Capítulo 1

Document Planning

En la arquitectura presentada en el capítulo **??** mencionamos que el *document planner* es el responsable de decidir que información comunicar (*determinación de contenido*) y como deberá estar estructurada esta información en el texto final (*estructuración de documento*). El document planner será el encargado de que el documento final contenga toda la información requerida por el usuario y que la misma se encuentre estructurada de una forma razonablemente coherente. El resultado de esta etapa será un *document plan* en el cual se especifica qué contenido debe ser incluido en el texto final y de que forma debe estar estructurado.

A continuación describiremos brevemente la entrada y salida de nuestro *document planner*, definiremos como modelar los elementos informativos (estos serán elementos de nuestro *document plan*) y finalmente describiremos las tareas de *determinación de contenido* y *estructuración de documento*.

1.1. Entrada y salida del document planner

Como el document planner es el primer módulo del pipeline, la entrada del document planner será la misma que la entrada de nuestro sistema. Reiter y Dale [RD00] generalizan la entrada de un sistema de NLG como una cuádrupla compuestas por los siguientes componentes:

Fuente de conocimiento: Se refiere a las bases de datos e información del dominio de aplicación que nos proporcionará el contenido de la información que los textos generados deberán contener. En nuestro caso la fuente de conocimiento estará compuesta por la especificación, las designaciones de la misma y las clases y casos de prueba generados.

Objetivo comunicacional: Especifica el propósito que debe cumplir el sistema. En general esta compuesto por un "tipo de objetivo" y un parámetro. En este trabajo tendremos solo un tipo de objetivo comunicacional: *Describir(x)*,

dónde el parámetro x será un conjunto de identificadores de las clases de prueba que desean describirse.

Modelo de usuario: Provee información acerca del usuario (nivel de experiencia, preferencias, etc.). En nuestro caso el sistema se comportará de la misma forma independientemente del usuario, por lo que no tendremos en cuenta información del mismo.

Historial de discurso: Consta de información sobre interacciones previas entre el usuario y el sistema. Este historial puede servir para algunos sistemas interactivos donde las interacciones previas con el usuario pueden resultar de utilidad.

La salida del document planner será un document plan. En nuestra arquitectura el document plan está estructurado como un árbol, donde las hojas representan el contenido y los nodos internos especifican información estructural, por ejemplo sobre como debe agruparse la el contenido a comunicar. En la sección 1.4 desarrollaremos este tema mas en detalle.

1.2. Representación del dominio

En los sistemas de NLG el texto generado se utiliza principalmente para transmitir información. Esta información será expresada generalmente en frases y palabras, pero estas frases y palabras no son en si mismo la información; la información subyace estos constructores lingüísticos y es "llevada" por ellos. Nos deberemos concentrar entonces en como representar este conocimiento y como "mapear" estas estructuras a una representación semántica.

El *corpus de descripciones* (apéndice **??**) resulta una buena fuente para estudiar y definir como deberemos representar la información o *mensajes*[1] a comunicar. Podemos ver en todos los ejemplos del *corpus* una relación directa entre la información a comunicar y las expresiones Z pertenecientes al dominio de aplicación, donde podemos observar que cada linea de una descripción se corresponde perfectamente con la verbalización de la expresión en lenguaje Z correspondiente. Veamos por ejemplo las siguientes líneas de la descripción de LookUp_SP_1 introducida anteriormente (pág. **??**):

1. *"El símbolo a buscar pertenece a los símbolos cargados en la tabla de símbolos."*
2. *"El símbolo a buscar es el único elemento del conjunto formado por los símbolos cargados en la tabla de símbolos."*

[1] Reiter y Dale llaman *mensajes* a elementos informativos que conceptualizan la información que queremos comunicar. Estos están compuesto en sí mismos por elementos del dominio de aplicación.

en las que podemos observar que la información de cada linea de la descripción se encuentra perfectamente caracterizada por la expresión Z que describe, enumeradas a continuación:

1. $s? \in \text{dom } st$
2. $\text{dom } st = \{s?\}$

Debido a esta correlación entre las expresiones de las clases de prueba y la información a generar, la verbalización de las mismas, podemos establecer un único tipo de *mensaje* para nuestro sistema: *VerbalizacionExpresion*. Este representa, como su nombre lo indica, una verbalización de una expresión Z. En la figura 1.1 podemos ver como quedarían definidos los mensajes para las dos lineas de la descripción de *LookUp_SP_1* vista anteriormente.

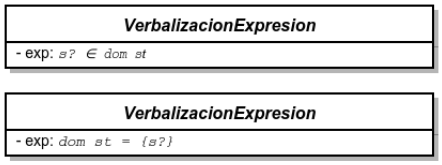

Figura 1.1: Mensajes a comunicar para el ejemplo de la figura **??**.

