EL DESARROLLO DE SOFTWARE EN EL PROYECTO DE AUTOMATIZACION DEL SISTEMA DE BIBLIOTECA DE LA PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE

Rodrigo Cifuentes S.(*)
Hugo Cisternas S..(*)
E. Alejandro Burgos M.(*)

RESUMEN

Se describen a continuación las características de una aplicación bibliotecológica y las peculiaridades de una implementación en particular. Además, se explicitan los criterios de diseño y programación de software, elegidos por el Departamento de Computación del Sistema de Bibliotecas de la Pontificia Universidad Católica de Chile, para el desarrollo del Sistema de Información Bibliográfica (SIBUC), en microcomputadores multiprocesador, multiusuario. Los criterios utilizados son: desarrollo por prototipo e interfaz grata al ususario, modularidad funcional, independencia entre la lógica de la aplicación y la lógica de la implementación, y uso de autómatas. Se destaca la importancia de la interacción con el usuario, el uso de técnicas para implementación modular y portable y el uso de autómatas, para mejorar tanto la adecuación del diseño a las espectativas del usuario, como la productividad computacional y la mantención del sistema.

1. INTRODUCCION

El sistema de Bibliotecas de la Pontificia Universidad Católica de Chile comprende 10 Bibliotecas repartidas en 4 Campus en la Región Metropolitana. Cuenta con una colección de 900.000 volúmenes y atiende a una población de usuarios potenciales que alcanza a 12.000 lectores. La automatización de los procesos bibliotecarios (1), tuvo sus orígenes en el proyecto "Catalogación Automatizada Centralizada", presentado a la O.E.A. en el bienio 1978-1979. Este fue replanteado para el bienio 1980-1981, conviertiéndose en un proyecto global, que culminará con la automatización de los procesos de catalogación, circulación, adquisiciones, control de publicaciones periódicas y con la formación de un catálogo en línea. Cada uno de estos procesos bibliotécnicos tiene sistemas computacionales asociados, que en su conjunto conforman lo que denominamos Sistema de Información Bibliográfica de la Pontificia Universidad Católica (SIBUC).

El proyecto tuvo un presupuesto inicial muy pequeño, lo que junto con la intención de crear un paquete que estuviera al alcance de las Bibliotecas Latinoamericanas, indujo a optar por microcomputadores. Las especificaciones técnicas que se impusieron a este equipamiento fueron:

(*) Pontificia Universidad Católica de Chile
Sistema de Bibliotecas

- Standard: deberían tener procesador Z80 (standard de la época).
- Bus S-100 que permitiera adquirir hardware de varios proveedores.
- Sistema Operativo CP/M o compatible.
- Posibilidad de administrar múltiples usuarios.
- Soporte de discos duros de alta capacidad.

Para llevar a cabo este proyecto, se creó un Departamento de Computación dentro del Sistema de Bibliotecas, el que ha implementado, a la fecha, los sistemas asociados a los procesos de catalogación y circulación.

El proceso bibliotecario de catalogación comprende la descripción, tanto de contenido como física, de los títulos que conforman el fondo bibliográfico de una Biblioteca. Nuestro sistema automatizado de catalogación está formado por los siguientes subsistemas:

- Entrada de datos permite crear registros bibliográficos y poblar la base de datos bibliográficos.
- Supervisión permite controlar y corregir los registros bibliográficos.
- Consultas a la base de datos bibliográficos: permite consultar un título por distintas claves de acceso.
- Inventario: permite llevar un control de existencias de los volúmenes y copias
- Producción de materiales: permite imprimir fichas catalográficas y otros materiales que acompañan al libro.

El proceso bibliotecario de circulación comprende las funciones necesarias para el préstamo, devolución y renovación del material bibliográfico. Nuestro sistema automatizado de circulación maneja tanto las transacciones como la creación, mantención y control de bases de datos de libros y usuarios, y está formado por los siguientes subsistemas:

- Transacciones de préstamo, devolución y renovación.
- Control de usuario: permite construir y mantener la base de datos de usuarios.
- Morosos: permite el control de los usuarios morosos.
- Libros: permite la administración de los datos de circulación de los libros.
- Reserva: permite avisar al sistema las reservas pendientes sobre un título.
- Inventario: permite controlar las existencias de la colección que esta circulando.

Ambos sistemas se han implementado sobre microcomputadores IMS multiprocesador/multiusuario, usando TurboDOS, un sistema operativo (SO) multiusuario compatible con CP/M, y el lenguaje dea programación Pascal.

La finalidad de este artículo es dar a conocer la problemática enfrentada y las técnicas utilizadas en el desarrollo de software del proyecto SIBUC. Los alcances, historia y perspectivas del proyecto están descritos en las referencias 1 y 2 de la bibliografía y exceden al ámbito técnico de este trabajo.

2. ANTECEDENTES

En el desarrollo de sistemas computacionales se trabaja simultáneamente en dos niveles:

- La aplicación: que origina la lógica propia del problema que se intenta resolver.
- La implementación: que pone las herramientas computacionales al servicio de la aplicación.

