tier within the support organization, characterized by a limited scope of knowledge that is confined to pre-existing faults or bugs. The system verifies the presence of an internal database or a repository to retrieve client case error codes. If the team possesses knowledge of the solution for a specific code, they proceed to provide the solution to the end-user. Conversely, if the team lacks the answer, they proceed to escalate the case to the Escalation Engineer (Tier II). This component utilizes a Queue<Case> data structure to manage and store cases in a sequential manner. The processing of these situations commences utilizing the First-in-First-out procedure.
When a case is received from the router, the Level I agent or technical support engineer does the following:

- Check the errorCode attribute on the case entity against a database/repository of known error codes and solutions.
- This database or HashMap will contains mappings like this:

Table 1: Internal Repository of Level I Eng.

ERROR CODE	***SOLUTION***
101	Sol I
102	Sol II
103	Sol III
104	Sol IV
105	Sol V
106	Sol VI
107	Sol VII
108	Sol VIII

```
public void preLoadTheInternalDatabase() {
    errorSolutions.put(100, "Reboot_your_computer_to_fix_error_100");
    errorSolutions.put(101, "Reinstall_the_application_to_fix_error_101");
    errorSolutions.put(102, "Clear_cache_and_cookies_to_fix_error_102");
    errorSolutions.put(103, "Check_internet_connectivity_to_fix_error_103");
    errorSolutions.put(104, "Update_network_drivers_to_fix_error_104");
    errorSolutions.put(105, "Error_150_requires_a_software_update");
    errorSolutions.put(106, "Free_up_disk_space_to_fix_error_160");
    errorSolutions.put(107, "Check_file_permissions_to_allow_access");
    errorSolutions.put(108, "Contact_administrator_to_fix_permission_issue");
    errorSolutions.put(109, "Rollback_recent_changes_to_fix_error_190");
    errorSolutions.put(110, "Critical_error,_contact_tech_support");

}
```

Figure 15: Error Solutions Repository for Level I

- If the solution is found in the database, the agent will add the solution text to the case entity. Mark the case as resolved and send the resolved case on its output port to the transducer/performance analyzer.
- If no solution is found in the database, the agent will add escalate the case by sending it to the next tier i.e., Tier II escalation Engineer and mark the case as pending escalation.
- The level I team uses a FIFO queue to store the incoming cases when agents are busy. This ensures cases are picked up in first-come-first served order. The queue capacity is limited, so new incoming cases may be dropped if the queue is full.

2. ***Escalation Engineer component:***

The Escalation Engineers are a group of support personnel who provide Tier II support, possessing enhanced technical expertise and a more comprehensive understanding of the product. When a case is elevated from Level 1, the Escalation Engineer will conduct a more in-depth investigation.

- Thoroughly examine the problem description.
- Utilize the available knowledge base articles and product documentation to effectively assess and prioritize the issue. The task at hand involves discerning whether the issue at hand is more likely to be attributed to a fault or bug within the system, or if it is instead a result of a configuration or usage issue.
- If a usage issue is observed, the engineer can discover pertinent documentation or KB (Knowledge base) articles that can assist in resolving the matter.
- Escalation Engineers possess an extensive repertoire of workarounds and solutions readily available to address specific error codes or problem descriptions. They can offer either product documentation with corresponding page numbers or a knowledge base article containing a viable solution.
- Update the case with investigation details and either recommended actions for the customer or details on the bug filed.

- Mark case resolved or escalated accordingly. If the case is resolved, then the case notes are updated otherwise the case is escalated to level III or Premium Support Engineers.
- Escalation Engineers has a comprehensive list of workaround/solutions at their disposal where for a particular error code or problem description, they can provide either a product documentation with page no, a Kb article with a workable solution.
- In the table 2, we can see that they have a bunch of solutions for a single error code i.e., 110 they have solution that could either come from a product documentation section or from the list of workaround solutions or from knowledgebase articles.
- The level II team uses a FIFO queue to store the incoming cases when agents are busy. This ensures cases are picked up in first-come-first served order. The queue capacity is limited, so new incoming cases may be dropped if the queue is full.

Table 2: Advanced Repository of Level 2 EE

Error Code	Product Doc Section	Workaround Sol	KB Article
110	Section 1.1	Sol 1	“https://kb_article_2”
111	Section 1.2	Sol 2	“https://kb_article_3”
112	Section 2.1	Sol 3	“https://kb_article_4”
113	Section 2.3	Sol 3	“https://kb_article_5”
114	Section 2.4	Sol 4	“https://kb_article_6”
115	Section 3.5	Sol 5	“https://kb_article_7”
116	Section 4.0	Sol 6	“https://kb_article_8”

```
//      List<String> docs116 = Arrays.asList("Section 2.1, Page 70");
//      List<String> workaround116 = Arrays.asList("Restart the application");
//      List<String> kbArticle116 = Arrays.asList("KB1512");
//      solutions116.add(docs116);
//      solutions116.add(workaround116);
//      solutions116.add(kbArticle116);
        advancedErrorSolutions.put(116, solutions116);

        String solutions117 = "Section2.2,Page80_Disable_antivirus_temporarily_KB1322";
//      List<String> docs117 = Arrays.asList("Section 2.2, Page 80");
//      List<String> workaround117 = Arrays.asList("Disable antivirus temporarily");
//      List<String> kbArticle117 = Arrays.asList("KB1322");
//      solutions117.add(docs117);
//      solutions117.add(workaround117);
//      solutions117.add(kbArticle117);
        advancedErrorSolutions.put(117, solutions117);

        String solutions118 = "Section2.3,Page90_Whitelist_website/app_in_firewall_KB1444";
        List<String> docs118 = Arrays.asList("Section 2.3, Page 90");
        List<String> workaround118 = Arrays.asList("Whitelist website/app in firewall");
        List<String> kbArticle118 = Arrays.asList("KB1444");
//      solutions118.add(docs118);
//      solutions118.add(workaround118);
//      solutions118.add(kbArticle118);
        advancedErrorSolutions.put(118, solutions118);

        String solutions119 = "Section2.4,Page100_Switch_to_guest_or_incognito_mode_KB1567";
        List<String> docs119 = Arrays.asList("Section 2.4, Page 100");
        List<String> workaround119 = Arrays.asList("Switch to guest or incognito mode");
        List<String> kbArticle119 = Arrays.asList("KB1567");
//      solutions119.add(docs119);
//      solutions119.add(workaround119);
//      solutions119.add(kbArticle119);
        advancedErrorSolutions.put(119, solutions119);


        String solutions120 = "Section2.5,Page101_Try_wired_network_connection_instead_of_wifi_KB9876";
        List<String> docs120 = Arrays.asList("Section 2.5, Page 101");
        List<String> workaround120 = Arrays.asList("Try wired network connection instead of wifi");
        List<String> kbArticle120 = Arrays.asList("KB9876");
//      solutions120.add(docs120);
//      solutions120.add(workaround120);
//      solutions120.add(kbArticle120);
        advancedErrorSolutions.put(120, solutions120);

        String solutions121 = "Section3.1,Page_120_Flush_DNS_and_renew_IP_address_KB9234";
        List<String> docs121 = Arrays.asList("Section 3.1, Page 120");
        List<String> workaround121 = Arrays.asList("Flush DNS and renew IP address");
```

Figure 16: Advanced and Multiple Solutions for Error Codes

3. ***Premium Support Engineers component:***

They are equipped with a profound understanding of the product's architecture and underlying code base. This level of knowledge empowers them to explore the product's source code to uncover and rectify defects or bugs, ensuring a highly effective and tailored solution for the customer.

The probability of case resolution within Tier III is nearly 100%, as their ultimate responsibility is to resolve cases comprehensively, regardless of their complexity.

4. Feedback mechanisms across Support Tiers:

- Feedback Loop Goals:
    - Enable continuous learning across support levels.
    - Reduce repeat escalations for same issues.

- Level 1 TSE Feedback:
    - For cases escalated from L1, solution is fed back from higher levels.
    - Updates internal knowledge base with new solutions.
    - Equips L1 to better handle similar issues in future.
- Level 2 Feedback:
    - For cases escalated to Level 3, final solution is shared with L2.
    - Opportunity to update workarounds/solutions database.
    - Closes loop on cases they couldn't previously resolve.
- Benefits:
    - Higher support engineers mentor lower tiers.
    - Greatly increases breadth of issues resolvable at L1/L2.
    - Continual learning across tiers.

d. Transducer/Performance Analyzer:

The technical support engineer or level -I is connected with the transducer that provide updates to the transducer or the performance analyzer about details regarding to each resolved case.

The transducer will computer metrics in a systematic manner.

- The resolution rate of Level -I, II and III
- The escalation rate of Level – I, II, and III
- The mean customer satisfaction score of Level – I, II, and III
- The mean time to resolve a case by Level – I, II and III.
- The percentage of cases handled by each level missed the Service Level Agreement.

Finally, we assume that the complexity of both the products are similar i.e. AWS and Amazon.com and thus we assume initially that all their respective support organization have

relevant skills to handle customer cases in their domain expertise. Keeping this in mind, we would like to compare the overall performances between two support centers.

3.2 Model specification

$$Case = CaseID \times Priority \times Severity \times Error\ Message \times Product\ Type \times Customer\ Name \times Problem\ Type \times Error\ Code \times Time\ Elapsed$$

where,

$CaseID \in \mathbb{Z}^+$, whose element represents the unique ID assigned to the case when created.

$Priority \in \{1,2,3,4,5\}$, whose element represents the priority of the case to be resolved. Here priority-1 stands for the highest priority and monotonically decreases in priority as the priority value increases.