1.3. Determinación del contenido

La determinación del contenido es el nombre que se le da a la tarea de decidir y obtener la información que se debe comunicar en un texto. Este proceso generalmente involucra una o más tareas de selección, resumen y razonamiento con los datos de entrada. El *proceso de selección* recopilará un subconjunto de la información de entrada para luego poder ser comunicada al usuario. El objetivo del mismo será el de proveer la información relevante requerida por el mismo. La tarea de *resumen* es necesaria cuando los datos de entrada son muy "granulados" para ser comunicados directamente o si la información relevante consiste alguna generalización o abstracción de los mismos, que no es el caso de nuestro sistema donde las expresiones de Z con las que trabajaremos contienen exactamente la información que se desea comunicar. Por último el *razonamiento con los datos* resulta un caso general de las dos anteriores. Finalmente, una vez seleccionada y procesada la información necesaria será esta etapa la encargada de construir los mensajes introducidos en la sección anterior, que luego formarán parte de nuestro *document plan*.

Para nuestro trabajo, la tarea de selección se resume en la búsqueda y filtrado de las clases de prueba indicadas por el usuario dentro de todo el conjunto de clases de prueba que forma parte de la entrada de nuestro sistema. Por ejemplo, si deseamos generar una descripción para la clase de prueba *LookUp_SP_1* de la figura **??**, la misión de de esta tarea será la de identificar

y seleccionar la clase de prueba *LookUp_SP_1* entre todas las clases de pruebas que forman parte de los datos de entrada de nuestro sistema de NLG.

Luego de la selección, nuestro sistema deberá procesar los datos de entrada con el fin de obtener mejores descripciones. Hemos observado[2] que trabajando ciertas expresiones en las clases de prueba podemos mejorar considerablemente los textos generados. Veamos por ejemplo la figura 1.2 donde se muestra una clase de prueba generada con Fastest en la que se pueden ver dos problemas que abordaremos en esta etapa.

Update_SP_4

$$st : SYM \pfun VAL$$
$$s? : SYM$$
$$v? : VAL$$

$$st \neq \{\}$$
$$\{s? \mapsto v?\} \neq \{\}$$
$$\mathrm{dom}\, st = \mathrm{dom}\{s? \mapsto v?\}$$

Figura 1.2: Clase de prueba para operación Update (pág. **??**).

Podemos observar que la siguiente expresión del ejemplo anterior:

$$\{s? \mapsto v?\} \neq \{\}$$

no aporta información importante para el usuario, de hecho esta expresión no agrega ninguna restricción para el caso de prueba ya que será siempre verdadera. De no filtrar esta expresión tempranamente, terminaríamos como resultado un texto generando parecido al siguiente:

"el conjunto formado por el par de el símbolo a actualizar y el nuevo valor, es distinto al conjunto vacío"

que además de resultar algo difícil de interpretar, no aporta nada al objetivo comunicacional.

Por otro lado, en un primer intento por describir automáticamente la expresión:

$$\mathrm{dom}\, st = \mathrm{dom}\{s? \mapsto v?\}$$

podríamos describirla como[3]:

[2] Las observaciones son en base a clases de prueba generados utilizando Fastest 1.6

[3] Esta descripción sería la generada utilizando el sistema de reglas propuesto en el capítulo **??** si no trabajamos la expresión en una etapa previa.

"el conjunto de símbolos cargados en la tabla es igual a el dominio del par formado por el símbolo a actualiza y el nuevo valor"

Es posible simplificar notablemente esta descripción si antes trabajamos la expresión anterior, que resulta equivalente a:

$$\mathrm{dom}\, st = \{s?\}$$

que luego podríamos describir como:

"el símbolo a actualizar es el único elemento en la tabla de símbolos cargados"

En conclusión, el procesamiento que nos proponemos a realizar en esta etapa tendrá dos objetivos:

1. Eliminar tautologías de las expresiones que forman parte de las clases de prueba seleccionadas.
2. Realizar algunas simplificaciones o reducciones triviales.

Una vez seleccionada y procesada la información deberemos construir los *mensajes* (*VerbalizacionExpresion*) que luego formarán parte del *document plan* desarrollado en el siguiente capítulo.

1.4. Estructuración del documento

Como dijimos antes, el texto generado no podrá ser una colección al azar de frases y palabras. Deberá tener coherencia y poseer una estructura que le permita al lector interpretar con facilidad el contenido del mismo. Necesitaremos considerar como organizar y estructurar la información que debemos comunicar con el fin de producir un texto razonablemente fácil de leer y comprender.

En esta tarea nos concentraremos en construir una estructura que contenga los *mensajes* seleccionados en la etapa de *determinación de contenido*; estableciendo el agrupamiento y ordenamiento de los mismos. Esta estructura deberá caracterizar la disposición de los elementos pertenecientes a los textos recopilados en el *corpus*.