En muchos casos estos niveles se confunden, pero el avance de las técnicas de desarrollo de software ha hecho evidente que es necesario separar estos dos aspectos en forma clara, para lograr una baja dependencia de la aplicaión con respecto a las herramientas disponibles en una instalación específica, lo que sin lugar a dudas permite una mayor portabilidad.

Los sistemas desarrollados para el SIBUC, han presentado características complejas en ambos niveles, las que son detalladas en los siguientes párrafos.

3. LA APLICACION

En el nivel de la aplicación, las complejidades son básicamente de dos tipos:

- La diversidad de sistemas y sus interrelaciones.
- La semántica y sintaxis de la información bibliográfica.

3.1. Diversidad de Sistemas y sus Interrelaciones

Las funciones que son factibles de automatizar en una biblioteca, van desde la adquisición de un libro hasta su salida en manos de un lector.

De este modo se tienen, entre otras, las funciones de:

- Adquisición
- Catalogación.
- Circulación.
- Control de Publicaciones Periódicas.
- Inventario y control de existencia.
- Recuperación de información : - orientada al Bibliotecario.
  - orientada al Lector.
- Desarrollo de la colección.
- Administración.

Cada una de ellas origina sistemas y subsistemas, que deben ser analizados y desarrollados de acuerdo a sus propias características y requerimientos. Estas funciones no son independientes una de la otra, más bien tienen múltiples relaciones, que se traducen en flujos de datos entre ellas. Catalogación, por ejemplo, alimenta a circulación y al sistema de recuperación de información. Este último, a su vez, retroalimenta a catalogación al permitir controlar si un libro es copia de otro existente o un nuevo título. Circulación puede, también, entregar información necesaria para el sistema de desarrollo de colecciones.

Esta malla de interrelaciones debe ser considerada al momento de difinir las estructuras de datos, ya que impacta directamente en el diseño de cada subsistema.

3.2. La semántica y sintaxis de la información bibliográfica

Todo libro en una Biblioteca debe ser representado en un catálogo (manual o automatizado), para que pueda ser ubicado por los usuarios. La representación se hace mediante un registro bibliográfico (p. ej. una ficha catalográfica de un catálogo manual), que incluya aquellos datos sobre el libro que lo representan e identifican. Para ello, se usan algunos datos, (elementos descriptivos) extraídos directamente del libro (título, pie de imprenta, etc.) y otros que son producto de una investigación por parte de un bibliotecario especializado, que permiten identificar su contenido.

Para poder producir descripciones consistentes, inteligibles y de calidad uniforme, se ha elaborado normas, tanto a nivel de contenido (semántica) como a nivel de forma (sintaxis), que son usadas internacionalmente. Ejemplo de normas semánticas son los vocabularios controlados (listas de términos, frases y nombres) y las reglas para elección de los elementos descriptivos. Ejemplo de normas, sintácticas son las que reglamentan la puntuación, el orden y disposición de los elementos descriptivos, etc. estas normas, al ser aceptadas internacionalmente, permiten intercambiar registros bibliográficos y comprender estas descripciones independientemente de su origen.

3.2.1. Semántica de la información bibliográfica

Por semántica de la información bibliográfica, entendemos el contenido de la descripción y su significado, agrupado en tres tipos de elementos descriptivos (3), que permiten organizar los registros bibliográficos:

- Elementos de identificación del libro, que permiten distinguir un libro de otro, por ejemplo: título, fecha de publicación, etc.
- Elementos que relacionan un libro con otro (s), por ejemplo: autor, distintas ediciones del mismo título, etc.
- Elementos que caracterizan el contenido intelectual, por ejemplo: encabezamientos de materia y descriptores.

Debido a que el libro es una producción intelectual, la diversidad y variabilidad de posibles contenidos es inmensa, y va desde un cuento o novela ("Raquela") a una memoria técnica de gran especificidad ("Estudio de la polimerización de bisfenol. A con tiofosgeno en condiciones de transferencia de fase"). El registro bibliográfico que lo representa, en un catálogo manual o automatizado, debe considerar esta diversidad y variabilidad, y representarla fielmente.

El aspecto semántico esta regido por normas que reglamentan la descripción, como son las Anglo American Cataloging Rules, segunda edición (4) (AACR2) y los vocabularios controlados, que normalizan la descripción de las materias de que trata un libro, eliminando sinónimos y ambiguedades. Por ejemplo: Library of Congress Subject Headings (5) (LCSH), Thesaurus of Engineering and Scientific Terms (6), etc.

Las características semánticas de la información bibliográfica se reflejan en la aplicación computacional, introduciendo las siguientes problemáticas.

- *Representación.* La variabilidad de la información se traduce en un modelo sofisticado, que debe garantizar la conformidad con las reglas y conveciones establecidas. Así por ejemplo, podríamos considerar que cada libro tiene un

título. Sin embargo, el título puede tener variantes a través de diferentes publicaciones ("El mio Cid", "El Cid Campeador", etc.) o pueden existir títulos paralelos en la obra, subtítulos, varios títulos en un mismo libro (una antología), etc. Esto ocurre para cualquier elemento descriptivo de un libro. La estructura de datos debe dar cabida a todas las posibles variantes para representar adecuadamente el modelo.

