|  |  |
| --- | --- |
| **Members**: | **CYS**: BSCOE 4-6 |
| Bacolod, Irel Kian C. | **Subject**: Trends in Software Development |
| De Guzman, John Carlo B. | Processes |
| Llena, Ram Ymac E. | **Date**: March 13, 2025 |
| Palmes, Roderick B. |  |
| Panuga, Kathlene Ann M. |  |
| Reyes, Erika Jane T. |  |
| Rodriguez, Beverly Ann L. |  |
| Viteño, Jerico James F. |  |

**Laboratory Activity: Understanding Software Trends and Computational Processes**

**Objective:**

1. To understand the scope of computing and its significance in modern technology.
2. To explore the relationships between computing domains and Metascience Expression Language.
3. To implement basic concepts of Computing (C/Δ) and Computational Implementation (Δ/C).
4. To analyze and apply interaction models in computing.

**Activity 1: Introduction to Computing and its Scope**

**Instructions:**

1. Research the historical evolution of computing and its impact on various industries.
2. Identify at least three different domains where computing plays a vital role (e.g., artificial intelligence, cybersecurity, embedded systems, etc.).
3. Create a mind map that illustrates the interconnectivity of computing across different domains.
4. Write a reflection on how computing influences everyday applications and emerging trends.

**Expected Output:**

* A detailed mind map.
* A one-page reflection on computing’s role in modern applications.

**MIND MAP**

**A diagram of a computer

AI-generated content may be incorrect.**

**REFLECTION**

Computing has become the foundation of nearly every aspect of modern life, revolutionizing the way we work, communicate, learn, and interact with the world around us. From smartphones and smart homes to medical diagnostics and industrial automation, computing plays a vital role in shaping today’s digital age. It has evolved from basic number-crunching machines to sophisticated systems that power artificial intelligence, cloud infrastructure, and embedded devices. This rapid evolution has made computing not just a support tool, but a central driver of innovation across multiple sectors.

One of the most visible impacts of computing is in Artificial Intelligence (AI), where machines are now capable of learning, adapting, and making decisions. AI technologies have enabled personalized recommendations, autonomous vehicles, virtual assistants, and intelligent healthcare systems. At the same time, Cybersecurity has become more critical than ever, as digital threats continue to grow alongside our reliance on technology. Robust encryption methods, ethical hacking techniques, and secure network protocols protect sensitive data and preserve trust in digital systems.

In the physical world, Embedded Systems have brought intelligence to everyday devices. From wearable health monitors to smart appliances, these systems integrate hardware and software seamlessly, often connected to the Cloud for real-time data processing and storage. Cloud Computing allows users and organizations to access powerful computing resources on demand, reducing the need for expensive infrastructure and enabling scalable solutions.

What makes computing even more powerful is the interconnectivity between these domains. AI enhances cybersecurity through intelligent threat detection systems, while embedded systems generate real-time data that is transmitted and managed through cloud platforms. Additionally, cloud computing provides the foundation for AI applications and supports the remote operation of embedded devices. This interconnected ecosystem allows for smarter systems, faster innovation, and greater efficiency in solving real-world problems.

Overall, computing is no longer just a field of study—it is a driving force behind the digital transformation of society. As technology continues to advance, computing will remain at the heart of emerging trends, shaping the future of industries and enhancing everyday life for individuals around the globe.

**Activity 2: Domains, Relationships, and the Metascience Expression Language**

**Instructions:**

1. Study the concept of Metascience Expression Language and how it structures relationships between computational domains.
2. Identify key relationships in computing (e.g., hardware-software interaction, cloud computing relationships, etc.).
3. Using a programming language of choice (Python, Java, or C++), write a simple program to demonstrate a structured relationship (e.g., a basic client-server interaction).

**Expected Output:**

* A brief explanation of Metascience Expression Language with examples.
* Source code demonstrating a structured relationship.

**EXPLANATION**

**What is MEL?**

Metascience Expression Language (MEL) is used to describe domains, relationships, and computational structures in a formal and standardized way. It plays a key role in enabling effective communication and modeling in scientific and software development contexts.

It is not a programming language, but rather a semantic modeling tool that helps express how different components in computing interact, much like an abstract blueprint for systems engineering.

**Purpose of MEL:**

* To create a structured representation of complex systems.
* To improve interoperability and clarity in multi-domain computational environments.
* To make interactions between components explicit and standardized.

**Core of Elements of MEL:**

|  |  |
| --- | --- |
| **Element** | **Description** |
| Entity | A component (e.g., Application, Server, Sensor) |
| Attribute | Metadata or property of an entity (e.g., OS, version) |
| Relation | How two entities are connected (e.g., runsOn, communicatesWith) |
| Operation | Action or process (e.g., sendData, invokeAPI) |

**Examples:**

|  |
| --- |
| **Cloud Computing System** |
| Entity: WebApp  Entity: CloudVM  Entity: Database  Relation: runsOn(WebApp, CloudVM)  Relation: connectsTo(WebApp, Database)  Attribute: CloudVM.provider = AWS  Attribute: Database.type = SQL |

**SOURCE CODE:**

**Client-Server Relationship in Python**

|  |  |
| --- | --- |
| **server.py** | **client.py** |
| import socket  # Define server socket  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  server\_socket.bind(('localhost', 8080)) # Attribute: port = 8080  server\_socket.listen(1)  print("Server is ready and listening on port 8080...")  # Accept client connection  client\_socket, client\_address = server\_socket.accept()  print(f"Client connected from: {client\_address}")  # Receive data from client  message = client\_socket.recv(1024).decode()  print("Message from client:", message)  # Send response  client\_socket.send("Hello from Server!".encode())  # Close sockets  client\_socket.close()  server\_socket.close() | import socket  # Define client socket  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # Connect to the server  client\_socket.connect(('localhost', 8080)) # Relation: communicatesWith(ServerService)  # Send message  client\_socket.send("Hello from Client!".encode())  # Receive response  response = client\_socket.recv(1024).decode()  print("Message from server:", response)  # Close the connection  client\_socket.close() |

**Activity 3: Implementing Computing (C/Δ) and Computational Implementation (Δ/C)**

**Instructions:**

1. Research the principles of Computing (C/Δ) and Computational Implementation (Δ/C).
2. Implement a small project demonstrating both paradigms (e.g., algorithmic computation vs. data-driven computation).
3. Analyze the differences between the two approaches and document the findings.

