Universidad Autónoma de
Ciudad Juárez
Instituto de Ingeniería y
Tecnología

Lenguaje de modelación unificada (UML)

Programación integrativa

Dr. Javitt Higmar Nahitt Padilla Franco

Raúl Ernesto Pérez Bárcenas, Mat. 148661

Cd. Juárez, Chih.

24 de marzo de 2018

Introducción

Los diagramas UML son una forma muy estandarizada de documentación de ingeniería de software que facilita el acople y mantenibilidad de un sistema informático y/o robusto. Dado que permite manejar de manera abstracta el fundamento de cualquier software a través de las diversas herramientas que este posee, no obstante, es importante remarcar que es una buena práctica la generación de este tipo de documentos en el acervo profesional, por lo que nos remarca y distingue frente a otro tipo de desarrolladores y/o programadores, pero para especializarse en dicha toma años de práctica y continuidad en diversos proyectos continuos.

El potencial del UML, esta en hacer de lo complejo y esporádico algo tangible y comprensible de manera visual por cualquier desarrollador de la industria del Software.

Diagrama de Casos de Uso

El objetivo de este diagrama es ejemplificar a través de actores y casos la relación existente entre agentes externos de la aplicación, así como elementos intermedios que interactúan para el proceso de ejecución de este. Es una esquematización muy ilustrativa que cumple el fin de desglosar los principales elementos de interacción en el software a desarrollar.

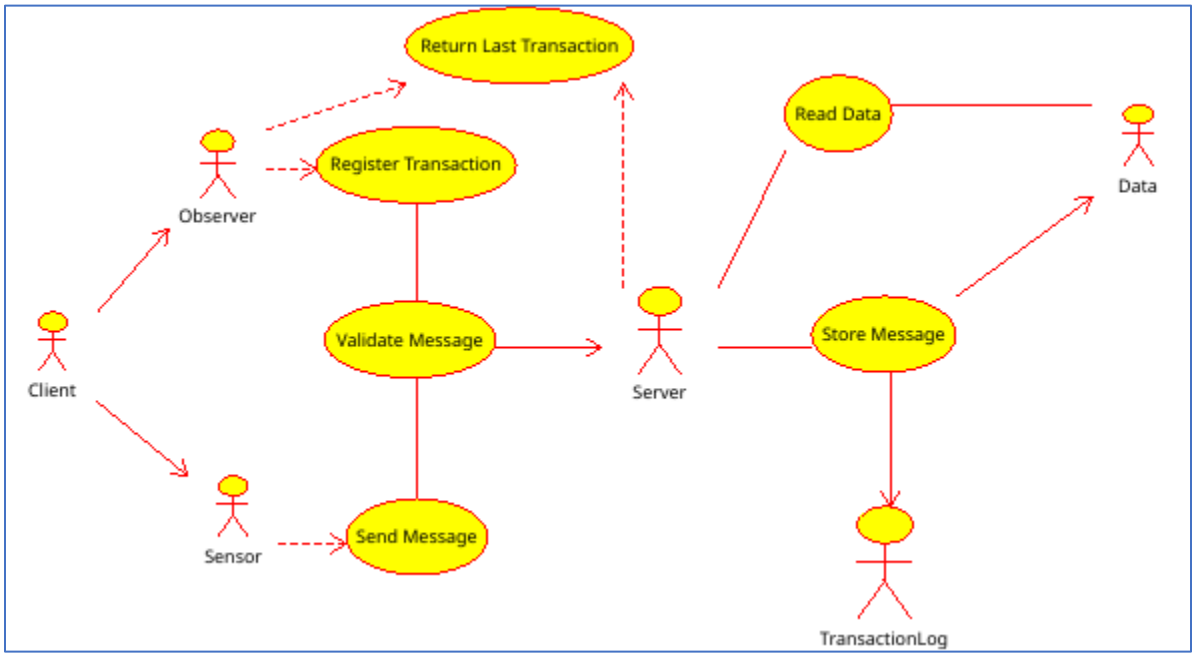


Diagrama de clases

Dicho diagrama representa la estructura de clases del programa y/o software, así como la dependencia y generación de instancias, al igual que el tipo de datos y/o funciones contenidas en ellas a través de una simbología especifica y delimitada como los símbolos (+) para público y (-) para privado.