- *Dispersión.* La variabilidad genera, además, una dispersión sobre los posibles puntos de acceso a la información, lo que se traduce en problemas de indización. Un caso típico se da con los autores: Un libro puede ser escrito por uno o varios autores, o puede ser el informe de un departamento técnico, el acta de un congreso o simplemente no tener autor claramente identificado. Para efectos de recuparación de información, la construcción de índices deberá considerar todos estos casos.

- *Relaciones.* Las relaciones entre distintos libros deben reflejarse en la descripción, y por lo tanto, en el sistema que las maneja. Las relaciones son de dos tipos fundamentales: relaciones de orden parcial, como por ejemplo., las relaciones cronológicas ("ser publicado antes, o el mismo año que); o relaciones de equivalencias, como por ejemplo, las obras de un mismo autor, ("ser escrito por el mismo autor"). Puede haber, además, combinaciones de estas (una nueva edición de un título, en la cual se incorpora un nuevo autor), las que definen intersecciones y/o sub-recorridos.

Las relaciones y sus combinaciones tienen un impacto directo sobre el diseño de una base de datos. Los modelos jerárquicos tradicionales permiten implementar razonablemente las relaciones de equivalencia, pero no son adecuados para resolver las relaciones de orden. Un modelo relacional soportaría, teóricamente, ambos tipos de relacines, pero su implementación se tornaría sumamente compleja.

3.2.2. Sintaxis de la información bibliográfica

Por sintaxis de la información bibliográfica entendemos la forma y el orden en que ésta debe disponerse en un registro bibliográfico, para ser comunicada sin ambiguedades.

Además debe considerarse que para poder describir cualquier libro, se necesita tener un universo de elementos descriptivos bastante amplio, y que sólo un subconjunto de ellos es necesario para construir el registro bibliográfico de un libro dado. Entonces, la estructura sintáctica debe permitir distinguir la (s) ocurrencia (s) de un elemento descriptivo específico. Ahora bien, algunos elementos deben estar presentes obligatoriamente en toda descripción; otros son excluyentes entre sí, es decir, sólo uno de ellos puede existir en un registro; y otros son simplemente opcionales.

Las reglas AACR2 contienen indicaciones precisas de como disponer cada elemento, su estructura, su orden de precedencia y la puntuación que debe separarlos. Esta estructura sintácticas es a la vez flexible, para absorver la variabilidad de los contenidos, y rígidas, para uniformar y discriminar cada descripción. Tal estructura, aplicada al campo computacional, da origen a formatos de registros y a algoritmos de manipulación de información altamente complejos. Uno de estos formatos es MARC

(Machine Readable Cataloging (7)). Un excelente análisis en profundidad, respecto de normas y compatibilidad de formatos, se encuentra en Lancaster (8).

El formato MARC es una norma para la representación de información bibliográfica en forma legible por computador. Permite organizar el contenido del registro computacional de tal manera que, por un lado, cada elemento descriptivo pueda ser identificado sin ambiguedades y por otro, se pueda establecer la relación que existe entre ellos. El conjunto de elementos descriptivos definidos en el formato permite describir cualquier ente bibliográfico; sin embargo, cada registro contiene sólo aquellos elementos que le son útiles para una descripción específica.

En el MARC los elementos descriptivos estan agrupados en campos, los cuales son identificados mediante códigos de 3 caracteres numéricos. Los elementos componentes de un campo son llamados subcampos, los cuales se identifican a través de un código compuesto de un caracter hexadecimal, normalmente representado como "$", seguido de una letra minúscula o un número.

El formato MARC establece una estructura para el registro bibliográfico, que corresponde a un registro de largo variable con directorio, y es una implementación de la norma ANSI Z-39.2-1979 (9) y de la ISO 2709-1973 (10). Esta estructura permite organizar la descripción a dos niveles:

- Los campos son representados a través del directorio, que indica, para cada campo presente en el registro, su código, posición y largo.
- Los subcampos de un campo, que contienen la información propiamente tal, se encuentran a partir de la posición indicada en el directorio. Cada subcampo es precedido por su código, para poder ser identificado adecuadamente.

4. LA IMPLEMENTACION

El software del proyecto de automatización del SIBUC se ha desarrollado sobre microcomputadores IMS International, usando sistema operativo TurboDOS y el lenguaje de programación Pascal. Los microcomputadores IMS tienen una arquitectura de múltiples procesadores Z80 (*) (64Kb RAM c/u), conectados en red estrella sobre un bus S-100, organizados en forma "master-slaves", que comparten un pool de impresoras y 96 Mb de disco duro.

Para la implementación del proyecto, se cuenta con 3 microcomputadores que asumen las siguientes funciones:

- Equipo de desarrollo, cuenta con 4 procesadores esclavos, una impresora y disco duro de 80 Mbytes.
- Equipo de catalogación, cuenta con 6 procesadores esclavos, dos impresoras y disco duro de 80 Mbytes.
- Equipo de circulación, cuenta con 5 procesadores esclavos, uno con impresora local una impresora central y disco duro de 80 Mbytes.

