УДК 574/578+004.38

О.А. Литвинов

# ВИКОРИСТАННЯ МЕТАІНФОРМАЦІЇ ДЛЯ РЕАЛІЗАЦІЇ ГНУЧКОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА ГОСПІТАЛЬНОЇ СИСТЕМИ

**Постановка проблеми.** Формулювання загальної проблеми з якою пов'язана дана робота вичерпано надане в роботі [1]:

Яким чином моделі можуть використовуватися як засоби подання (representation tools) в розробці інтерактивних систем у сенсі інтеграції розробки інформаційних систем з розробкою інтерфейсу.

Комплекс питань пов'язаних з даною проблемою може бути розподілений на три загальних класи: які типи моделей (техніки формалізації, нотації, порядок їх застосування) можливо використовувати як джерело для інтерпретації/генерації інтерфейсу користувача; які засоби можливо використовувати для інтерпретації/відображення моделей на механізми реалізації; який ефект може бути отриманий від використання зв'язку «моделі – засоби» для реалізації тих чи інших задач інформаційних систем.

Основним напрямком пов'язаним з вказаним комплексом питань може вважатися модельно-орієнтована розробка інтерфейсу користувача (Model-Based User Interface Development )[1,2,3] . Другим аспектом побудови, з якою пов'язана дана робота є забезпечення динамічної інтерпретації моделей. Важливими напрямками у цьому сенсі є: MDA(model-driven architecture)[4], AOM(adaptive object models) [5].

**Аналіз публікацій та постановка задачі.** Практично в будь-якій інформаційної системі (ІС) можна виділити взаємодію "користувач - інтерфейс - бізнес логіка – база даних". З огляду на можливі зміни структури бази даних (БД), а також специфіку бізнес-логіки, що полягають у основі інформаційних процесів лікувального закладу, доцільно передбачити реалізацію гнучкості інтерфейсу користувача госпітальної інформаційної системи (ГІС), особливо відносно елементів доступу до даних рівня предметної області, враховуючі при



 ISSN 1562-9945

цьому такі важливі характеристики розробки, як пониження коштів розробки, спрощення процесу модифікації та повторного використання розроблених компонентів.

Засоби, які звичайно використовуються для вирішення задачі зміни можна поділити на три категорії[6]:

Засоби розробки інтерфейсів (Interface Builders) які існують в сучасних інтегрованих середовищах програмування (Delphi, Visual Studio та інш.) та дозволяють здійснювати проектування візуальної складової інтерфейсу користувача використовуючи стандартні елементи.

Системи керування інтерфейсом користувача (User Interface Management Systems), які підтримують розподілену реалізацію та модифікацію інтерфейсу та логіки програми та мають засоби декларативного опису моделі зв'язку «інтерфейс – логіка». Типовим прикладом є мова XAML.

Модельно-орієнтовані засоби розробки інтерфейсу (Model-Based Interface Development Systems), які дозволяють отримати специфікацію усього інтерфейсу та по специфікації згенерувати код інтерфейсу. Модель у загальному випадку охоплює: семантику додатку (об'єкти, оператори), рівень презентації(елементи), послідовність діалогу(обмеження порядку на команди) [1,2,3].

Підхід розробки інтерфейсу на базі онтологій. Авторами [6] декларується застосування декларативних моделей високого рівня з повним виключенням програмування на процедурній мові. Розробники мають справи з універсальними онтологіями для формування компонентів моделі конкретного інтерфейсу. Код інтерфейсу генерується автоматично на базі декларативної моделі.

Важливими рисами двох останніх підходів є: генерація на відміну від інтерпретації; претензія на повну генерацію інтерфейсу користувача без застосування процедурних мов; увага сфокусована на моделюванні, розробці, модифікації саме інтерфейсу.

Задача, яка розглядається в роботі: чи можна використовуючи єдину модель предметної області отримати відповідні зміни як елементів інтерфейсу, що відображають об'єкти типів заданих предметною областю та відповідної бізнес логіки(механізмів валідації заданих значень, контролювання взаємодії елементів).