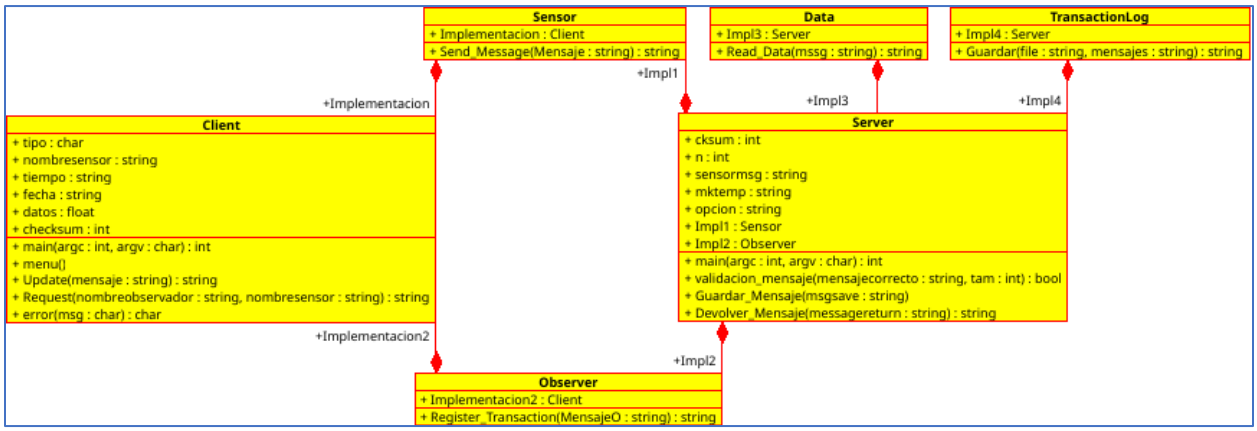
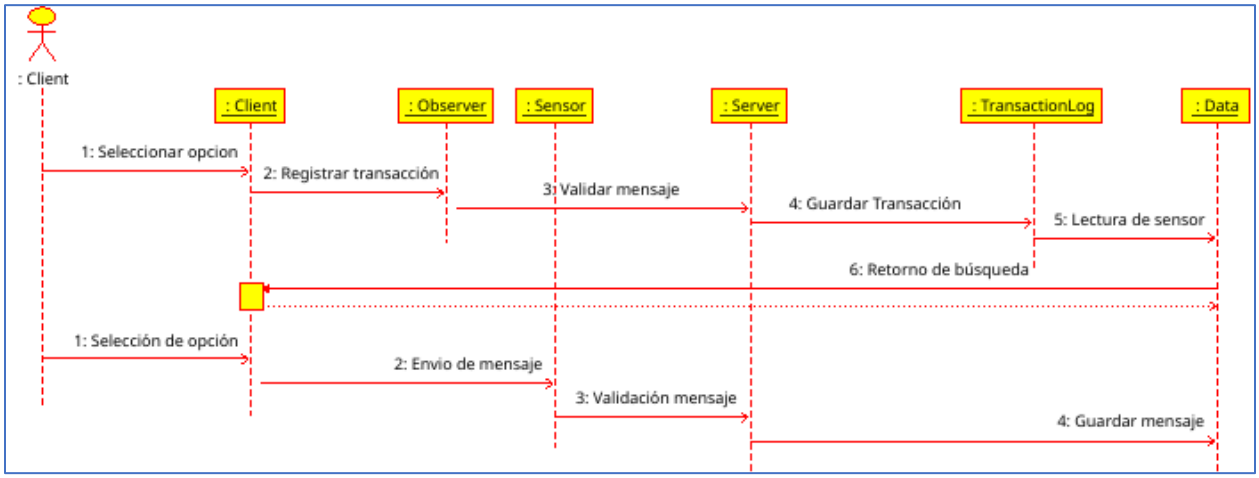


Diagrama de secuencia

En este caso la representación de este sistema busca contemplar el orden de interacción a través del paso de mensajes o bien ejecución de funciones determinadas para un fin determinado, suelen enumerarse, pero depende exclusivamente del encargado de gestión de dichos diagramas. Interactúan actores del diagrama de casos de uso y objetos delimitados del diagrama de casos, se recomienda basarse del diagrama de clases para una estructura de recuperación de mensajes correcta, si así se requiere.



Código generado

A raíz de los diagramas plasmados anteriormente con la herramienta Umbrello se genero la codificación abstracta de su implementación en el lenguaje Java específicamente.