$Severity \in \{1,2,3,4,5\}$, whose element represents the case severity …
Let *ASCII* be the set of all ASCII characters officially defined and *STRINGS* be the set of all possible sequences of values belonging to *ASCII.*

$Error\ Message \in STRINGS$, whose element represents the case description entered by the customer.

$Product\ Type \in \{pdt \mid pdt \in \{"Amazon.\ com", "Amazon\ Web\ Services"\}\}$, whose element represents the product corresponding to the case.

$Customer\ Name \in STRINGS$, whose element represents the name of the customer who raised the case.

$Problem\ Type \in \{prb \mid prb \in \{"Configuration","Non-Configuration"\}\}$, whose element represents the type of problem the case is classified.

$Error\ Code \in \{v \mid v \in \mathbb{Z}^+, v \geq 100, v \leq 110\}$, whose element represents the defined error codes for known errors 100-109 and 110 is defined for unknown error.

$Time\ Elapsed \in \{t \mid t \in \mathbb{R}^+, t < \infty\}$, whose element represents the time elapsed from raising the ticket to the present time instance.

$$Solution = Case \times Solution\ Instructions$$

were,
$Case$ represents the set of cases for which solutions are defined.

$Solution\ Instructions \in STRINGS$ represents the set of strings which are the strings containing the sequence of instructions to resolve the case.

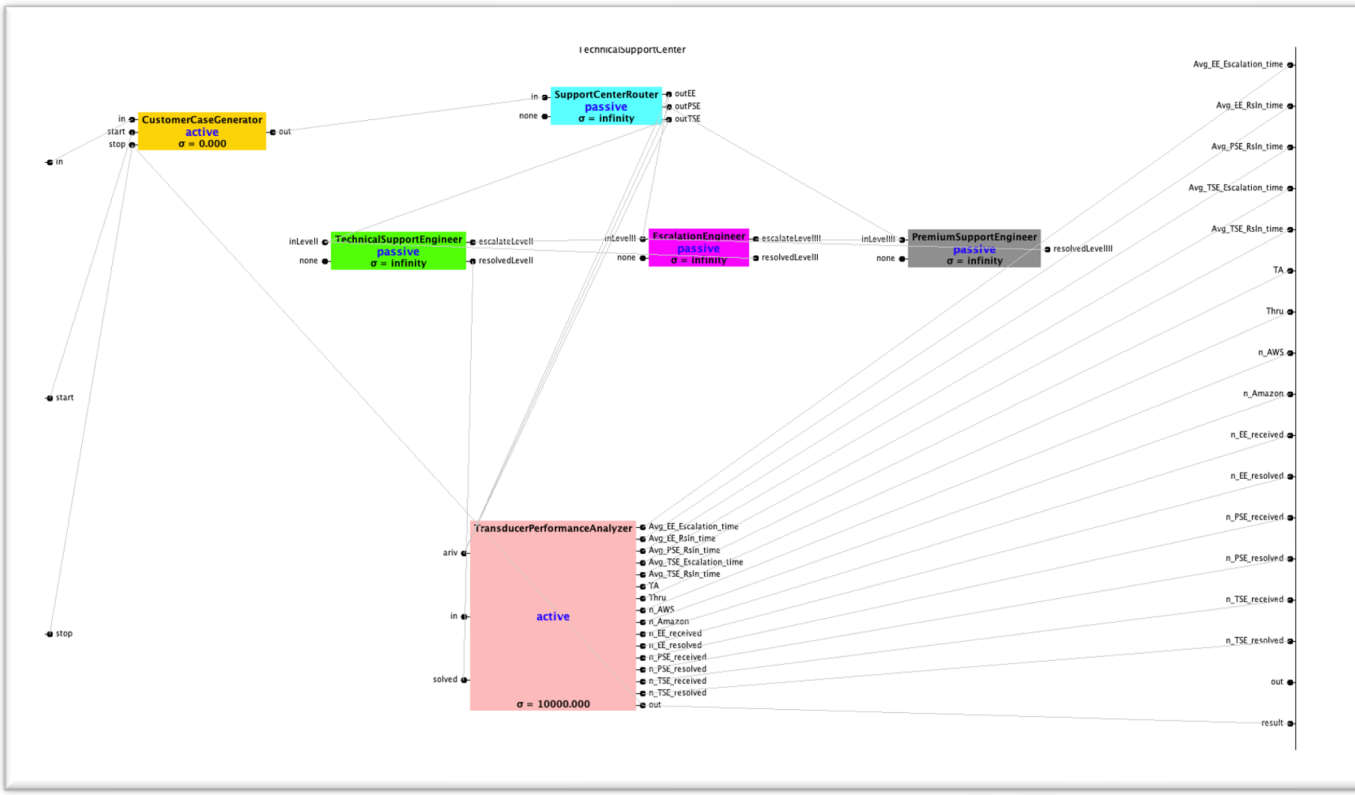


Figure 17 : Technical Support Center Coupled Models

The couplings for the above Figure 17 are explained in the below Figure 18. It has all interconnections that consist of Internal Couplings, External Input Couplings and External Output Couplings.

```
        addCoupling(this,"in",customerCaseGenerator,"in");

        addCoupling(this,"start",customerCaseGenerator,"start");
        addCoupling(this,"stop",customerCaseGenerator,"stop");

        addCoupling(customerCaseGenerator,"out",supportCenterRouter,"in");

        addCoupling(supportCenterRouter,"outTSE",transducer,"ariv");
        addCoupling(supportCenterRouter,"outEE",transducer,"ariv");
        addCoupling(supportCenterRouter,"outPSE",transducer,"ariv");
        addCoupling(supportCenterRouter,"outTSE",technialSupportEngineer,"inLevelI");
        addCoupling(supportCenterRouter,"outPSE",premiumSupportEngineer,"inLevelIII");
        addCoupling(supportCenterRouter,"outEE",escalationEngineer,"inLevelII");
        addCoupling(technialSupportEngineer,"escalateLevelI", escalationEngineer, "inLevelII");
        addCoupling(technialSupportEngineer,"resolvedLevelI", transducer, "solved");
        addCoupling(escalationEngineer, "escalateLevelIII", premiumSupportEngineer, "inLevelIII");
        addCoupling(escalationEngineer,"resolvedLevelII", technialSupportEngineer,"inLevelI");
        addCoupling(premiumSupportEngineer, "resolvedLevelIII", escalationEngineer,"inLevelII");
        addCoupling(transducer,"out",customerCaseGenerator,"stop");


//      addCoupling(,"out",this,"out");
        addCoupling(transducer,"out",this,"result");
        addCoupling(transducer,"n_TSE_resolved",this,"n_TSE_resolved");
        addCoupling(transducer,"n_EE_resolved",this,"n_EE_resolved");
        addCoupling(transducer,"n_PSE_resolved",this,"n_PSE_resolved");
        addCoupling(transducer,"n_TSE_received",this,"n_TSE_received");
        addCoupling(transducer,"n_EE_received",this,"n_EE_received");
        addCoupling(transducer,"n_PSE_received",this,"n_PSE_received");
        addCoupling(transducer,"Avg_TSE_Escalation_time",this,"Avg_TSE_Escalation_time");
        addCoupling(transducer,"Avg_EE_Escalation_time",this,"Avg_EE_Escalation_time");
        addCoupling(transducer,"n_AWS",this,"n_AWS");
        addCoupling(transducer,"n_Amazon",this,"n_Amazon");

        addCoupling(transducer,"TA",this,"TA");
        addCoupling(transducer,"Thru",this,"Thru");
        addCoupling(transducer,"Avg_TSE_Rsln_time",this,"Avg_TSE_Rsln_time");
        addCoupling(transducer,"Avg_EE_Rsln_time",this,"Avg_EE_Rsln_time");
        addCoupling(transducer,"Avg_PSE_Rsln_time",this,"Avg_PSE_Rsln_time");
```

Figure 18: IC, EIC, and EOC

3.2.1 Customer Case Generator Subcomponent Specifications:

$$M_{\text{CSG}} = <X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta>$$

Table 3 : Customer Case Generator Specs

X_b $(Inputs)$	$X = \{(p, v) \mid p \varepsilon IPorts, v \varepsilon Xp$ is the set of input ports and values. $IPorts = \{\text{in}, start, stop\}$
Y_b $(Outputs)$	$Y = \{(p, v) \mid p \in Oports, v \in Yp\}$ $Yout_1 = \{caseCaseID \times Priority \times Severity \times Error\ Message \times Product\ Type \times Customer\ Name \times Problem\ Type \times Error\ Code \times Time\ Elapsed\}$
S $(States)$	$S = Phase \times Sigma \times \text{caseCount}$ $Phase = \{\text{"active", "passive"," finishing"}\}$ $Sigma = \{\sigma \mid \sigma \in \mathbb{R}^+ \cup \{0\}\}$ $caseCount = \mathbb{N}$ as it represents the Natural Numbers
δ_{ext} $(External\ Transition\ Function)$	$R+$ is the set of positive real numbers (e represents the elapsed time) $BAG(M)$ is the set of bags over messages (x is the input bag of messages) $if\ s == passive\ for\ each\ m\ \varepsilon\ x\ if\ m == start\ s: = active\ hold(int\_arr\_time)$ $if\ s == active\ for\ each\ m\ \varepsilon\ x\ if\ m == stop\ s: = finishing$ $Formally:$ $\forall s\ \varepsilon\ S, \forall e\ \varepsilon\ R+, \forall x\ \varepsilon\ BAG(M):$ $if\ s = passive: \forall m\ \varepsilon\ x: if\ m = start: s: = active\ hold(int\_arr\_time)$ $if\ s = active: \forall m\ \varepsilon\ x:$ $if\ m = stop: s: = finishing$ $where:$ $Quantifiers\ \forall\ denote\ "for\ all."$ $s: = newState$ denotes transition to new state $hold(t)$ triggers an internal event after time t.
δ_{int} $(Internal\ Transition\ Function)$	$Formally:$ $\forall s\ \varepsilon\ S,$ $\forall n\ \varepsilon\ N: if\ s = active:$ $n: = n + 1$ $y: = generateCase()$ $output(y)\ s: = active\ hold(int\_arr\_time)$ $else: s: = passive$ $Where:$ $y\ \varepsilon\ Y$ is an output event $output(y)$ generates the output. $hold(t)$ triggers next internal event after time t

