4TO CONGRESO SOBRE MEDIOS DE TRANSPORTE Y TECNOLOGÍAS ASOCIADAS

Simulación de Unidad de Visualización y Control De Vuelo Usando Patrones de Diseño

MONSERRAT, D. [12], CABALLINI, V. [12]

1. **Grupo de Simulación Dinámica del Vuelo, U.T.N. F.R.H. – Haedo – Buenos Aires, Argentina**
2. **Laboratorio de Simulación y Control de Vuelo, U.T.N. F.R.H. – Haedo - Buenos Aires, Argentina**

dmonserrat@frh.utn.edu.ar

RESUMEN

La unidad de Visualización y Control de Vuelo (CDU) es la unidad en cabina que controla y muestra información sobre el Sistema de Gestión de Vuelo (FMS) que es parte fundamental de la aviónica de una aeronave comercial. En un simulador de ingeniería como el que cuenta el Laboratorio de Simulación y Control de Vuelo (LSCV), las funciones de estos dispositivos implementan características adicionales a las de su contraparte reales. Dichas características distintivas, son motivadas porque muchos de los datos de la simulación de la aeronave deben ser presentados y modificados en la cabina de vuelo por quienes se encuentren realizando estudios, sean pilotos o técnicos. De esta manera podemos mencionar como ejemplo, que, a los fines de algún ensayo en vuelo, se necesita tanto conocer los niveles de combustible, como también modificarlos a voluntad. Esta necesidad suele deberse a extender el tiempo de vuelo o también a modificar la posición del centro de gravedad de la aeronave. Este requerimiento adicional exige también una dinámica de cambio controlada de manera de no introducir errores en el software y una flexibilidad en el mismo para adecuarlo fácilmente a los nuevos requerimientos.

En este trabajo presentamos modelos de implementación de dichas unidades, utilizando Patrones de Diseño de Software simplificando así el desarrollo del instrumento simulado, además de demostrar que lo hace más fácil de comprender y mantener agregando flexibilidad a dicho software.

DESARROLLO

El desarrollo de proyectos de software, en especial aquellos que son complejos o como en el caso de simuladores, que deben acompañar la vida de la aeronave y sus cambios evolutivos, tiene como objetivo el lograr un fácil mantenimiento, extensión y confiabilidad, y es justamente en vista de estos objetivos, que el uso de Patrones de Diseño (SDP) y la programación Orientada a Objetos son técnicas ya bien establecidas.

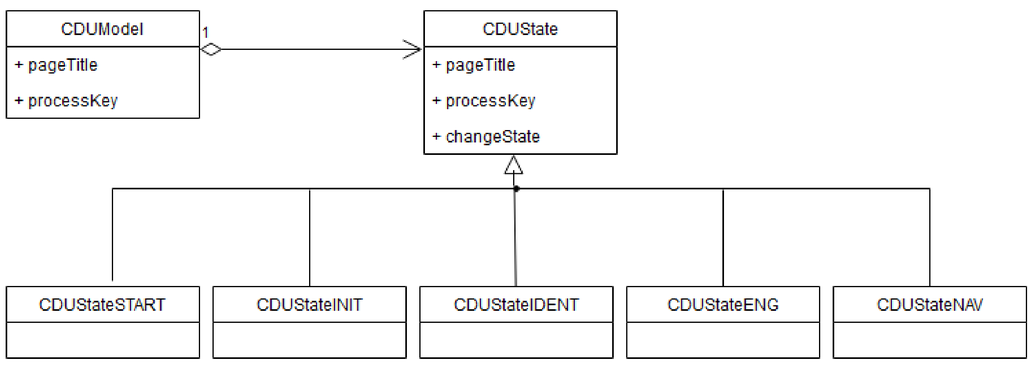


Figura 1: Simplificación de la Implementación del Patrón de Diseño State

El uso del trio de clases que componen el Model-View-Controller, más el uso de otros SDP del State que presentamos aquí en la implementación de la CDU Simulada, permitieron un desarrollo que es muy flexible ya que permite generar nuevos modelos y funcionalidades de un CDU de forma rápida. Ejemplo de esto es que la última funcionalidad incorporada consistente en una pantalla para ver la posición de los controles primarios tomó un tiempo total de desarrollo y pruebas operativas en el simulador de dos horas.

Otra ventaja de esta implementación es la generalización la cual permitió que ambas CDU aquí presentadas se encuentren en operación. La primera en la estación de instructor, con las características de una computadora personal estándar, es decir teclado, mouse y monitor. La segunda en el pedestal del simulador que es un monitor táctil de doce pulgadas. En ambos casos el software utilizado es el mismo.

```
/**
 * Class that models each state of the CDU
 * is the same as each page.
 */
class CDUState
{
protected:
  cduitm lines[NUMLINES][2];
  char pageTitle[50];
  char tmpLine[50];

  void setPageTitle(const char *v);

public:

  void toText(char *outBuffer, int line, int lr),

  char *getText(int line, int lr, bool editing, char *editBuf);

  char *getTitle(int line, int lr);

  char *getPageTitle();

  virtual void processKey(CDUModel *model, int
  virtual cduitm *getLine(int ndx, int lr) { r
  virtual bool validate(cduitm *itm, float val
};
```