Client.java

```
/**
 * Class Client
 */
public class Client {

  //
  // Fields
  //

  public char tipo;
  public String nombresensor;
  public String tiempo;
  public String fecha;
  public float datos;
  public int checksum;

  //
  // Constructors
  //
  public Client () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of tipo
   * @param newVar the new value of tipo
   */
  public void setTipo (char newVar) {
    tipo = newVar;
  }

  /**
   * Get the value of tipo
   * @return the value of tipo
   */
  public char getTipo () {
```

```
  return tipo;
}

/**
 * Set the value of nombresensor
 * @param newVar the new value of nombresensor
 */
public void setNombresensor (String newVar) {
  nombresensor = newVar;
}

/**
 * Get the value of nombresensor
 * @return the value of nombresensor
 */
public String getNombresensor () {
  return nombresensor;
}

/**
 * Set the value of tiempo
 * @param newVar the new value of tiempo
 */
public void setTiempo (String newVar) {
  tiempo = newVar;
}

/**
 * Get the value of tiempo
 * @return the value of tiempo
 */
public String getTiempo () {
  return tiempo;
}

/**
 * Set the value of fecha
 * @param newVar the new value of fecha
 */
public void setFecha (String newVar) {
  fecha = newVar;
}

/**
 * Get the value of fecha
 * @return the value of fecha
 */
public String getFecha () {
  return fecha;
}
```

```
/**
 * Set the value of datos
 * @param newVar the new value of datos
 */
public void setDatos (float newVar) {
  datos = newVar;
}

/**
 * Get the value of datos
 * @return the value of datos
 */
public float getDatos () {
  return datos;
}

/**
 * Set the value of checksum
 * @param newVar the new value of checksum
 */
public void setChecksum (int newVar) {
  checksum = newVar;
}

/**
 * Get the value of checksum
 * @return the value of checksum
 */
public int getChecksum () {
  return checksum;
}

//
// Other methods
//

/**
 * @return       int
 * @param        argc
 * @param        argv
 */
public int main(int argc, char argv)
{
}


/**
 */
public void menu()
{
}
```

```
    /**
     * @return       String
     * @param        mensaje
     */
    public String Update(String mensaje)
    {
    }


    /**
     * @return       String
     * @param        nombreobservador
     * @param        nombresensor
     */
    public String Request(String nombreobservador, String nombresensor)
    {
    }


    /**
     * @return       char
     * @param        msg
     */
    public char error(char msg)
    {
    }


}
```

Data.java

```
/**
 * Class Data
 */
public class Data {

  //
  // Fields
  //

  public Server Impl3;

  //
  // Constructors
  //
  public Data () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of Impl3
   * @param newVar the new value of Impl3
   */
  public void setImpl3 (Server newVar) {
    Impl3 = newVar;
  }

  /**
   * Get the value of Impl3
   * @return the value of Impl3
   */
  public Server getImpl3 () {
    return Impl3;
  }

  //
  // Other methods
  //

  /**
   * @return       String
   * @param        mssg
   */
  public String Read_Data(String mssg)
  {
  }
}
```

Observer.java

```
/**
 * Class Observer
 */
public class Observer {

  //
  // Fields
  //

  public Client Implementacion2;

  //
  // Constructors
  //
  public Observer () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of Implementacion2
   * @param newVar the new value of Implementacion2
   */
  public void setImplementacion2 (Client newVar) {
    Implementacion2 = newVar;
  }

  /**
   * Get the value of Implementacion2
   * @return the value of Implementacion2
   */
  public Client getImplementacion2 () {
    return Implementacion2;
  }

  //
  // Other methods
  //

  /**
   * @return       String
   * @param        MensajeO
   */
  public String Register_Transaction(String MensajeO)
  {
  }
}
```

Sensor.java

```
/**
 * Class Sensor
 */
public class Sensor {

  //
  // Fields
  //

  public Client Implementacion;

  //
  // Constructors
  //
  public Sensor () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of Implementacion
   * @param newVar the new value of Implementacion
   */
  public void setImplementacion (Client newVar) {
    Implementacion = newVar;
  }

  /**
   * Get the value of Implementacion
   * @return the value of Implementacion
   */
  public Client getImplementacion () {
    return Implementacion;
  }

  //
  // Other methods
  //

  /**
   * @return       String
   * @param        Mensaje
   */
  public String Send_Message(String Mensaje)
  {
  }
}
```