**Основна частина.** У роботі [7] було запропоновано використати підхід проектування і реалізації системи, сутність якого полягає в інтерпретації метаінформації, що міститься у зовнішньому джерелі, на ту або іншу функціональну вісь системи. Основою підходу слід вважати використання фреймової парадигми опису знань[8] для репрезентації метаінформації. Основна від'ємність від об'єктно-орієнтованого складається в вилученні поняття «грані» при описі об'єктів. Так, тип, розмір, значення за замовченням та інші характеристики властивостей або атрибутів є гранями опису атрибутів. Структура таблиць, що містять мета-інформацію,показана на рис.1.

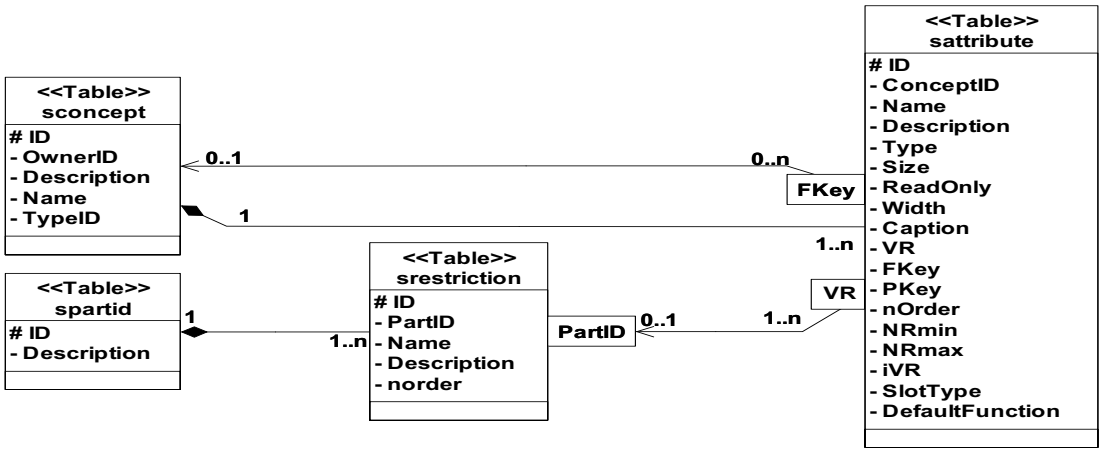


Рисунок 1 - Структура та зв'язок довідників метаінформації

В основі запропонованого підходу до застосування цієї моделі для генерації елементів інтерфейсу лежить побудова ряду візуальних компонентів, взаємодіючих з контуром "метаінформації", що дозволяє динамічно (у процесі роботи програми) створювати елементи інтерфейсу користувача з можливістю забезпечення контролю інформації, яка вводиться, на базі описаної моделі, при цьому зміна в моделі веде за собою одночасну зміну частини інтерфейсу додатка [9]. Використання тієї же моделі для задачі побудови аналітичної системи надано у роботі [10].

Розробка компонентів відповідаючи за інтерпретацію моделі базується на використанні шаблону Layers, що припускає нарощування рівня більш складних елементів над рівнем більш простих. Так до основних елементів найнижчого рівня можна віднести CComboBox, CTextBox, що наслідуються від відповідних

 ISSN 1562-9945

елементів Windows.Forms, які під час взаємодії з моделлю дозволяють реалізовувати елементи "простий атрибут" і "слот". Діаграма взаємодії елементів з контуром "метаінформації" показана на рис. 2. У якості "фасаду" даного рівня виступає клас CGUIFactory, що дозволяє без перешкод розширювати безліч елементів, виключаючи при цьому будь-які зміни інших компонентів додатку.

У якості прикладу компоненту більш високого рівня, представленого на рис. 2 класами CClass1 і CClass2, можна розглянути компонент "форма-об'єкт", завданням якого є запит і відображення візуальних елементів, що відповідають атрибутам об'єкта концепту по його імені (поле name у таблиці sconcept, де ім'я концепту є унікальним).