δ_{con} $(Confluent\ Transition\ Function)$	$\delta_{con}\ (s, ta(s), x) = \delta_{ext}(\delta_{int}(s), 0, x)$
ta $(Time - Advance\ Function)$	$ta(\text{passive}, \infty) = \infty$ $ta(\text{active}, 10) = 10$
λ $(Output\ Function)$	$\forall n\ \varepsilon\ N,$ $\forall C\ \varepsilon\ BAG(C)$: $n := n + 1$ $c := C(n)$ $y := createOutput(c)\ output(y)$ $Where$: $c\ \varepsilon\ C$ *is the current customerCase* $y \in Y$ *is the output event* $createOutput(c)$ *generates* y *from* c $output(y)$ *sends* y *out*

3.2.2 Case Router Subcomponent Specifications:

$$M_{\text{CR}} = <X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta>$$

Table 4 : Case Router Specifications

X_b $(Inputs)$	$InPorts \in \{"in1"\}$ $XInPorts = \{(p, v) \mid p \in \in X_p\}$ $X_{in1} = \{case \mid case\ Case\}$
Y_b $(Outputs)$	$OutPorts = \{"out1", "out2", "out3", "out4", "out5", "out6"\}$ $Y_p = \{solution \mid solution \in Solution\}^{\forall}\ p \in Outports$ $Y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$
S $(States)$	$S = Phase \times Sigma \times Queue_{amazon} \times Queue_{AWS} \times Case \times CaseSvrtyOrPrty_{amazon} \times rtyOrPrty_{AWS}$ Where, $Phase = \{"Idle", "Routing_n"\}$ $Sigma = \{\sigma \mid \sigma \in \mathbb{R}^+ \cup \{0\}\}$ $Queue_{amazon} = \{sequence \mid sequence \in Case^n, n \in \mathbb{Z}^+ \cup \{0\}\}$ $Queue_{AWS} = \{sequence \mid sequence \in Case^n, n \in \mathbb{Z}^+ \cup \{0\}\}$ $= \{1,2,3,4,5\} \cup \Phi$ $CaseSvrtyOrPrty_{amazon} = \{1,2,3,4,5\} \cup \Phi$ $CaseSvrtyOrPrty_{AWS}$
δ_{ext} $(External\ Transition\ Function)$	$\delta_{ext}(phase,\sigma,Q_{amazon},Q_{AWS},case,p_{amazon},p_{aws},e,x)$ $(phase, \sigma, Q_{amazon}', Q_{AWS}', case, p_{amazon}, p_{aws}, \Phi, x, \Phi)$ $("Routing", t_p, Q_{amazon}, max.severity, x.priority)$ $("Routing", t_p, Q_{amazon}, Q_{AWS}'', case_{AWS}, max(case_{AWS}.severity, case_{AWS}.priority))$ $("Routing", t_p, Q_{amazon}'', \Phi, case_{amazon}, max(case_{amazon}.severity, case_{amazon}.priority))$ $\{("Routing", t_p, (x, Q_{amazon}), Q_{AWS}'', case_{AWS}, max(case_{AWS}.severity, case_{AWS}.priority), \Phi)$ $"Idle" \wedge x.product_{type} = "AWS" \wedge Q_{AWS} \neq \Phi$ if phase $= "Idle"$ $"Idle" \wedge x.product_{type} = "AWS" \wedge Q_{AWS} = \Phi$ if phase $= "Idle"$ $"Idle" \wedge x.product_{type} = "Amazon.in" \wedge Q_{AWS} = \Phi$ if phase $= "Idle"$ $"Amazon.in" \wedge Q_{AWS} \neq \Phi$ if phase $=$ Φ otherwise where, $Q_{amazon}' = \{(x, Q_{amazon}) \text{ if } x.product_{type} = "Amazon.in"; Q_{amazon} \text{ otherwise}$ $Q_{AWS}' = \{(x, Q_{AWS}) \text{ if } x.product_{type} = "AWS"; Q_{AWS} \text{ otherwise}$ $Q_{amazon}'' = (x, Q_{amazon}) \setminus case_{amazon}$ $Q_{AWS}'' = \{(x, Q_{AWS}) \setminus case_{AWS} \text{ if } x.product_{type} = "AWS"$

	$Q_{AWS} \backslash case_{AWS}$ $otherwise$ $\neq \Phi$ $Q^{amazon \mid Q_{amazon}}$, $if\ Q^{amazon}$, $the\ last\ element\ of\ the\ Q_{amazon}.\ case_{amazon} = \{$ x, $otherwise$

3.2.3 Technical Support Engineer (Level-I) Subcomponent Specifications:

$$M_{TSE} = <X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta>$$

Table 5 : TSE Specs

X_b $(Inputs)$	$InPorts = \{"in1", "in2"\}$ $X_{in1} = \{case \mid case \in \quad Case\}$ $= \{(p, v) \mid p \in \quad \in X_p\} \quad X_{in2} = \quad \{redirected\ solution \mid redirected\ solution \in Solution\}$ $XInPorts, v$
Y_b $(Outputs)$	$OutPorts = \{"out \quad \in \quad 1", "out2"\}$ $Y_{out1} = \{case \mid case \quad Case\}$ $Y_{out2} = \{solution \mid solution \in \quad Solution\}$ $Y = \{(p, v) \mid p \in_{OutPorts,\ v} \in Y_p\}$
S $(States)$	$S = Phase \times Sigma \times Task \times Solution \times Queue \times Known\ Defects\ Database$ where $Phase = \quad \{"Idle", "Processing", "Updating\ DB", "Respond"\}$ $Sigma = \{\sigma \mid \sigma \in \mathbb{R}^+ \cup \{0\}\}$ $Task = Case \cup \quad \in \quad Solution \cap \emptyset = \quad \{task \mid task \in \ Case \vee Solution \cap \emptyset\}$

	$Solution = \{sol \mid sol\ Solution\} \quad \in \quad \in \mathbb{Z}^+ \cup \{0\}\}$ $Queue = \{sequence \mid sequence \quad Task^n, n$ $Defect\ Mapping = Error\ Code \times Solution\ Instructions$ $Known\ Defects\ Database = \{defect \mid defect \in \quad Defect\ Mapping^n, n \in \mathbb{Z}^+ \cup \{0\}\}$
δ_{ext} $(External\ Transition\ Function)$	$(phase, \sigma - e, task, sol, Q', kdb), \quad if\ p \quad hase \neq \quad "Idle"$ $\delta_{ext}(phase, \sigma, task, sol, Q, kdb, e, x_{in1}, x_{in2}) = \{ ("Processing", t_p, task', sol', Q'', pdb) \quad oth\ erwise$ where, $_{in2}, x_{in1}, Q) \quad if\ x_{in}(x \quad _1 \neq \Phi, x_{in2} \neq \Phi$ $Q' = \quad (x \quad _1 \neq \Phi, x_{in2} = \Phi \quad _{in1}, Q) \quad if\ x_{in} \quad _{in1} = \Phi, x_{in2} \neq \Phi \quad (x_{in2}, Q) \quad if\ x$ $\{ \quad Q \quad if\ x_{in1} = \Phi, x_{in2} = \Phi$ $task' = Q_{\mid' Q' \mid}$ $Q'' = Q' \backslash task'$
	$= \begin{cases} sol' task' & if\ task' \in \quad Solution \\ \Phi \quad oth\ erwise \end{cases}$
δ_{int} $(Internal\ Transition\ Function)$	$("Idle", \quad \infty, \Phi, \Phi, \Phi \quad , kdb) \quad hase \in \quad if\ p\{"Idle", \quad , Q = \Phi \quad Q_{\mid Q \mid} > \in \ Case\ Solution \quad Q_{\mid Q \mid} > \in$ $"Respond"\}$ $("Processing", t \quad \Phi, Q \backslash < \quad > \quad _p, \quad hase \in \quad Q_{\mid Q \mid}, Q_{\mid Q \mid}, kdb) \quad , Q \neq \Phi, <$ $if\ p \quad \{"Idle", "Respond"\}$ $\delta_{int}(phase, \sigma, task, sol, Q, kdb) = \quad ("Updating\ DB", t_u, Q_{\mid Q \mid}, task, Q \backslash < \ Q_{\mid Q \mid} >, kdb) \quad if\ phase \in \{"Idle", "Respond"\}, Q \neq \Phi, <$ $("Respond", t_{rs}, task, sol_{task}, Q, kdb) \quad if\ phase = \ "Processing"$ $("Respond", t_{ru}, task, sol, Q, kdb') \quad if\ p \quad hase = \ "Updating\ DB"$ $\{$ where, sol_{task} is the searched solution from kdb. It would be Φ if there is no associated key − value pair in kdb. kdb' is the updated kdb with added element of pair $(task.\ errorcode, solutioninstructions)$
δ_{con} $(Confluent\ Transition\ Function)$	$\delta_{con}(s, ta(s), x) = \delta_{ext}(\delta_{int}(s), 0, x)$
ta $(Time - Advance\ Function)$	$ta(phase, \sigma, task, sol, Q, kdb) = \sigma$
λ $(Output\ Function)$	$((\text{"}out1\text{"}, task), (\text{"}out2\text{"}, \Phi)), \quad if\ phase = "Respond", task \in Case, sol \ = \Phi$ $\lambda(phase, \sigma, task, sol, Q, kdb) = \{ \quad ((\text{"}out1\text{"}, \Phi), (\text{"}out2\text{"}, sol)), \quad if\ phase = "Respond", sol \neq \Phi$ $((\text{"}out1\text{"}, \Phi), (\text{"}out2\text{"}, \Phi)), \quad otherwise$