Además, se cuenta con una unidad de cinta magnética de 1/2 pulgada, marca CIPHER, para respaldo de archivos grandes (más de 1 Mbyte), que puede ser usada en cualquiera de los microcomputadores.

(*) En la actualidad se estan sustituyendo los procesadores Z80 por Intel 80186 de 16 bits con 256 Kbytes de memoria RAM cada uno (de reciente aparición en el mercado), con el objeto de mejorar el rendimiento y capacidad.

Los microcomputadores no están conectados entre sí, ya que el software y el hardware de comunicación de alta velocidad no estaban disponibles hasta hace muy poco tiempo, y el costo de desarrollarlo localmente era muy alto. La comunicación a bajas velocidades, para transferencia de archivos, siendo fácil de implementar no se justifica, por cuanto ella requiere de un procesador esclavo dedicado en cada microcomputador siendo más barato, para nuestro volumen de intercambio de información, hacer los traspasos a través de diskettes o cinta magnética.

En este ambiente, se han enfrentado problemas tanto de hardware como de software.

- El hardware impone restricciones tecnológicas relativas a rendimiento, capacidad y expansión.
- El software de sistema tiene limitaciones de disponibilidad, documentación, confiabilidad, capacidad y rendimiento.

4.1. Restricciones impuestas por el hardware

En cuanto a rendimiento, los microprocesadores de 8 bits, y los controladores de dispositivos periféricos asociados, son limitados en velocidad. Esto tiene dos consecuencias en el desarrollo de software:

- Los algoritmos deben ser optimizados en rápidez. Por ejemplo, un programa de entrada de datos tiene que hacer una cantidad baja de operaciones entre la lectura de una digitación y la siguiente, para no dificultar al digitador. Esta situación es más crítica si se usa lenguajes interpretados, debido a su inherente lentitud.

- La Entrada/Salida (E/S) a memoria secundaria es lenta. Las rutinas de E/S de los lenguajes agregan un nivel m'as de lentitud a éstas, ya que estan construidas de manera de cubrir todos los posibles casos que un programa pudiera necesitar. Esto nos ha obligado a escribir nuestras propias rutinas de acceso a los archivos, llamando directamente al S0, lo que ha mejorado significativamente el tiempo de respuesta.

Referente a capacidad, hay nuevamente dos aspectos del problema

- Capacidad de direccionamiento a memoria principal limitada a 64Kb, que obliga a invertir una gran cantidad de tiempo en optimizaciones de espacio, manejo de overlays, construcción cuidadosa de paquetes de rutinas, etc.

- Capacidad de direccionamiento de la memoria secundaria. Si bien es cierto que los microcomputadores IMS, bajo sistema operativo TurboDOS soportan 16 discos de hasta 256 Mbytes cada uno, estos máximos se ven severamente restringidos por el uso de software compativle con CP/M (por ejemplo: compiladores), que impide direccionar más de 8 Mb en un archivo. Para sobrepasar esta limitación, es necesario usar técnicas sofisticadas como, por ejemplo, dividir (y administrar) el archivo en segmentos o programar en Assembler haciendo uso de las funciones específicas de Turbo DOS para direccionar archivos de hasta 134 Mbytes.

Estas restricciones hacen que el micro pueda crecer significativamente, para lo que se entiende habitualmente por microcomputadores, pero limitadamente para una aplicación de gran envergadura como es este proyecto.

4.2. Restricciones impuestas por el software de sistemas

Este software impone, a causa de varias limitaciones:

- Disponibilidad y documentación: al comenzar la parte computacional del proyecto (1981) había poco software disponible. Los manejadores de archivos estaban recién apareciendo en el mercado. Los lenguajes disponible seran pocos, la mayoría implementaciones de BASIC. Utilitarios para ordenamiento o comunicación de datos eran escasos. La documentación disponible era difícil de conseguir. Esto obligó a probar el software disponible y seguir investigando, paralelamente al desarrollo.

- Confiabilidad y capacidad: los paquetes con que se ha hecho el desarrollo no son totalmente estables. Los compiladores tienen problemas que se reflejan en mensajes confusos y localizados en lugares distantes del error real. Además, las funciones de E/S son limitadas, por ejemplo: el Pascal MT+ no provee record-lock para implementar aplicaciones multiusuario, por lo que hemos tenido que desarrollar estas funciones en assembler. Los manejadores de archivos son demasiado lentos, de baja capacidad o contiene fallas que imposibilitan su uso en aplicaciones sofisticadas. Por ejemplo. en 1981 se seleccionó el paquete MDBS I para ser usado en el proyecto. Si bien este manejador es bastante completo, bajo procesadores, Z80 es muy limitado en cuanto a capacidad de almacenamiento y, además, es monousuario. Esto nos obligó a desarrollar un manejador de archivos propio para el sistema de Circulación y, además, a sustituir en 1984 MDBS I por un paquete desarrollado internamente para Catalogación. Hemos debido escoger un conjunto pequeño de herramientas que reune las características de confiabilidad requeridas para el desarrollo, y conocer sus peculiaridades, para poder evitar o solucionar los problemas que presentan.