Server.java

```
/**
 * Class Server
 */
public class Server {

  //
  // Fields
  //

  public int cksum;
  public int n;
  public String sensormsg;
  public String mktemp;
  public String opcion;
  public Sensor Impl1;
  public Observer Impl2;

  //
  // Constructors
  //
  public Server () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of cksum
   * @param newVar the new value of cksum
   */
  public void setCksum (int newVar) {
    cksum = newVar;
  }

  /**
   * Get the value of cksum
   * @return the value of cksum
   */
  public int getCksum () {
    return cksum;
  }

  /**
   * Set the value of n
   * @param newVar the new value of n
   */
  public void setN (int newVar) {
    n = newVar;
  }

  /**
   * Get the value of n
   * @return the value of n
   */
  public int getN () {
    return n;
  }

  /**
   * Set the value of sensormsg
   * @param newVar the new value of sensormsg
   */
  public void setSensormsg (String newVar) {
    sensormsg = newVar;
  }

  /**
   * Get the value of sensormsg
   * @return the value of sensormsg
```

```
 */
public String getSensormsg () {
  return sensormsg;
}

/**
 * Set the value of mktemp
 * @param newVar the new value of mktemp
 */
public void setMktemp (String newVar) {
  mktemp = newVar;
}

/**
 * Get the value of mktemp
 * @return the value of mktemp
 */
public String getMktemp () {
  return mktemp;
}

/**
 * Set the value of opcion
 * @param newVar the new value of opcion
 */
public void setOpcion (String newVar) {
  opcion = newVar;
}

/**
 * Get the value of opcion
 * @return the value of opcion
 */
public String getOpcion () {
  return opcion;
}

/**
 * Set the value of Impl1
 * @param newVar the new value of Impl1
 */
public void setImpl1 (Sensor newVar) {
  Impl1 = newVar;
}

/**
 * Get the value of Impl1
 * @return the value of Impl1
 */
public Sensor getImpl1 () {
  return Impl1;
}

/**
 * Set the value of Impl2
 * @param newVar the new value of Impl2
 */
public void setImpl2 (Observer newVar) {
  Impl2 = newVar;
}

/**
 * Get the value of Impl2
 * @return the value of Impl2
 */
public Observer getImpl2 () {
  return Impl2;
}

//
// Other methods
//

/**
 * @return       int
 * @param        argc
```

```
     * @param        argv
     */
    public int main(int argc, char argv)
    {
    }


    /**
     * @return       boolean
     * @param        mensajecorrecto
     * @param        tam
     */
    public boolean validacion_mensaje(String mensajecorrecto, int tam)
    {
    }


    /**
     * @param        msgsave
     */
    public void Guardar_Mensaje(String msgsave)
    {
    }


    /**
     * @return       String
     * @param        messagereturn
     */
    public String Devolver_Mensaje(String messagereturn)
    {
    }
}
```

TransactionLog.java

```
/**
 * Class TransactionLog
 */
public class TransactionLog {

  //
  // Fields
  //

  public Server Impl4;

  //
  // Constructors
  //
  public TransactionLog () { };

  //
  // Methods
  //


  //
  // Accessor methods
  //

  /**
   * Set the value of Impl4
   * @param newVar the new value of Impl4
   */
  public void setImpl4 (Server newVar) {
    Impl4 = newVar;
  }

  /**
   * Get the value of Impl4
   * @return the value of Impl4
   */
  public Server getImpl4 () {
    return Impl4;
  }

  //
  // Other methods
  //

  /**
   * @return       String
   * @param        file
   * @param        mensajes
   */
  public String Guardar(String file, String mensajes)
  {
  }
}
```

Conclusión

Los diagramas UML nos permiten agilizar los procesos de documentación de Software que a su vez permiten una mayor robustez y abstracción del mundo real con nuestra aplicación, no obstante, es propio el considerar que dichos elementos son pautas para la generación de un software altamente sostenible dado que los cambios implementados en los diseños se pueden replicar de manera sencilla y rápida a través de herramientas CASE como lo es Umbrello.

Asimismo, existen muchos diagramas diferentes dentro de este, más sin embargo los más prácticos son Generalmente estos tres procesos ya que facilitan la mayor parte del trabajo para la generación de otros sub-diagramas del mismo software y/o aplicación.