Tomando el corpus de descripciones como una especificación de los documentos que debemos generar podemos observar que estos documentos poseen una estructura bastante simple y rígida a la vez. Estos documentos deben estar formados por una secuencia de descripciones para las clases de prueba indicadas por el usuario, ordenadas alfabéticamente según el nombre de la clase de prueba. A su vez, cada una de estas descripciones deberá agrupar las verbalizaciones de las expresiones seleccionadas en la etapa de *determinación de contenido*, ordenadas de la misma forma en la que aparecen en el

esquema de la clase de prueba en cuestión. En la figura 1.3 podemos observar una representación abstracta de la estructura propuesta para modelar el documento.

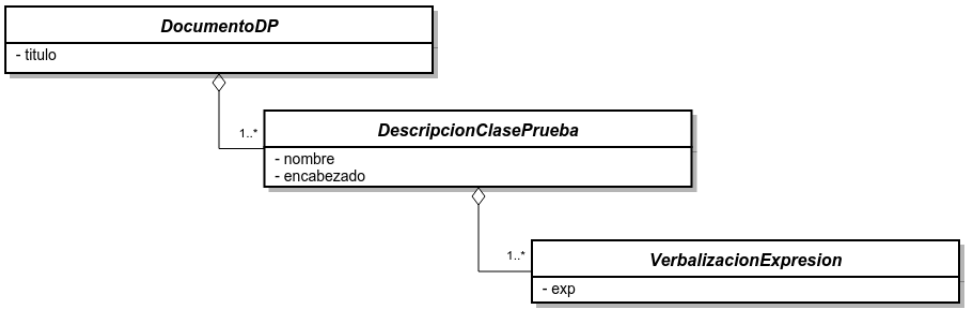

Figura 1.3: Document plan.

Llamaremos *DocumentoDP* a la raíz de nuestro *document plan*, *DocumentoDP* contendrá a su vez una lista ordenada de las descripciones de las clases de prueba (*DescripcionClasePrueba*) que debemos incluir en el texto final. El elemento *DescripcionClasePrueba* representa el texto a generar para una clase de prueba (por ejemplo el texto de la figura **??**) y tendremos uno de estos elementos por cada clase de prueba indicada por el usuario. Finalmente los mensajes seleccionados en la etapa anterior formarán se encontrarán agrupados en la *DescripcionClasePrueba* correspondiente. Podemos ver en la figura 1.4 un ejemplo del document plan para la descripción de la clase de prueba *LookUp_SP_1* introducida anteriormente.

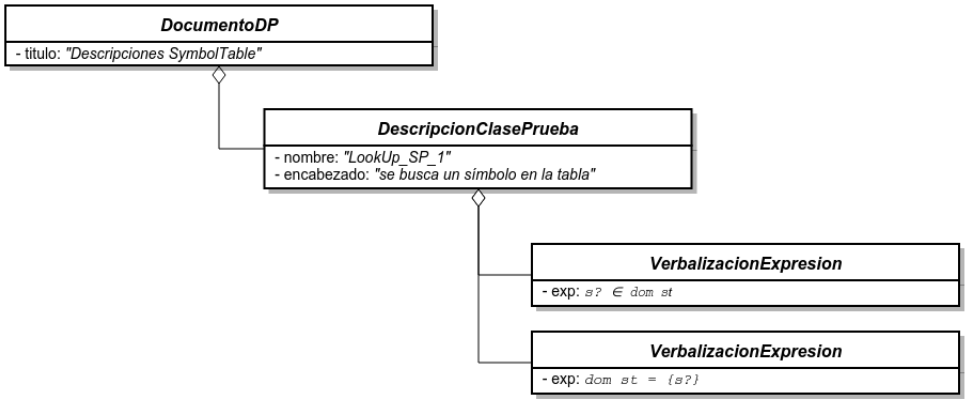

Figura 1.4: Document plan correspondiente al texto de la figura **??**.

Bibliografía

[CAF+11] Maximiliano Cristí, Pablo Albertengo, Claudia Frydman, Brian Pluss, and Pablo Rodriguez Monetti. Applying the test template framework to aerospace software. In *Proceedings of the 2011 IEEE 34th Software Engineering Workshop*, SEW '11, pages 128–137, Washington, DC, USA, 2011. IEEE Computer Society.

[CM09] Maximiliano Cristiá and Pablo Rodríguez Monetti. Implementing and applying the stocks-carrington framework for model-based testing. In *Proceedings of the 11th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, ICFEM '09, pages 167–185, Berlin, Heidelberg, 2009. Springer-Verlag.

[CP10] Maximiliano Cristiá and Brian Plüss. Generating natural language descriptions of z test cases. In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 173–177, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[RD00] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000.

[SC96] Phil Stocks and David Carrington. A framework for specification-based testing. *IEEE Trans. Softw. Eng.*, 22(11):777–793, November 1996.

[Spi92] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1992.