- Rendimiento: debido a las características anteriores, la productivilidad, usando estos paquetes, tiende a ser menor que la que se logra en otro medio ambiente. El tiempo invertido en segmentar un programa de 80Kb para que funcione en un ambiente de 64Kb es significativo. El tiempo invertido en tratar de hacer uso óptimo de un paquete de manejo de archivos, para que su tiempo de respuesta sea apropiado, es también importante.

5. CRITERIOS

Para el desarrollo de los sistemas del SIBUC, tanto en el nivel de la aplicación como en el de la implementación, se utilizaron cuatro criterios básicos para enfrentar el diseño y programación, que conforman un estilo fuertemente marcado por la participación e intercción con el usuario.

5.1. Desarrollo por prototipo e interfaz grata al usuario

Este método consiste en desarrollar el sistema a partir de la interfaz con el usuario. Tiene básicamente dos etapas: la primera es un análisis inicial de la aplicación, realizado en conjunto y sobre la base de los requerimientos especificados por el usuario. En base a esta información, se elabora una interfaz, que simula el futuro sistema.

En la segunda etapa, este prototipo es sometido a diferentes pruebas, de las cuales surge el diseño lógico del sistema en una forma natural. Esta metodología logra anticipar detalles en la lógica menor de la aplicación, que en otros enfoque sde desarrollo aparecen en la puesta en marcha o como fallas del sistema. Además, como resultado del uso del prototipo por parte del usuario, se logra una mayor adaptación del sistema a las necesidades de forma y contenido de la interfaz. con lo cual se hace grata al usuario.

La experiencia nos ha demostrado que para lograr una buena interfaz, deben usarse 5 elementos:

- *Menus*: el usuario elige una entre varias alternativas en lugar de recordar códigos nemotécnicos.

- *Mensajes comprensibles*: el usuario no debe interpretar códigos numéricos o nemotécnicos, para comprender los mensajes provenientes del programa.

- *Retroalimentación*: cada acción del usuario es seguida de una respuesta inmediata por parte del programa. Incluso los programas no interactivos, muestran información que permite saber en que etapa de proceso estan.

- *Consistencia*: las interfaces tienen características similares a través de todo el sistema, de manera que el ususario se acostumbra a una presentación uniforme de la información y aprende un único conjunto de caracteres de control para la entrada y edición de datos.

- *Niveles de ayuda diferenciados*: de acuerdo a la experiencia de cada usuario, los programas ofrecen ayudas distintas que facilitan el aprendizaje del usuario inexperto y la productividad del más experimentado.

Ha sido fundamental, para este desarrollo, un equipo de trabajo integrado por especialistas tanto de Bibliotecología como de Computación, los que intervienen directamente en el desarrollo de los sistemas.

5.2. Modularidad funcional

Esta idea está dirigida a ofrecer al usuario funciones con un alto grado de desacoplamiento, con dos finalidades:

- Que el sistema pueda crecer en forma gradual.
- Que un módulo pueda ser modificado o reemplazado en su totalidad sin afectar la interacción del usuario con el resto del sistema.

El sistema de Catalogación, por ejemplo, esta compuesto por los siguientes módulos funcionales:

- Entrada de datos, en conjunto con la Supervisión y la Producción de fichas,
- Base de datos bibliográficos.
- Base de datos de inventario.
- Producción de materiales.

Cada uno de ellos ha sufrido cambios con distinto grado de importancia, afectando (cuando lo hace) sólo la interacción del usuario con el módulo afectado. Dos ejemplos de modificación son:

- En este módulo de inventario se cambió el manejador de archivos: se usaba MDBS I (un paquete comercial) que fue reemplazado por un grupo de rutinas programadas localmente, denomidadas MABI (Manejador de Archivos Bibliográficos).

  Este cambio produjo alteraciones menores en la interfaz con el usuario y algunos más notorios en la administración del módulo (respaldos, informe de consistencia, etc.), pero no afectó en absoluto al resto de los programas.

- El módulo de Producción de Materiales fue cambiado radicalmente, debido a que era necesario obtener información más completa, tanto para la impresión de marbetes (la etiqueta que se coloca en el lomo del libro con el propósito de identificarlo) y etiquetas internas, como para controlar la cantidad de fichas de cada tipo a imprimir. Esta transformación cambió los procedimientos administrativos de impresión de materiales. pero no afectó al resto del sistema, ni la interacción con el usuario.

El hecho de que el sistema pueda ser montado en forma gradual se refleja en que se puede, por ejemplo, hacer trabajar primero el módulo de entrada de datos y supervisión, luego la impresión de fichas, a continuación la base de datos bibliográficos, posteriormente se puede instalar el inventario para el control de existencias y, finalmente, la producción de los materiales que acompañan al libro.

5.3. Independencia entre la lógica de la aplicación y la lógica de la implementación

Esto se ha logrado a través de dos técnicas de programación: modularidad estructural y niveles superpuestos. Estas dos técnicas estan íntimamente relacionadas, y a su vez, hacen posible un alto grado de portabilidad.