3.2.4 Escalation Engineer (Level-II) Subcomponent Specifications:

$$M_{EE} = <X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta>$$

Table 6: EE Specs

X_b $(Inputs)$	$InPorts = \{"in1", "in2", "in3"\}$ $X_{in1} = \{case \mid case \in Case\}$ $X_{in2} = \{tse\_case \mid tse\_case \in Case\}$ $= \{(p, v) \mid p \in \in X_p\}$ $X_{in3} = \{redirected\ solution \mid redirected\ solution \in Solution\}$ $XInPorts, v$
Y_b $(Outputs)$	$OutPorts = \{"out1", "out2"\}$ $Y_{out1} = \{case \mid case \in Case\}$ $Y_{out2} = \{solution \mid solution \in Solution\}$ $Y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$
S $(States)$	$S = Phase \times Sigma \times Task \times Solution \times Queue \times Product\ Database$ where $Phase = \{"Idle", "Processing", "Updating\ DB", "Respond"\}$ $Sigma = \{\sigma \mid \sigma \in \mathbb{R}^+ \cup \{0\}\}$ $Task = Case \cup \in Solution \cap \emptyset = \{task \mid task \in Case \vee Solution \cap \emptyset\}$ $Solution = \{sol \mid sol\ Solution\} \in \in \mathbb{Z}^+ \cup \{0\}\}$ $Queue = \{sequence \mid sequence\ Task^n, n$ $Defect\ Mapping = Error\ Code \times Solution\ Instructions$ $Product\ Database = \{defect \mid defect \in Defect\ Mapping^n, n \in \mathbb{Z}^+ \cup \{0\}\}$
δ_{ext} $(External\ Transition\ Function)$	$(ph_{ase}, \sigma - e, task, sol, Q', pdb), \quad (("in1", x_{in1}), ("in2", x_{in2}), ("in3", x_{in3})) = \begin{cases} if\ p \quad "Idle" & hase \neq \\ e, ("Processing", t_p, task', sol', Q'', pdb) & otherwise \end{cases}$ $\delta_{ext}(phase, \sigma, task, sol, Q, pdb,$ where,

	$Q' = \begin{cases} (x_{in2}, x_{in1}, Q) & if\ x_{in1} \neq \Phi, x_{in2} \neq \Phi, x_{in3} = \Phi \\ (x_{in1}, Q) & if\ x_{in1} \neq \Phi, x_{in2} = \Phi, x_{in3} = \Phi \\ (x_{in2}, Q) & if\ x_{in1} = \Phi, x_{in2} \neq \Phi, x_{in3} = \Phi \\ Q & if\ x_{in1} = \Phi, x_{in2} = \Phi, x_{in3} = \Phi \\ (x_{in3}, x_{in2}, x_{in1}, Q) & if\ x_{in1} \neq \Phi, x_{in2} \neq \Phi, x_{in3} \neq \Phi \\ (x_{in3}, x_{in1}, Q) & if\ x_{in1} \neq \Phi, x_{in2} = \Phi, x_{in3} \neq \Phi \\ (x_{in3}, x_{in2}, Q) & if\ x_{in1} = \Phi, x_{in2} \neq \Phi, x_{in3} \neq \Phi \\ (x_{in3}, Q) & if\ x_{in1} = \Phi, x_{in2} = \Phi, x_{in3} \neq \Phi \end{cases}$ $task' = Q'_{\lvert Q' \rvert}$ $Q'' = Q' \backslash task'$ $sol' = \begin{cases} task' & if\ task' \in Solution \\ \Phi & otherwise \end{cases}$
δ_{int} $(Internal\ Transition\ Function)$	$\delta_{int}(phase, \sigma, task, sol, Q, pdb) = \begin{cases} ("Idle", \infty, \Phi, \Phi, \Phi, pdb) & if\ phase \in \{"Idle", "Respond"\}, Q = \Phi \\ ("Processing", t_p, Q_{\lvert Q \rvert}, \Phi, Q \backslash < Q_{\lvert Q \rvert} >, pdb) & if\ phase \in \{"Idle", "Respond"\}, Q \neq \Phi, < Q_{\lvert Q \rvert} > \in Case \\ ("Updating\ DB", t_u, Q_{\lvert Q \rvert}, Q_{\lvert Q \rvert}, Q \backslash < Q_{\lvert Q \rvert} >, pdb) & if\ phase \in \{"Idle", "Respond"\}, Q \neq \Phi, < Q_{\lvert Q \rvert} > \in Solution \\ ("Respond", t_{rs}, task, sol_{task}, Q, pdb) & if\ phase = "Processing" \\ ("Respond", t_{ru}, task, sol, Q, pdb') & if\ phase = "Updating\ DB" \end{cases}$ where, sol_{task} is the searched solution from pdb. It would be Φ if there is no associated key − value pair in pdb. pdb' is the updated pdb with added element of pair $(task.errorcode, solutioninstructions)$
δ_{con} $(Confluent\ Transition\ Function)$	$\delta_{con}(s, ta(s), x) = \delta_{ext}(\delta_{int}(s), 0, x)$
ta $(Time - Advance\ Function)$	$ta(phase, \sigma, task, sol, Q, pdb) = \sigma$
λ $(Output\ Function)$	$\lambda(phase, \sigma, task, sol, Q, pdb) = \begin{cases} (("out1", task), ("out2", \Phi)), & if\ phase = "Respond", task \in Case, sol = \Phi \\ (("out1", \Phi), ("out2", sol)), & if\ phase = "Respond", sol \neq \Phi \\ (("out1", \Phi), ("out2", \Phi)), & otherwise \end{cases}$

3.2.5 Premium Support Engineer (Level-III) Subcomponent Specifications:

$$M_{PSE} = < X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta >$$

Table 7: PSE Specs

X_b $(Inputs)$	$InPorts = \{"in1", "in2"\}$ $X_{in1} = \{case \mid case \in \quad \in \quad Case\}$ $= \{(p, v) \mid p \in \quad \in X_p\}$ $X_{in2} = \{ee\_case \mid ee\_case \quad Case\}$ $XInPorts, v$
Y_b $(Outputs)$	$OutPorts = \{"out1"\}$ $Y_{out1} = \{solution \mid solution \in \quad Solution\}$ $y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$

S $(States)$	$S = Phase \times Sigma \times Task \times Solution \times Queue$ where $Phase = \{"Idle", "Processing", "Respond"\}$ $Sigma = \{\sigma \mid \sigma \in \mathbb{R}^+ \cup \{0\}\}$ $Task = \{sol \mid sol \in Solution\}$ $Case = \{task \mid task \in Case\}$ $\in \mathbb{Z}^+ \cup \{0\}\}$ $Queue = \{sequence \mid sequence \quad Task^n, n$
δ_{ext} $(External\ Transition\ Function)$	$\delta_{ext}(phase, \sigma, task, sol, Q, e, x_{in1}, x_{in2}) = \{$ $(phase, \sigma - e, task, sol, Q')$ $if\ p\ hase \neq$ $"Idle"$ $("Processing", t_p, task', \Phi, Q'', pdb)$ where, (x_{in2}, x_{in1}, Q) $= {}_1 \quad {}_1 \neq \Phi$ $if\ x_{in}$ $(x_{in}$ $Q'_{in}, Q)$ $if\ x_{in}$ $\Phi, x_{in2} \neq \Phi$ $= \Phi$ $_{in1} = \Phi, x_{in2} \neq \Phi$ $, x_{in}$ (x_{in2}, Q) $if\ x$ Q $if\ x_{in1} = \Phi, x_{in2} = \Phi$
	$task' = Q_{\lvert Q' \rvert}$ $Q'' = Q' \backslash task'$
δ_{int} $(Internal\ Transition\ Function)$	$\delta_{int}(phase, \sigma, task, sol, Q) = \{$ $("Idle", \infty, \Phi, \Phi, \Phi)$ $if\ phase \in \{"Idle", "Respond"\}, Q = \Phi$ $("Processing", t_p, Q_{\lvert Q \rvert}, \Phi, Q \backslash < Q_{\lvert Q \rvert}$ $>)$ $if\ phase \in \{"Idle", "Respond"\}, Q \neq \Phi$ $("Respond", t_{rs}, task, sol_{task}, Q)$ $if\ phase =$ $"Processing"$ where, sol_{task} *is the solution generated for task. In the context of this simulation, let this be string* $[task + "Solution"]$.
δ_{con} $(Confluent\ Transition\ Function)$	$\delta_{con}(s, ta(s), x) = \delta_{ext}(\delta_{int}(s), 0, x)$
ta $(Time - Advance\ Function)$	$ta(phase, \sigma, task, sol, Q) = \sigma$
λ $(Output\ Function)$	$\lambda(phase, \sigma, task, sol, Q) = ("out1", sol)$

