Trabajo Final *Técnicas y Herramientas 2018*

Maestría en Ingeniería de Software

Facultad de Informática - UNLP

- **Ciochini, Lautaro**
- **Ludemann, Mauricio**
- **Ortu, Agustín**

PELLETLAND

Casos de uso:

- **Ingreso de un batch al sistema de acopio**
  El sistema de acopio recibe un batch, y debe darlo de alta, registrando el peso, el costo, la fecha, la calidad inicial, el deterioro estimado y el contenedor asignado para su almacenamiento.
- **Armado de un producto**
  El sistema debe intentar armar el producto solicitado, con los silos que posean las calidades requeridas por dicho producto.
- **Cálculo del deterioro de un batch**
  El sistema debe calcular el deterioro de un batch, el cual afecta su calidad, para una fecha solicitada. De acuerdo a la calidad inicial y al deterioro estimado, el sistema debe calcular la calidad resultante a cierta fecha.
- **Obtención del stock de materias primas**
  El sistema conoce su stock por calidad de materias primas, al momento de consultarse, o a una fecha determinada en el futuro.
- **Proyección de cantidad de producto que se puede realizar con la materia prima actual**
  El sistema proyecta la cantidad de cierto producto que puede realizar, dada una fecha. Para ello, tiene en cuenta el stock total y el deterioro de cada batch.

Diseño de la solución:

Para modelar el sistema de acopio completo, respondiendo a los requerimientos expuestos en la consigna, se plantearon los siguientes objetos con sus responsabilidades:

- **StorageSystem**: Representa el sistema de acopio; es el que gestiona los silos y la entrada y salida de materia prima; también es la clase que va a implementar los requerimientos de alto nivel, es decir, armar los productos, conocer el stock actual (y discriminado por calidad), así como proyectar la producción y el stock para una fecha dada.

- **Container**: Representa un silo, y son los encargados de almacenar los diversos lotes (objetos instancia de **Batch**) que posee la empresa. Tiene una capacidad limitada y posee la responsabilidad de brindar información acerca de la materia prima que tiene almacenada por calidad, tanto en peso absoluto como en porcentual. Esta información es utilizada por el StorageSystem para saber si puede elaborar un cierto producto con la materia prima de dicho Container. También es el encargado de administrar los registros históricos de entrada y salida de pellets.

- **Batch**: Es el lote de pellets, y se registra al momento del ingreso al sistema de acopio. Posee características como el peso, el costo, la calidad inicial, la fecha de incorporación y las fechas de deterioro, que se asignan al momento del ingreso del lote. Dichas fechas de deterioro se modelan con objetos **QualityDeterioration**.
  El **Batch** es responsable de conocer su calidad en una fecha dada, en base al deterioro percibido para esa fecha.
  En el batch también se pueden registrar incidentes, los cuales aceleran su deterioro. Los incidentes son modelados con objetos **Incident**.

- **QualityDeterioration**: Posee una cantidad de semanas, y la calidad a la que pasaría el **Batch**. Cada **Batch** estará relacionado a una colección de **QualityDeterioration**, representando cada transición de una calidad a otra. La cantidad de semanas es acumulativa, por ej:
  Si el deterioro de calidad *Alta* a *Media* dura 5 semanas, y de *Media* a *Baja* dura 3, entonces ese batch va a tener una colección con dos **QualityDeterioration**:
  - ➔ Un elemento que tenga 5 semanas y calidad Media,
  - ➔ Otro elemento con 8 semanas y calidad Baja.

- **Incident**: Modela un incidente que puede afectar a un **Batch** determinado, y posee una descripción, y la cantidad de semanas que adelanta el proceso de deterioro del **Batch**.

- **QualityType**: Es un objeto que modela un tipo de calidad, cada instancia tiene el nombre correspondiente. Estos objetos también saben compararse entre sí, de modo que distintas instancias de **QualityType** con el mismo nombre serán consideradas iguales.

- **QualityRule**: Representa las reglas de calidad con las que se conforman las diferentes líneas de productos que produce la empresa. Cada **QualityRule** se compone con los mínimos y máximos porcentajes admitidos para una calidad determinada. Los productos se componen de una colección de estas reglas, y el sistema se encarga de verificar que es capaz de satisfacerlas todas, de este modo puede determinar si es capaz de producir o no un determinado producto. Los objetos **QualityRule** exponen un método que permite verificar si es posible satisfacer la regla o no, ante un determinado nivel de calidad.

- **Product**: Modelan las distintas líneas de producto que puede producir la empresa; tienen un nombre y una colección de **QualityRule** que determinan los requisitos para poder armar el producto. Para instanciar un nuevo producto, solamente se deben crear las reglas necesarias, y colocarle un nombre nuevo. De esta manera, se pueden crear distintas configuraciones de nuevos productos, o modificar los ya existentes cuando se requiera.

- **BatchRecord**: Representa un registro histórico de los **Batch**. Los **Container** administran una colección de **BatchRecord**, creando instancias a medida que ingresan y egresan pellets del mismo. Existen dos tipos distintos de **BatchRecord**: **StorageRecord** y **WithdrawRecord**, ambos subclases de **BatchRecord**, modelando en el primer caso registros de alta o ingreso de pellet, y salida o egreso de pellets en el segundo. **BatchRecord** es una clase abstracta pero que define la interfaz y atributos. Las subclases son responsables de implementar un método que devuelve una descripción acorde a cada caso particular.

Aclaraciones:

- Como pauta de diseño, se modeló el registro histórico de pellets, como responsabilidad del contenedor. Esto se debe a que el contenedor es quien almacena y mezcla los diferentes batches, y que va a perdurar durante todo el tiempo de vida del sistema. En cambio, un batch se da de baja y desaparece del contenedor.
- Utilizamos Pharo Smalltalk 7
- Convención de nombramiento de variables: CamelCase en inglés. lowerCamelCase para atributos y métodos, y UpperCamelCase para clases.

Code Critics

El Critic Browser arrojó 7 críticas, todas correspondientes a la categoría “Methods implemented but not send”, es decir, métodos que hemos implementados pero no se han usado en la solución. En los 7 casos ocurre que son métodos **accessors**, es decir, para obtener o asignar valores de las variables de instancia de los objetos.
El motivo por el cual decidimos ignorar las críticas y dejar los métodos es porque estos permiten modificar o consultar el estado de los objetos, algo que puede llegar a ser útil en diversas situaciones.
Por ejemplo, entre los requerimientos se establece que la empresa debe poder modificar, en cualquier momento, la conformación de las distintas líneas de producto; para hacer esto, es necesario modificar las variables de instancia de objetos **QualityRule**, para lo cual es necesario utilizar **métodos accessors**; estos métodos son **#minValue, #minValue:, #maxValue, #maxValue:**, los cuales aparecen listados en las críticas; de eliminarlos, perderíamos está flexibilidad
Otro método que aparece como crítica es el **#cost** de la clase **Batch**; si bien, el costo no lo hemos usado para implementar la solución, en los requerimientos se menciona que al ingresar los lotes de pellet se registra el costo del mismo, y por este motivo modelamos la variable de instancia y su correspondiente accessor; de no hacerlo, el sistema no permitiría consultar un valor que fue ingresado por los operadores.