- Modularidad estructural: usando esta técnica, los programas son contruídos en base a piezas independientes que se comunican a través de interfaces bien definidas, de manera que los cambios en un sector del programas no afecten el resto.

- Niveles superpuestos: esta técnica, tomada de la filosofía de Software Tools, descrita por Kernighan y Plauger (11), significa desarrollar herramientas sencillas, que cumplan funciones puntuales y que combinadas forman herramientas más poderosas. Haciendo las combinaciones apropiadas, se puede construir programas bastante complejos, o se puede contruir un paquete de rutinas que faciliten el desarrollo de aplicaciones complejas. Nosotros hemos puesto énfacsis en esta técnica para el manejo de E/S (pantallas y discos, el área más dependiente de la implementación en cualquier proyecto de software).

El uso de estas técnicas posibilita la portabilidad : el desarrollo por niveles superpuestos permite sustentar toda la implementación en un grupo pequeño de rutinas que cumplen funciones básicas. Normalmente estas son dependientes del

hardware; p. ej. la forma de desplegar un caracter en modo reverso es distinta en una pantalla TeleVideo 912c que en una VT100. El aislar estas dependencias en las rutinas básicas permite hacer cambios de hardware sin afectar al conjunto de los programas, con lo cual se aumenta notoriamente el índice de portabilidad del sistema.

En nuestro caso, el cambio de un modelo de pantalla a otro, se resuelve modificando 4 rutinas en todo el sistema, con lo cual todos los programas pueden hacer uso extensivo de sus capacidades de manejo de pantalla. Las rutinas afectadas tiene, en total, alrededor de 60 líneas de código. Los sistemas que hacen uso de ellas, están compuestos de más de 50.000 líneas de código fuente, distribuídas en unas 2.500 rutinas y módulos re-utilizables.

El software de implementación desarrollado ha logrado proveernos de una amplia base de apoyo para la programación de las aplicaciones, que permite hoy escribir programas con manejo sofisticado de pantalla y archivos. Estas herramientas están en constante perfeccionamiento y ampliación. Todo este software está compuesto de un conjunto de módulos fuente (básicamente definiciones y constantes simbólicas) y de paquetes de rutinas objeto reubicables, que son integrados, los primeros a través de un Preprocesador, y los últimos a través de un Linker.

Esto se puede ejemplificar con los siguientes paquetes de módulos y filtros:

a) Módulos de E/S de pantalla.
   - Módulo básico para direccionamiento del cursor, escritura de un caracater, lectura de un caracter con o sin eco, manejo de las características de un caracter o campo de la pantalla (reverse, blink, blank, etc.), insertar o borrar caracteres o líneas (CONACT).

   - Módulo de manejo de strings de caracteres con edición, emisión de mensajes comunmente utilizados, tales como títulos, mensaje, peticiones de opciones, etc, (UTIL).

   - Módulo para E/S de pantallas completas, permite definir los campos que la componen, desplegarla y leerla (SCRMGR).

b) Módulos de E/S de disco
   - Módulo de E/S directa de bloques físicos de 128 bytes, en una dirección del archivo, incluyendo lock de registros (IO):

   - Módulo de E/S de registros lógicos usando una técnica de hashing, de manejo de colisiones (HASH).

   - Módulos para E/S de archivos específicos, hacen la lectura (con o sin lock), escritura, reescritura. creación y borrador de un registro lógico (XBD, UBD, CBD, OBD).

   - Módulo de funciones lógicas que afectan a más de un archivo; p. ej., en el sistema de circulación, una función efectúa las actualizaciones debidas al préstamo de un libro, en todos los registros afectados: anota el número de usuario en el registro de la copia y actualiza la cadena de préstamos del usuario (BD).

- Rutina que sirve de puerta de llamado para las rutinas de BD, maneja además las excepciones (DBKS).

c) Filtros para impresión de fichas catalográficas.
- El primer filtro genera fichas en un formato lógico (MSTP).

- El segundo filtro las imprime por una impresora específica (en nuestro caso, una NEC 5520). Este filtro conoce las peculiaridades de la impresora y es capaz de generar diversos diacríticos (acentos, diéresis, etc.), así como signos de puntuación y caracteres especiales (cedillas, enes, S alemana, etc.). Al cambiar de impresora, sólo hay que modificar este último filtro sin alterar el de generación de fichas, que es mucho más complejo (PRSTP).

5.4. Autómatas

Por autómatas entendemos un mecanismo lógico, que procesa una entrada y cuyo comportamiento está definido por cambios de estado interno en forma tal que, para cada símbolo a la entrada y estado interno, una tabla de transición define el nuevo estado en que debe quedar el autómata, una acción a seguir y (opcionalmente) un símbolo de salida.

Los autómatas, son herramientas que permiten dar mayor generalidad al software escrito, ya que el comportamiento del programa depende del contenido de las matrices de transición tanto como del algoritmo usado. Si el algoritmo es suficientemente general y las peculiaridades de la aplicación se detallan en las tablas, se logra tener programas con una baja sensibilidad a los cambios en los requerimientos de la aplicación. Adicionalmente, el examen de las combinaciones Estados/Entradas permite una mayor claridad conceptual del problema a resolver y un mejor control de los casos excepcionales, todo lo cual simplifica el desarrollo del algoritmo. El énfasis de esta técnica reside en la construcción de las tablas de transición que definen el comportamiento del autómata.