3.2.6 Transducer Subcomponent Specifications:

$$M_{\mathrm{T}} = <X_b, Y_b, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta>$$

Table 8: Transducer Specs

X_b $(Inputs)$	$X = \{arrive, solved\}$ $arrive = \{case \mid case \in Cases\}$ where Cases is the set of customer cases $solved = \{tse\_case \mid tse\_case \in Cases\}$ where tse_case is a TSE customer case that was solved
Y_b $(Outputs)$	$Y = \{out, TA, Thru, n\_TSE\_received, n\_EE\_received, n\_PSE\_received,$ $Avg_{TSE_{Escalation_{time}}}, Avg_{EE_{Escalation_{time}}}, n_{TSE_{resolved}}, n_{EE_{resolved}},$ $n_{PSE_{resolved}}, Avg_{TSE_{Rsln_{time}}}, Avg_{EE_{Rsln_{time}}},$ $Avg_{PSE_{Rsln_{time}}},$ $n_{AWS},$ n_{Amazon} $\}$ Where: out: output customer case TA: timing alignment Thru: throughput n_TSE_received: TSE cases received. n_EE_received: EE cases received. n_PSE_received: PSE cases received. Avg_TSE_Escalation_time: Avg TSE escalation time. Avg_EE_Escalation_time: Avg EE escalation time. n_TSE_resolved: TSE cases resolved. n_EE_resolved: EE cases resolved. n_PSE_resolved: PSE cases resolved. Avg_TSE_Rsln_time: Avg TSE resolution time. Avg_EE_Rsln_time: Avg EE resolution time. Avg_PSE_Rsln_time: Avg PSE resolution time. n_AWS: AWS cases. n_Amazon: Amazon case.
S $(States)$	$s = \{passive, active\}$
δ_{ext} $(External\ Transition\ Function)$	$deltext: R+ \times BAG(X) \rightarrow S$ Where: $R+$ is the set of positive real numbers (e is the elapsed time) $BAG(X)$ is the set of bags over the input event set X S is the set of states deltext transitions the state $s \in S$ based on the input bag of events $x \in BAG(X)$: $\forall e \in R+, \forall x \in BAG(X)$: $clock := clock + e$ $for\ each\ a \in x\ if\ a \in arrive$ add arrival time of a to arrived if $a \in solved$ update stats based on a update turnaround time add solved time for a to solved Formally:

	$t := current\_time - arrived(a)$ $total\_ta := total\_ta + t\ solved(a) := current\_time$ *Where*: *a is an input event* $arrived(a)$ *stores the arrival time of a* $solved(a)$ *stores the solved time of a* $update\_stats()$ *updates the statistics* *t is the turnaround time*
δ_{int} $(Internal\ Transition\ Function)$	*deltint transitions the state* $s \in S$ *as follows*: $clock := clock + sigma\ s := passive$ *Formally*: $\forall s \in S: clock := clock + sigma\ s := passive$ *Where*: *sigma is the time advance* $:=$ *denotes transition to next state* *passive is the passive state*
ta $(Time - Advance\ Function)$	$ta(S, \sigma) = \sigma$
λ $(Output\ Function)$	$message\ m = \textbf{new}\ message();$ $content\ con1 = makeContent("TA", \textbf{new}\ entity(" " + compute\_TA()));$ $content\ con2 = makeContent("out", \textbf{new}\ entity(count.toString()));$ $content\ con3 = makeContent("Thru", \textbf{new}\ entity(" " + compute\_Thru()));$ $content\ con4 = makeContent("n\_TSE\_resolved", \textbf{new}\ doubleEnt(n\_TSE\_resolved));$ $content\ con5 = makeContent("n\_EE\_resolved", \textbf{new}\ doubleEnt(n\_EE\_resolved));$ $content\ con6 = makeContent("n\_PSE\_resolved", \textbf{new}\ doubleEnt(n\_PSE\_resolved));$ $content\ con7 = makeContent("n\_TSE\_received", \textbf{new}\ doubleEnt(n\_TSE\_received));$ $content\ con8 = makeContent("n\_EE\_received", \textbf{new}\ doubleEnt(n\_EE\_received));$ $content\ con9 = makeContent("n\_PSE\_received", \textbf{new}\ doubleEnt(n\_PSE\_received));$ $content\ con10 = makeContent("Avg\_TSE\_Rsln\_time", \textbf{new}\ doubleEnt(total\_TSE\_rsln\_time/n\_TSE\_resolved));$ $content\ con11 = makeContent("Avg\_EE\_Rsln\_time", \textbf{new}\ doubleEnt(total\_EE\_rsln\_time/n\_EE\_resolved));$ $content\ con12 = makeContent("Avg\_PSE\_Rsln\_time", \textbf{new}\ doubleEnt(total\_PSE\_rsln\_time/n\_PSE\_resolved));$ $content\ con13 = makeContent("Avg\_TSE\_Escalation\_time", \textbf{new}\ doubleEnt(total\_TSE\_escalation\_time/(n\_TSE\_received - n\_TSE\_resolved)));$ $content\ con14 = makeContent("Avg\_EE\_Escalation\_time", \textbf{new}\ doubleEnt(total\_EE\_escalation\_time/(n\_EE\_received - n\_EE\_resolved)));$ $content\ con15 = makeContent("n\_AWS", \textbf{new}\ doubleEnt(n\_AWS));$ $content\ con16 = makeContent("n\_Amazon", \textbf{new}\ doubleEnt(n\_Amazon));$

3.3 Model implementation

There are following classes that we have implemented that represents the component of a coupled model in a technical support center.

- CustomerCaseGenerator
- SupportCenterRouter
- TechnicalSupportEngineer
- EscalationEngineer
- PremiumSupportEngineer
- Transducer

The customer case is a Java Pojo file.

- CustomerCase

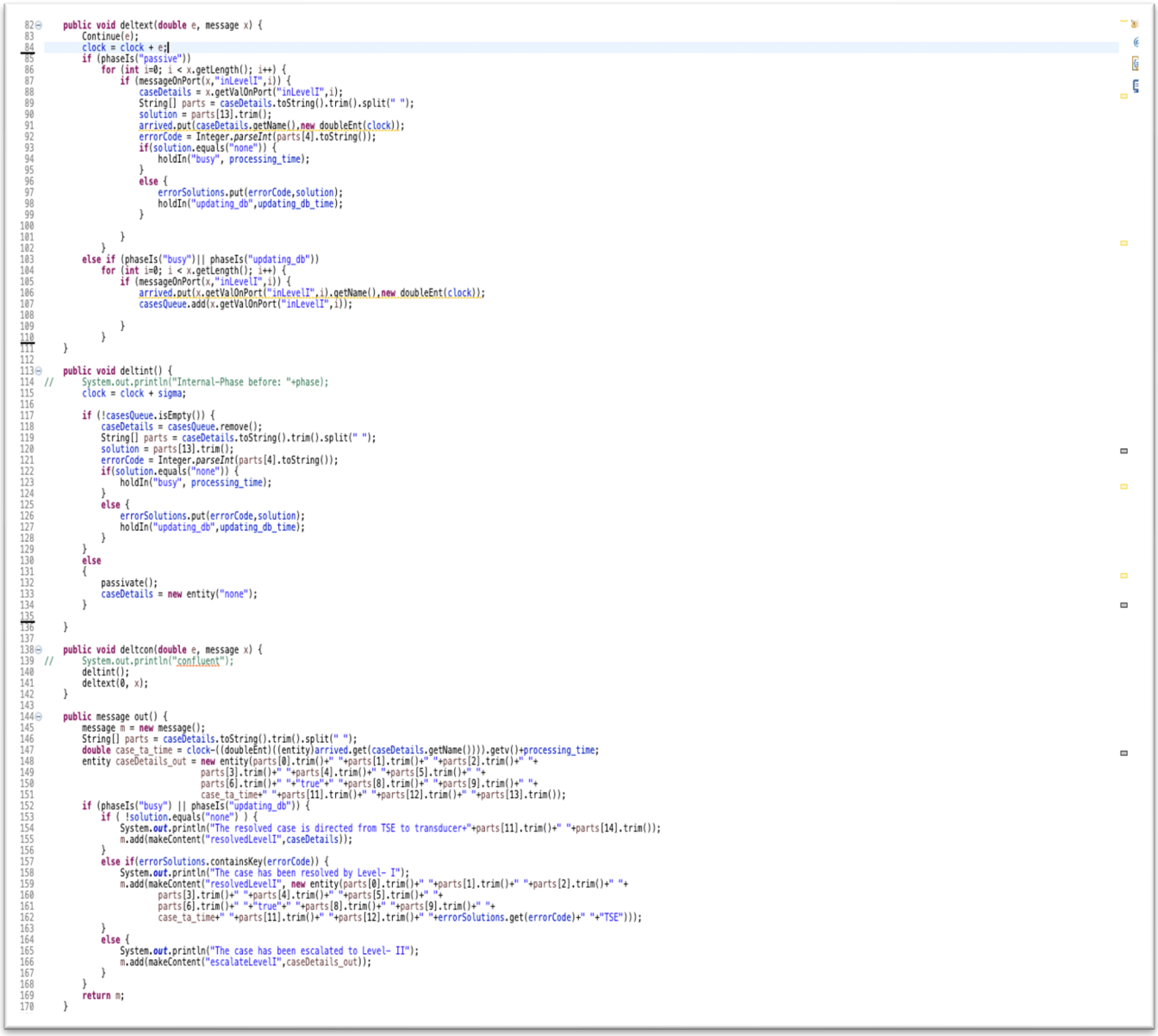
Few code snippets are provided below:

- **SupportCenterRouter External Transition Function**

```
    public void deltext(double e, message x) {
        Continue(e);
//      It is because of all the simulators has to be executed. What is the minimum of nextTN of these three atomic models
//

//      System.out.println("The elapsed time of the processor is" + e);
//      System.out.println("*****************************************");
//      System.out.println("external-Phase before: "+phase);



        if (phaseIs("passive"))
            for (int i = 0; i < x.getLength(); i++)
                if (messageOnPort(x, "in", i)) {
                    caseDetails = x.getValOnPort("in", i);
//                  System.out.println(caseDetails.toString());
                    String[] parts = caseDetails.toString().trim().split(" ");
                    severity = Integer.parseInt(parts[1].trim());
                    priority = Integer.parseInt(parts[2].trim());
                    String customerName = parts[3].trim();
//                  System.out.println(severity);
//                  System.out.println(customerName);
                    if((severity == 1) || (priority == 1) || (customerName.equalsIgnoreCase("LockheedMartin"))) {
                        casesRoutedToLevelIII.add(caseDetails);
                        System.out.println("*** Cases Routed to Level III ***");
                        System.out.println(caseDetails);
                        routedToLevelIII++;
                    }
                    else if((severity == 2) || (priority == 2) || (severity == 3) || (priority == 3)) {
                        casesRoutedToLevelII.add(caseDetails);
                        System.out.println("*** Cases Routed to Level II ***");
                        System.out.println(caseDetails);
                        routedToLevelII++;
                    }
                    else {
                        casesRoutedToLevelI.add(caseDetails);
                        System.out.println("** Cases Routed to Level I ***");
                        System.out.println(caseDetails);
                        routedToLevelI++;
                    }
                    holdIn("busy", processing_time);
//                  System.out.println("processing tiem of proc is"
//                          + processing_time);
                }

//      System.out.println("external-Phase after: "+phase);
    }
```

Code Snippet 1: External Transition Function of Router

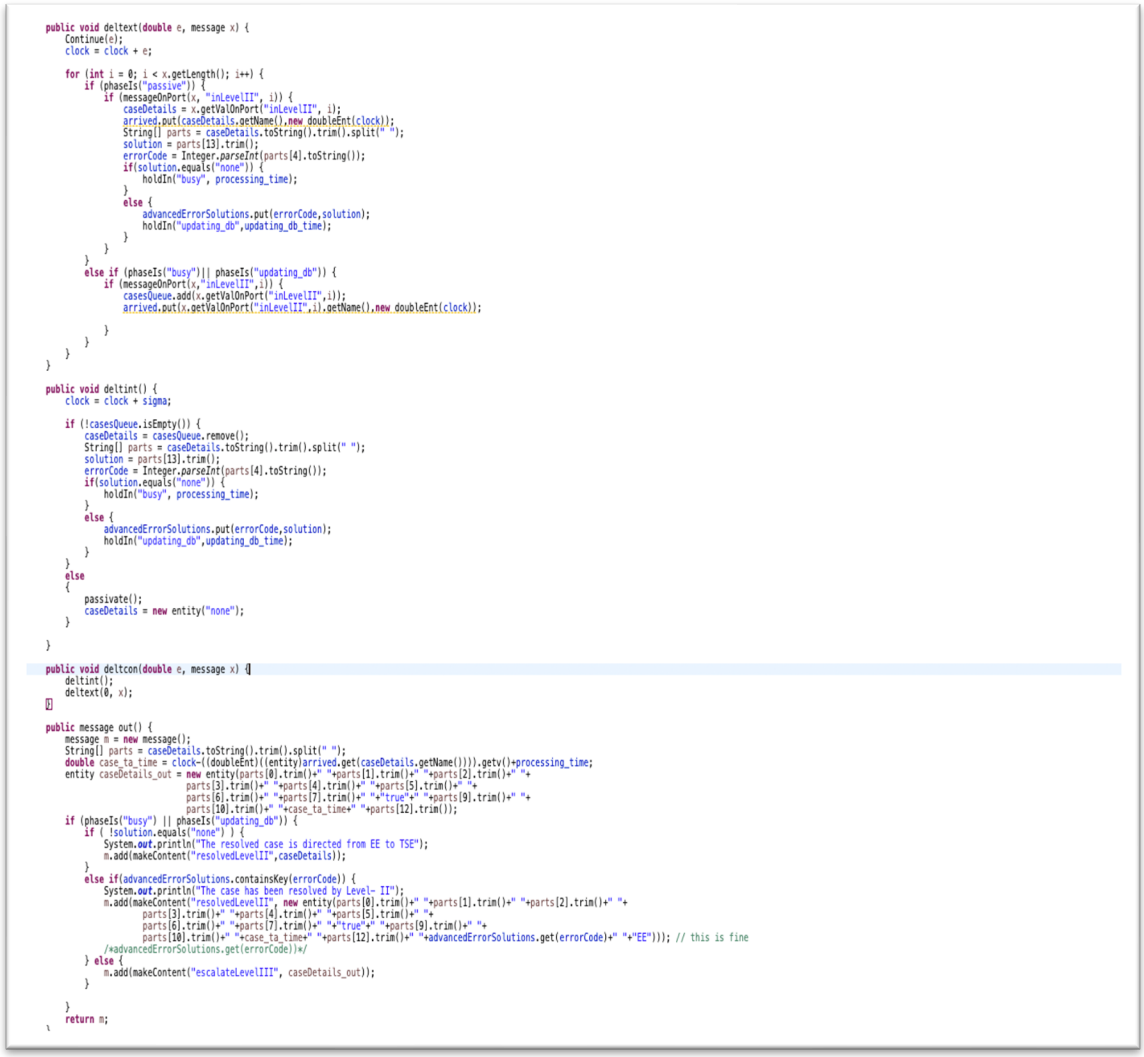
- **TechnicalSupportEngineer external, internal, confluent and output function:**



```
    public void deltext(double e, message x) {
        Continue(e);
        clock = clock + e;
        if (phaseIs("passive"))
            for (int i=0; i < x.getLength(); i++) {
                if (messageOnPort(x,"inLevelI",i)) {
                    caseDetails = x.getValOnPort("inLevelI",i);
                    String[] parts = caseDetails.toString().trim().split(" ");
                    solution = parts[13].trim();
                    arrived.put(caseDetails.getName(),new doubleEnt(clock));
                    errorCode = Integer.parseInt(parts[4].toString());
                    if(solution.equals("none")) {
                        holdIn("busy", processing_time);
                    }
                    else {
                        errorSolutions.put(errorCode,solution);
                        holdIn("updating_db",updating_db_time);
                    }

                }
            }
        else if (phaseIs("busy")|| phaseIs("updating_db"))
            for (int i=0; i < x.getLength(); i++) {
                if (messageOnPort(x,"inLevelI",i)) {
                    arrived.put(x.getValOnPort("inLevelI",i).getName(),new doubleEnt(clock));
                    casesQueue.add(x.getValOnPort("inLevelI",i));

                }
            }
    }

    public void deltint() {
//      System.out.println("Internal-Phase before: "+phase);
        clock = clock + sigma;

        if (!casesQueue.isEmpty()) {
            caseDetails = casesQueue.remove();
            String[] parts = caseDetails.toString().trim().split(" ");
            solution = parts[13].trim();
            errorCode = Integer.parseInt(parts[4].toString());
            if(solution.equals("none")) {
                holdIn("busy", processing_time);
            }
            else {
                errorSolutions.put(errorCode,solution);
                holdIn("updating_db",updating_db_time);
            }
        }
        else
        {
            passivate();
            caseDetails = new entity("none");
        }

    }

    public void deltcon(double e, message x) {
//      System.out.println("confluent");
        deltint();
        deltext(0, x);
    }

    public message out() {
        message m = new message();
        String[] parts = caseDetails.toString().trim().split(" ");
        double case_ta_time = clock-((doubleEnt)((entity)arrived.get(caseDetails.getName()))).getv()+processing_time;
        entity caseDetails_out = new entity(parts[0].trim()+" "+parts[1].trim()+" "+parts[2].trim()+" "+
                                 parts[3].trim()+" "+parts[4].trim()+" "+parts[5].trim()+" "+
                                 parts[6].trim()+" "+"true"+" "+parts[8].trim()+" "+parts[9].trim()+" "+
                                 case_ta_time+" "+parts[11].trim()+" "+parts[12].trim()+" "+parts[13].trim());
        if (phaseIs("busy") || phaseIs("updating_db")) {
            if ( !solution.equals("none") ) {
                System.out.println("The resolved case is directed from TSE to transducer+"+parts[11].trim()+" "+parts[14].trim());
                m.add(makeContent("resolvedLevelI",caseDetails));
            }
            else if(errorSolutions.containsKey(errorCode)) {
                System.out.println("The case has been resolved by Level- I");
                m.add(makeContent("resolvedLevelI", new entity(parts[0].trim()+" "+parts[1].trim()+" "+parts[2].trim()+" "+
                        parts[3].trim()+" "+parts[4].trim()+" "+parts[5].trim()+" "+
                        parts[6].trim()+" "+"true"+" "+parts[8].trim()+" "+parts[9].trim()+" "+
                        case_ta_time+" "+parts[11].trim()+" "+parts[12].trim()+" "+errorSolutions.get(errorCode)+" "+"TSE")));
            }
            else {
                System.out.println("The case has been escalated to Level- II");
                m.add(makeContent("escalateLevelI",caseDetails_out));
            }
        }
        return m;
    }
```