**Expected Output:**

* A working prototype of a computing implementation using both paradigms.
* A comparison report between C/Δ and Δ/C approaches.

**PROTOTYPE**

|  |  |
| --- | --- |
| **c\_delta\_computing.py** | **delta\_c\_computational.py** |
| # Computing Implementation (C/Δ) - Fibonacci using algorithm  def fibonacci(n):  sequence = [0, 1]  for i in range(2, n):  next\_val = sequence[i - 1] + sequence[i - 2]  sequence.append(next\_val)  return sequence  # Example usage  terms = 10  print(f"Fibonacci Sequence (First {terms} terms):")  print(fibonacci(terms)) | # Computational Implementation (Δ/C) - React to changing temperature  import random  def monitor\_temperature():  for \_ in range(10):  temp = random.randint(20, 100) # Simulated sensor input  print(f"Current Temperature: {temp}°C")  if temp > 80:  print("⚠️ ALERT: Overheating detected!")  elif temp < 30:  print("✅ Temperature is low. System running efficiently.")  else:  print("ℹ️ Normal operating temperature.")  # Example usage  monitor\_temperature() |

**COMPARISON REPORT**

**1. Definition and Principle**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **C/Δ (Computing Implementation)** | **Δ/C (Computational Implementation)** |
| Core Driver | Algorithmic design initiates change | Data or change triggers computational response |
| Behavior | Static, predictable process | Dynamic context-sensitive response |

**2. Implementation Comparison**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **C/Δ (Computing Implementation)** | **Δ/C (Computational Implementation)** |
| Input | Number of terms (static) | Random dynamic temperature (simulated sensor) |
| Output | Fibonacci sequence | System decisions based on input |
| Role of Algorithm | Main control of logic and flow | Secondary; algorithm adapts to data |

**3. Use Cases**

|  |  |
| --- | --- |
| **Paradigm** | **Real-World Examples** |
| C/Δ | Sorting algorithms, Calculators, Simulators |
| Δ/C | Fibonacci sequence  System decisions based on input |
| Role of Algorithm | Smart thermostats, IoT-based alert systems, Stock market dashboards |

**Activity 4: Interaction in Computing**

**Instructions:**

1. Explore different interaction models in computing (e.g., user-computer interaction, machine-to-machine interaction).
2. Develop a simple interactive application (e.g., chatbot, basic UI interaction in Python, JavaScript, or C#).
3. Test and evaluate the efficiency and usability of your interactive model.

**Expected Output:**

* Source code of the interactive application.
* An evaluation report on the usability and efficiency of the interaction model.

**INTERACTIVE APPLICATION**

|  |
| --- |
| **simple\_chatbot.py** |
| # Simple Interactive Chatbot using User-Computer Interaction model  def chatbot():  print("🤖 Hello! I am ChatBot. Type 'bye' to exit.")  while True:  user\_input = input("You: ").strip().lower()    if user\_input == "bye":  print("ChatBot: Goodbye! 👋")  break  elif "hello" in user\_input or "hi" in user\_input:  print("ChatBot: Hi there! How can I help you today?")  elif "help" in user\_input:  print("ChatBot: Sure! I can answer simple questions or chat with you.")  elif "how are you" in user\_input:  print("ChatBot: I'm just code, but thanks for asking! 😄")  else:  print("ChatBot: Sorry, I don't understand. Try saying 'help'.")  # Run the chatbot  chatbot() |

**EVALUATION REPORT**

**1. Usability Evaluation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Remarks** |
| Ease of Use | Simple input/output text interface | User-friendly for beginners |
| Clarity of Responses | Easy-to-understand replies | Conversational style makes it engaging |
| Navigation | Loop with exit keywoard bye | No complex commands required |
| Learning Curve | Very low | Ideal for first-time user |

**2. Efficiency Evaluation**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Remarks** |
| Response Time | Instant (real-time input/output) | Efficient for small-scale use |
| Resource Usae | Minimal (runs on terminal only) | No GUI or heavy memory load |
| Scalability | Can handle simple conversations | Can be extended with NLP or databases |

* This chatbot demonstrates a basic UCI model.
* It's lightweight, fast, and user-friendly, suitable for beginners.
* Can be enhanced by:
* Adding GUI (e.g., tkinter)
* Integrating with APIs (e.g., Weather, Wikipedia)
* Using NLP libraries (e.g., NLTK or ChatGPT API)

**Assessment Criteria:**

1. **Understanding and Analysis (30%)** – Clarity and depth of explanations.
2. **Implementation (40%)** – Correctness and efficiency of implemented programs.
3. **Creativity and Innovation (20%)** – Originality in approach and application.
4. **Documentation (10%)** – Clear and concise reports.

**Submission Guidelines:**

* Submit all source codes and documentation in a single ZIP file.
* Include a README file explaining execution steps.
* Deadline: March 16, 2025

Upload your Lab Activity 1 in this linkhttps://docs.google.com/forms/d/e/1FAIpQLSdusIJhVhSQ5NWAyQzfT92mH4MRnscHeLV-va88WKmu\_izAOg/viewform?usp=dialog