Nosotros hemos recurrido a autómatas para resolver varios problemas complejos de la aplicación. Hay un autómata en el programa de entrada de datos, varios más en el programa de impresión de fichas, otro en el programa de transacciones, etc. Estos autómatas persiguen fines distintos y su grado de complejidad es también distinto. A continuación se presentan algunos ejemplos de autómatas utilizados en el sistema.

- *El autómata para entrada de datos (Catalogación).*

  El formato MARC es general y puede ser utilizado para describir diferentes tipos de material bibliográfico (monografías, mapas, música, etc.). El formato MARC es actualizado periódicamente por la Biblioteca del Congreso de los E.E.U.U. El grupo de trabajo involucarado en el desarrollo del Sistema definió un subconjunto del formato MARC, que se utilizaría para el procesamiento de las monografías, el que es actualizado periódicamente. De este subconjunto, se escribió un resumen, el que incluye, para cada campo, su código, nombre extendido, algunos atributos (p. ej.: obligatorio, repetible, excluyente, etc.) y subcampos que lo componen. Para cada subcampo se incluye también su código, nombre extendido y atributos.

Este resumen es leído por un programa que construye las siguientes tablas:

- Código de campos.
- Código de subcampos.
- Nombres extendidos de campos.
- Nombres extendidos de subcampos.
- Obligatoriedad de campos y subcampos, y
- Códigos de subcampos válidos por cada campo.

Estas tablas son leídas por un programa que cosntruye un menu general de campos y menus de subcampos. Las tablas y los menus son utilizados por el programa de entrada de datos para proveer de ayudas al usuario, y para decidir la validez de la secuencia de códigos ingresados al programa.

Cualquier cambio al formato es ingresado en la descripción, la que es sometida a los programas constructores de tablas y de menús. Dado que las acciones que toma el programa de entrada de datos estan determinadas por el contenido de las tablas, que reflejan el estado de formato en un instante dado, no existe necesidad de hacer modificaciones en la lógica o estructura del programa.

El autómata para generación de fichas (Catalogación)

En una ficha catalográfica, los elementos que identifican un título se distinguen mediante tres mecanismos: diferentes párrafos, diferentes indentaciones y una puntuación específica que separa los campos y subcampos entre si.

El autómata lee registros en formato MARC y produce un set completo de fichas catalográficas

El programa dispone de la tabla de transición, que contiene la puntuación y algunas frases constantes como símbolos de salida. Esta tabla define un autómata que produce la puntuación que separa cada campo y subcampo, según las reglas AACR2. Los registros se digitan sin puntuación para evitar errores, acelerar la entrada de datos e independizar la información bibliográfica contenida en el registro, del medio en que la información será mostrada al usuario; un beneficio adicional es el ahorro de espacio de almacenamiento, que puede alcanzar hasta el 9% del total del registro.

6. CONCLUSIONES

Importancia de la interacción con el usuario

En nuestro sistema, la participación directa y constante del usuario en todas las etapas del desarrollo e implementación del SIBUC, ha demostrado ser fundamental para obtener un producto computacional que satisfaga sus necesidades. Este enfoque requiere de un grupo interdisciplinario que colabore estrechamente. En el esquema tradicional de desarrollo, el usuario es simplemente una ficción literaria (12), que permite darle homogeneidad y una sensación de validez al sistema, y el programador es sistemáticamente aislado del usuario real.

Por otro lado, los programas una vez terminados, satisfacen solamente una realidad momentánea, por cuanto la dinámica propia de cada aplicación hace necesario considerar futuras mejoras y cambios. Los criterios usados en el desarrollo de este proyecto han demostrado que la mentención y mejora del sistema puede simplificarse al involucrar al usuario en el desarrollo.

Productividad computacional:
El uso de las técnicas modernas de desarrollo de software, como son la modularidad y el uso de Software Tools, permiten mejorar la productividad dcomputacional, a través de la reutilización de código, en la forma de módulos y/o rutinas de uso general.

Portabilidad:
Tanto la separación de la lógica de la aplicación con la lógica de la implementación, en módulos independientes, como el uso de lenguajes de alto nivel, permiten mejorar significativamente el índice de portabilidad del sistema.

Uso de autómatas:
A través del uso de autómatas, definidos por tablas de transición que residen en archivos accesibles al usuario, es posible reducir el impacto de las modificaciones a la lógica de los programas, entregando la responsabilidad de la mantención al usuario. Además, la definición de un autómata exige realizar un estudio que se traduce en una formalización del problema, lo que su vez se traduce en una mejor comprensión de esos aspectos de la aplicación. Este es un ejemplo de retroalimentación de las Ciencias de la Computación hacia la disciplina que sustenta la aplicación (en nuestro caso la Bibliotecología).