Code Snippet 2: Functionality of TSE

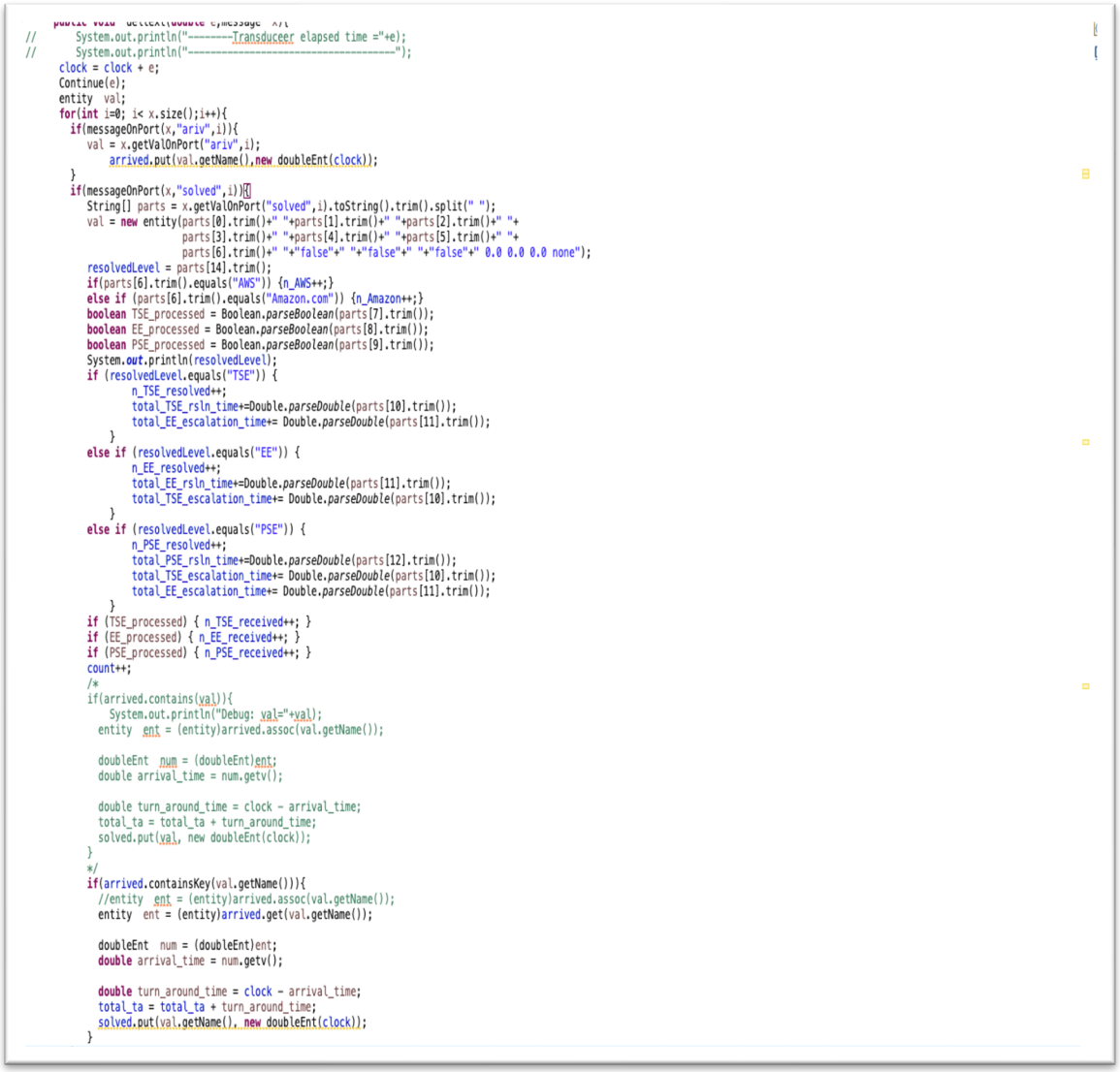
- **Core functionality of EE that includes external, internal, confluent and output function:**



```
public void deltext(double e, message x) {
    Continue(e);
    clock = clock + e;

    for (int i = 0; i < x.getLength(); i++) {
        if (phaseIs("passive")) {
            if (messageOnPort(x, "inLevelII", i)) {
                caseDetails = x.getValOnPort("inLevelII", i);
                arrived.put(caseDetails.getName(),new doubleEnt(clock));
                String[] parts = caseDetails.toString().trim().split(" ");
                solution = parts[13].trim();
                errorCode = Integer.parseInt(parts[4].toString());
                if(solution.equals("none")) {
                    holdIn("busy", processing_time);
                }
                else {
                    advancedErrorSolutions.put(errorCode,solution);
                    holdIn("updating_db",updating_db_time);
                }
            }
        }
        else if (phaseIs("busy")|| phaseIs("updating_db")) {
            if (messageOnPort(x,"inLevelII",i)) {
                casesQueue.add(x.getValOnPort("inLevelII",i));
                arrived.put(x.getValOnPort("inLevelII",i).getName(),new doubleEnt(clock));

            }
        }
    }
}

public void deltint() {
    clock = clock + sigma;

    if (!casesQueue.isEmpty()) {
        caseDetails = casesQueue.remove();
        String[] parts = caseDetails.toString().trim().split(" ");
        solution = parts[13].trim();
        errorCode = Integer.parseInt(parts[4].toString());
        if(solution.equals("none")) {
            holdIn("busy", processing_time);
        }
        else {
            advancedErrorSolutions.put(errorCode,solution);
            holdIn("updating_db",updating_db_time);
        }
    }
    else
    {
        passivate();
        caseDetails = new entity("none");
    }

}

public void deltcon(double e, message x) {
    deltint();
    deltext(0, x);
}

public message out() {
    message m = new message();
    String[] parts = caseDetails.toString().trim().split(" ");
    double case_ta_time = clock-((doubleEnt)((entity)arrived.get(caseDetails.getName()))).getv()+processing_time;
    entity caseDetails_out = new entity(parts[0].trim()+" "+parts[1].trim()+" "+parts[2].trim()+" "+
                             parts[3].trim()+" "+parts[4].trim()+" "+parts[5].trim()+" "+
                             parts[6].trim()+" "+parts[7].trim()+" "+"true"+" "+parts[9].trim()+" "+
                             parts[10].trim()+" "+case_ta_time+" "+parts[12].trim());
    if (phaseIs("busy") || phaseIs("updating_db")) {
        if ( !solution.equals("none") ) {
            System.out.println("The resolved case is directed from EE to TSE");
            m.add(makeContent("resolvedLevelII",caseDetails));
        }
        else if(advancedErrorSolutions.containsKey(errorCode)) {
            System.out.println("The case has been resolved by Level- II");
            m.add(makeContent("resolvedLevelII", new entity(parts[0].trim()+" "+parts[1].trim()+" "+parts[2].trim()+" "+
                  parts[3].trim()+" "+parts[4].trim()+" "+parts[5].trim()+" "+
                  parts[6].trim()+" "+parts[7].trim()+" "+"true"+" "+parts[9].trim()+" "+
                  parts[10].trim()+" "+case_ta_time+" "+parts[12].trim()+" "+advancedErrorSolutions.get(errorCode)+" "+"EE"))); // this is fine
            /*advancedErrorSolutions.get(errorCode))*/
        } else {
            m.add(makeContent("escalateLevelIII", caseDetails_out));
        }

    }
    return m;
}
```

Code Snippet 3: Functionality of EE

- **Transducer Core functionality that includes external transition function**



```
public void deltext(double e,message x){
//      System.out.println("--------Transduceer elapsed time ="+e);
//      System.out.println("-----------------------------------");
     clock = clock + e;
     Continue(e);
     entity  val;
     for(int i=0; i< x.size();i++){
       if(messageOnPort(x,"ariv",i)){
          val = x.getValOnPort("ariv",i);
              arrived.put(val.getName(),new doubleEnt(clock));
       }
       if(messageOnPort(x,"solved",i)){
          String[] parts = x.getValOnPort("solved",i).toString().trim().split(" ");
          val = new entity(parts[0].trim()+" "+parts[1].trim()+" "+parts[2].trim()+" "+
                           parts[3].trim()+" "+parts[4].trim()+" "+parts[5].trim()+" "+
                           parts[6].trim()+" "+"false"+" "+"false"+" "+"false"+" 0.0 0.0 0.0 none");
          resolvedLevel = parts[14].trim();
          if(parts[6].trim().equals("AWS")) {n_AWS++;}
          else if (parts[6].trim().equals("Amazon.com")) {n_Amazon++;}
          boolean TSE_processed = Boolean.parseBoolean(parts[7].trim());
          boolean EE_processed = Boolean.parseBoolean(parts[8].trim());
          boolean PSE_processed = Boolean.parseBoolean(parts[9].trim());
          System.out.println(resolvedLevel);
          if (resolvedLevel.equals("TSE")) {
                n_TSE_resolved++;
                total_TSE_rsln_time+=Double.parseDouble(parts[10].trim());
                total_EE_escalation_time+= Double.parseDouble(parts[11].trim());
             }
          else if (resolvedLevel.equals("EE")) {
                n_EE_resolved++;
                total_EE_rsln_time+=Double.parseDouble(parts[11].trim());
                total_TSE_escalation_time+= Double.parseDouble(parts[10].trim());
             }
          else if (resolvedLevel.equals("PSE")) {
                n_PSE_resolved++;
                total_PSE_rsln_time+=Double.parseDouble(parts[12].trim());
                total_TSE_escalation_time+= Double.parseDouble(parts[10].trim());
                total_EE_escalation_time+= Double.parseDouble(parts[11].trim());
             }
          if (TSE_processed) { n_TSE_received++; }
          if (EE_processed) { n_EE_received++; }
          if (PSE_processed) { n_PSE_received++; }
          count++;
          /*
          if(arrived.contains(val)){
              System.out.println("Debug: val="+val);
            entity  ent = (entity)arrived.assoc(val.getName());

            doubleEnt  num = (doubleEnt)ent;
            double arrival_time = num.getv();

            double turn_around_time = clock - arrival_time;
            total_ta = total_ta + turn_around_time;
            solved.put(val, new doubleEnt(clock));
          }
          */
          if(arrived.containsKey(val.getName())){
            //entity  ent = (entity)arrived.assoc(val.getName());
            entity  ent = (entity)arrived.get(val.getName());

            doubleEnt  num = (doubleEnt)ent;
            double arrival_time = num.getv();

            double turn_around_time = clock - arrival_time;
            total_ta = total_ta + turn_around_time;
            solved.put(val.getName(), new doubleEnt(clock));
          }
```