Figura 2: Ejemplo de Declaración en C++

Para el desarrollo de software, la figura 1 muestra un diagrama sobre la estructura del SDP State. Este patrón de diseño se utiliza cuando un objeto cambia su comportamiento dependiendo del estado de este. En el caso de la CDU se presentó como una solución práctica ya que la interfaz es fija, es decir botones multifunción que dependen en la página en que se encuentre el operador para dar sentido a estos botones. Con el fin de encapsular el cambio de comportamiento de las diferentes páginas se ha utilizado el SDP State que permite que, en vez de manejar todos estos estados en un mismo bloque de código, se crea un objeto por cada estado para manejar el comportamiento.

```
CDUStateINIT::CDUStateINIT()
{
  this->setPageTitle("INIT/REF     INDEX");

  lines[0][0].set(0,cduitm::TXT, "", NULL, "<IDENT"      ,cduitm::PROMPT);
  lines[1][0].set(0,cduitm::TXT, "", NULL, "<POS"        ,cduitm::PROMPT);
  lines[2][0].set(0,cduitm::TXT, "", NULL, "<PERF"       ,cduitm::PROMPT);
  lines[3][0].set(0,cduitm::TXT, "", NULL, "<THURST LIM",cduitm::PROMPT);
  lines[4][0].set(0,cduitm::TXT, "", NULL, "<TAKEOFF"    ,cduitm::PROMPT);
  lines[5][0].set(0,cduitm::TXT, "", NULL, "<APPROACH"   ,cduitm::PROMPT);

  lines[0][1].set(0,cduitm::TXT, "", NULL, "NAV DATA>"   ,cduitm::PROMPT);
  lines[1][1].set(0,cduitm::TXT, "", NULL, "ENG>"        ,cduitm::PROMPT);
  lines[2][1].set(0,cduitm::TXT, "", NULL, ""            ,cduitm::PROMPT);
  lines[3][1].set(0,cduitm::TXT, "", NULL, ""            ,cduitm::PROMPT);
  lines[4][1].set(0,cduitm::TXT, "", NULL, "MAINT>"      ,cduitm::PROMPT);
  lines[5][1].set(0,cduitm::TXT, "", NULL, ""            ,cduitm::PROMPT);
}
```

Figura 3: Ejemplo de Implementación de un estado

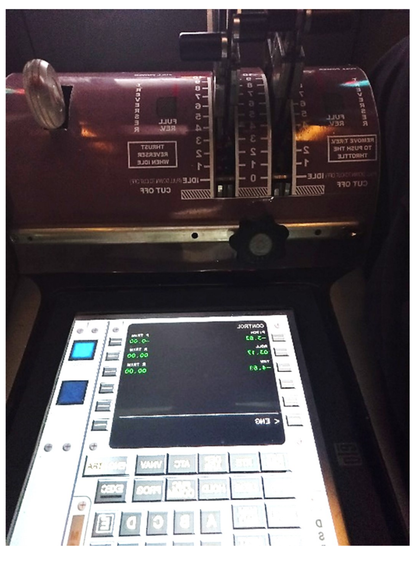
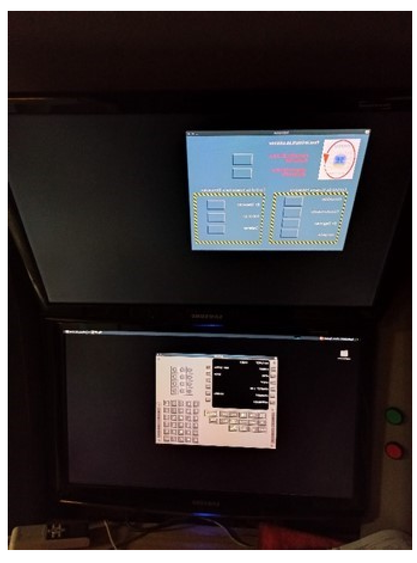


Figura 4: Fotografías de Modelo de Pedestal

Conclusiones

Como conclusión el desarrollo de las CDU utiliza exitosamente los SDP en sus diferentes niveles, es decir, tanto en alto nivel como son los componentes de la arquitectura como así también en los niveles inferiores para implementar conjuntos de trabajo lógico. El uso de State ha resultado en software que es más fácil de probar a nivel componente, por lo cual facilita una mayor calidad y confianza, sobre todo en estos frameworks utilizados para simulación de problemas de ingeniería.



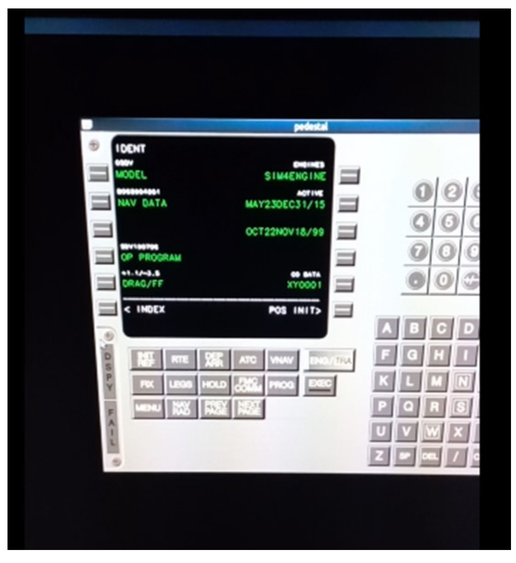


Figura 5: Fotografías Puesto de Instructor