Microcomputadores:
La evolución de la tecnología de los microcomputadores hace que estos sean cada vez más útiles para implementar aplicaciones complejas, en la medida en que se seleccione un equipo adecuado y que las aplicaciones se diseñen con criterios modernos de Ingeniería de Software.

ADDENDUM

En el último año (1985), el Sistema de Bibliotecas ha adquirido un minicomputador Data General MV4000, con 600 Mbytes en disco, para soportar la base de datos bibliográficos. En este computador se cuenta, además del SO AOS/VS, con UNIX.

A este equipo se ha portado la base de datos bibliográficos (*) desde el microcomputador de catalogación, con sus respectivos programas de administración en Pascal. Los índices de acceso (autor, clasificación, materias y títulos) se han implementado en INFOS II. El traspaso se realizó en sólo 15 días, en que quedó operativa la consulta a la base de datos desde PRESENT (un utilitario presentador de información de Data General).

El desarrollo siguiente en el Data General, ha continuado utilizando las ventajas de desarrollo que ofrece UNIX y el lenguje C. Además UNIX ofrece utilitarios que permiten realizar una serie de análisis sobre los datos bibliográficos (13).

Por otro lado se han realizado experiencias como MUMPS, SQL y DBMS para determinar los ambientes de implementación más adecuados para el SIBUC.

(*) A la fecha de revisión de este trabajo, la base de datos bibliográficos mantenía 27.000 registros en formato MARC, los que originan más de 150.000 puntos de acceso.

En el microcomputador de catalogación se mantiene la entrada de datos y la producción de materiales, de tal manera que se ha implementado un sistema de comunicaciones entre el microcomputador y el Data General a través de puertos RS232. El programa de comunicación en el Data General se ha implementado en C, y en el microcomputador en Pascal. Este sistema de comunicaciones permite transmitir los registros MARC de producción diaria para ser cargados en la base de datos central.

En la actualidad se están desarrollando nuevas aplicaciones y restructurando otras, entre las que destaca el sistema de inventario.

En ningún caso el Data General implica eliminar lo micros, más bien permite optimizar recursos computacionales, manteniendo el minicomputador para la administración de la base de datos bibliográfica que deberá superar los 150.000 registros bibliográficos y sus correspondientes ítemes de inventario. En los microcomputadores se mantendrán las funciones de entrada de datos, producción de materiales y circulación, liberando de estas tareas a la CPU del Data General.

En este contexto, la comunicación entre los micros y el mini es fundamental y se trabaja extensamente en este campo.

REFERENCIA

(1) Arenas, María Luisa y otros. El Sistema de Información Bibliográfica de la Pontificia Univerisidad Católica de Chile. Trabajo presentado a la Reunión Latinoamericana de Bibliotecas Nacionales y Universitarias, Santiago de Chile, 8-12 de Julio de 1984.

(2) Salas, Ricardo A Library cataloguing system using microcomputers. Analisys of an experience. Proceedings of the Interanational. Conference on the Application of Mini and Microcomputers in Information. Documentation and Libraries. Tel-Aviv, Isrrael, March 13-18, 1983 pp. 351-357.

(3) Hagler, Roland y Simmons, Peter. The bibliographic record and information technology. Chicago ALA, 1982 346 p.

(4) Anglo-American Cataloguing Rules prepared by The American Library Association, The British Library, The Canadian Committee on Cataloguing, The Library Association, The Library of Congress; edited by Michael Gorman and Paul W. Winkler. Second edition. Chicago, American Library Assocition, 1978 620 p.

(5) United States Librery of Congress. Subject Headings (9th edition). Washington, Library of Congress, 1980. 2v.

(6) Estados Unidos. Engineers Joint Council. Thesaurus of Engineering and Scientific Terms. New York, Engineers Joint Council, 1969. 690 p.

(7) United States. Library of Congress. Automated Systems Office. MARC Formats for Bibliographic Data. Washington. Library of Congress, 1980.

(8) Lancaster, F. Wilfed y Smith Linda C. Compatibility issues affecting information systems and services (PGI-83/WS/23). París, General Information programme and UNISIST, Unesco, 1983. 209 p.

(*) A la fecha de revisión de este trabajo, la base de datos bibliográficos mantenía 27.000 registros en formato MARC, los que originan más de 150.000 puntos de acceso.

(9) Estados Unidos American National Standards Institute, Inc. American national standard for bibliographic information interchange on magnetic tape (ANSI Z39 2-1979, revision of ANSI Z39 2-1971). New York, American National Standards Institue, c 1979. 12 p.

(10) international Organization for Standarization. Documentation-Format for bibliographic information interchange on magnetic tape (ISO 2709-1981 (E)). Switzerland, ISO 1981. 5 p.

(11) Kernighan, Brian W. y Plauger, P.J. Software Tools in Pascal. Reading, Mass., Addison-Wesley, 1981. 365 p.

(12) Kling, Rob. The organizational context of user-centered software designs. MIS Quarterly 1 (4), December 1977.

(13) Estudio de palabras claves en BDMARC (Informe técnico interno). Pontificia Universidad Católica, Sistema de Bibliotecas, Mayo 6, 1985. 19 p.