Code Snippet 4: Transducer Ext function

4 SIMULATION EXPERIMENTS

Some of the key metrics for an individual support centers are calculated as follows:

R_1 : *No of cases resolved by level I or tier I*

E_1: *No of cases escalated by level I to higher tiers*

AE_1: *No of cases escalated to higher tiers but could have been avoided*

N_1 : *Total no of cases recieved by level I*

S_i: *Represents the customer satsifaction score of case handled by level I for* i^{th} *case*

T_i: *Time to resolve a case by level I for the* i^{th} *case*

Based on the above values, we can predetermine some of the key metrics associated only at the level I:

- $The\ resolution_{rate}\ of\ level\ I \rightarrow Resolution_{rate}: Equation(I)$ $\frac{R_1}{N_1} * 100 -$
- $\frac{1}{1} * 100 -$
- E ○ $The\ escalation_{rate} of\ level\ I \rightarrow Escalation_{rate}$: $_N Equation(II)$
- AE^1 ○ $The\ escalation_{avoidablerate} of\ level\ I \rightarrow Avoidable_{rate}$: $\_\_\_{N1} * 100 - Equation(III)$
- s ○ $The\ mean\ Customer_{satisfactionscore}\ of\ level\ I \rightarrow CSAT_{level1}$: $\sum_{j=1}^{n} \frac{j}{n} - Equation(IV)$
- ○ $The\ mean\ time\ to\ resolve\ a\ case\ by\ level\ I \rightarrow Mean_{resolve}$: $\sum_{j=1}^{n} \frac{j}{t} - Equation\ (V)$

- **Key Metrics for Level II (Escalation Engineers):**

R_2 : *No of cases resolved by level II or tier II*

E_2: *No of cases escalated by level II to premium tiers*

AE_2: *No of cases escalated to higher tiers but could have been avoided*

N_2 : *Total no of cases resolved by level II*

S_i: *Represents the customer satsifaction score of case handled by level II for* i^{th} *case*

T_i: *Time to resolve a case by level II for the* i^{th} *case*

Based on the above values, we can predetermine some of the key metrics associated only at the level II.

R^2 ○ $The\ resolution_{rate}\ of\ level\ II \rightarrow Resolution_{rate}$: $N2 * 100 - Equation(I)$ ○ $The\ escalation_{rate} of\ level\ II \rightarrow Escalation_{rate}$: $EN2_2 * 100 - Equation(II)$ ○ $The\ escalation_{avoidablerate} of\ level\ II \rightarrow Avoidable_{rate}$: $A\_\_\_NE2^2 * 100 - Equation(III)$

○ $The\ mean\ Customer_{satisfactionscore}\ of\ level\ II \rightarrow CSAT_{level2}$: $Equation(IV)$ $\sum_{j=1}^{n} \frac{j}{s} - n$

○ $The\ mean\ time\ to\ resolve\ a\ case\ by\ level\ II \rightarrow Mean_{resolve}$: $Equation\ (V)$ $\sum_{j=1}^{n} \frac{j}{t} - n$

- **Key Metrics for Level III (Premium Support Engineers):**

Since this is the last level and this team can't escalate cases to any higher level thus we don't need to consider the escalation rate.

R^3 ○ $The\ resolution_{rate}\ of\ level\ III \rightarrow Resolution_{rate}$: $N3 * 100 - Equation(I)$

○ $The\ mean\ Customer_{satisfactionscore}\ of\ level\ III \rightarrow CSAT_{level2}$: $Equation(II)$ $\sum_{j=1}^{n} \frac{j}{s} - n$

○ $The\ mean\ time\ to\ resolve\ a\ case\ by\ level\ III \rightarrow Mean_{resolve}$: $Equation\ (III)$ $\sum_{j=1}^{n} \frac{j}{t} - n$

- **(% of cases each level missed the SLA):**

Finally we want to capture a really important metrics to track cases that missed their SLA(Service Level Duration) in a simulated technical support center:

- When generating each customer case, we would assign a random SLA duration(e.g., 4 hours, 24 hours etc) ○ $SLA_{caseid} \rightarrow random(\ 1{,}24\ )$: $Generates\ a\ random\ no\ from\ 1\ to\ 24\ hours.$
- The SLA timestamp attribute will be added to the case entity on its creation time.
- For example:

$$Case\ created\ at\ 10\ AM\ with\ 4\ hour\ SLA$$

$$SLA\ timestamp: 10\ AM + 4\ hour -> 2\ PM$$

- As the case passes through each support tier, check if current time > SLA timestamp
- If current time exceeds the case SLA, a tag "Missed SLA" will be noted on the customer case notes.
- The "Missed SLA" flag will be tracked at each tier as cases are resolved.
- Report metrics:

$$SLA_{missed} \rightarrow Level\ I$$

$$SLA_{missed} \rightarrow Level\ II$$

$$SLA_{missed} \rightarrow Level\ III$$

Calculate % of SLA missed at each tier.

5 EVALUATIONS

This section is for the final report.

6 CONCLUSIONS

This section is for the final report.

7 REFERENCES

[1] S. Robinson, "Optimizing the customer experience in service system design" in proc. 2013 Winter Simulations Conference, pp. 1513-1524

[2] J. Smith, K. Zhou, and M. Lee, "Agent-based modeling of a help desk service system," in Proc. 2016 IEEE Systems and Information Engineering Design Symposium, pp. 137-141.

[3] K. Rustogi and V. G. Kulkarni, "A simulation framework for service centers," in Proc. 2015 Annual Reliability and Maintainability Symposium, pp. 1-7

A APPENDICES

Appendix A - Sample Knowledge Base Article

Article Title: Troubleshooting Error Code 110
Product Category: Cloud Drive Backup Software
Error Code: 110
Title: Failed backup job
Description: This error occurs when Cloud Drive Backup is unable to complete the backup job due to failure in writing backup data to the designated location. Some common reasons include insufficient storage space, corrupted backup destination, network connectivity issues, or problems with backup account credentials and permissions.
Troubleshooting Steps:

1. Verify storage space in backup destination. Make sure there is adequate free disk space to complete the backup job. The amount required depends on the size of the backup.
2. Check connectivity and credentials for backup storage location. If backing up to cloud or networked storage, confirm you can access the location through file explorer. Re-enter credentials if connectivity failures occur.
3. Scan storage device for errors and repair corrupt disk drives if necessary, before backup location. This resolves any surface problems impeding data writes.
4. Adjust backup job settings to exclude unnecessary large files or file types that may be timing out the job. Test with smaller backup set first.
5. As last resort tries new backup destination media in case current hardware itself is faulty and failing written backups intermittently.

Related Errors: 107, 112
Keywords: Failed backup, incomplete backup, storage space errors, credential issues, network troubleshooting.

Appendix B - Sample Support Ticket

Ticket Details
Status: Resolved Assigned to: Mary S (Tier 2 Support) Created by: John W

Contact Details
Name: John W Email: johnw@acmecompany.com Phone: 555-0123
Company: Acme Company Account #: 2451

Product Details
Product: AcmeCloud Data Backup Service Version: 2.1.0

Action History
Sep 5, 2022, 3:24 PM: Ticket created Sep 5, 2022, 3:29 PM: Assigned to Mary S Sep 5, 2022 5:11 PM: Mary reached out for additional specifics Sep 5, 2022 5:24 PM: John provided config file and screenshots Sep 6, 2022 10:16 AM: Mary provided solution steps Sep 6, 2022 11:03 AM: John confirmed the issue is now resolved

Issue Description

AcmeCloud backup jobs failing on the first Tuesday of every month. Error shown is "Backup unsuccessful due to storage limit errors." Our account has 500 GB allocated backup storage, with only 5GB currently used. The failures only occur on the first Tuesday, all other day's backups run fine. Need resolution urgently as this also causes our internal backups to fail.

Resolution

The issue was caused by a recently added monthly maintenance cron job running every 1st Tuesday across AcmeCloud shared backup servers. The job temporarily takes 50% storage offline for maintenance, causing active jobs hitting storage limit errors. As a workaround, your backup schedule has been shifted to Wednesdays. The underlying issue has been escalated by AcmeCloud engineering team for permanent resolution later.

Appendix C – List of tools used.

- Eclipse IDE
- Mac os
- DEVS-SUITE_6.10
- JRE v 11
- JavaFx v 19
- Astah - UML