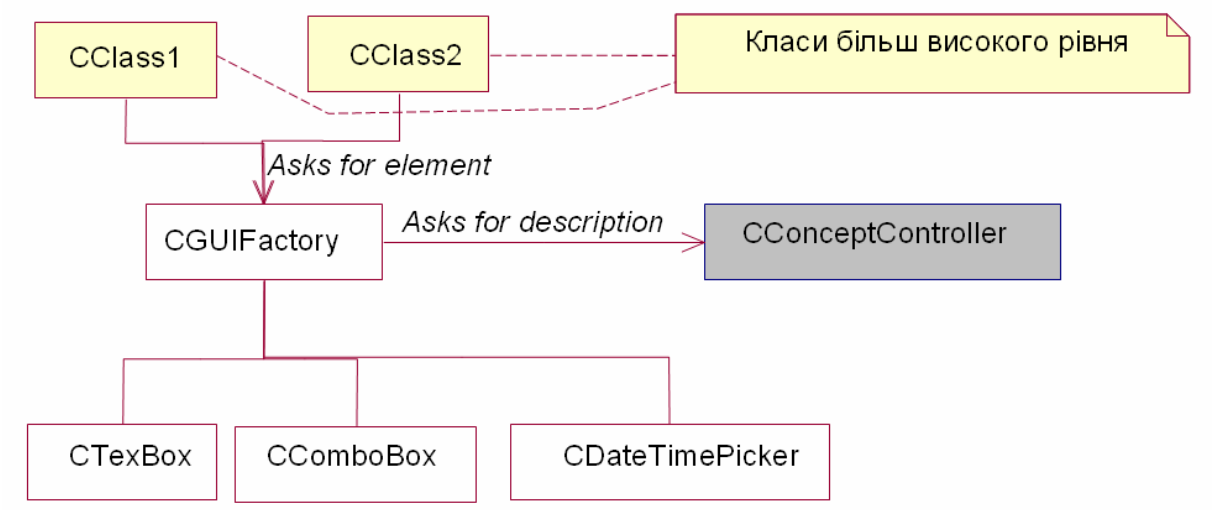


Рисунок 2 - Схема взаємодії компонентів вводу системи

При цьому у випадку редагування елементи містять поточні значення атрибутів, а у випадку додавання нового запису можуть бути порожніми або містити значення за умовчанням. Слід відзначити компонент для відображення множини записів-об'єктів заданого концепту, з огляду на особливості відображення й редагування атрибутів, які задані гранями відповідного опису моделі (caption, width, readonly, nrmin, nrmax і інш.). Класом-фасадом компонента є CDctGrid. Діаграма прецедентів компонента представлена на рис. 3, особливості представлені в табл. 1. На рис. 4 зображена діаграма, яка показує основні класи, що становлять компонент "Grid концепту", що включає класи із префіксом CDctGrid-, а також взаємодії компонента і контуру метаінформації (показані на рисунку більш темним кольором).

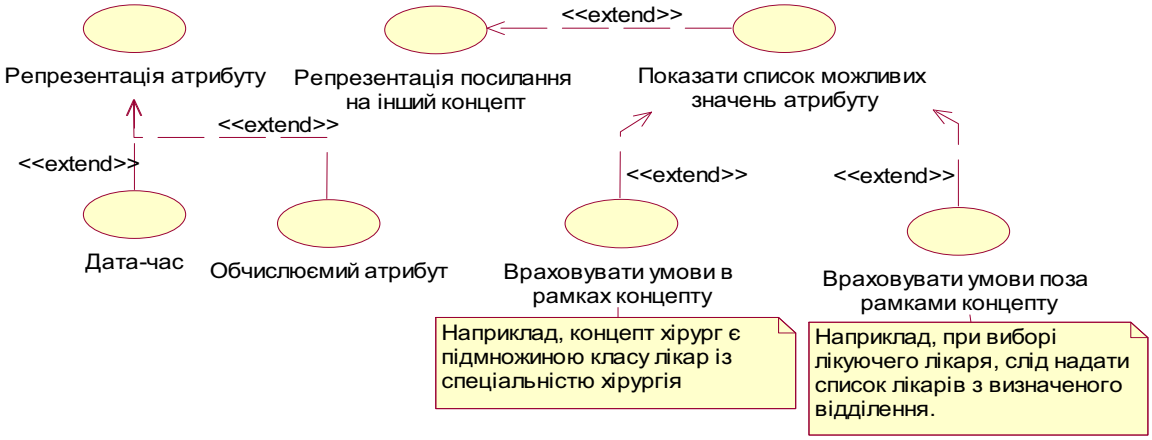


Рисунок 3 – Діаграма основних прецедентів елемента CDctGrid

Таблиця 1

Основні прецеденти використання компоненту CDctGrid

| Взаємозв'язок атрибутів | Проблема |
|---|---|
| Відображення атрибута таблиці | Необхідно відобразити тільки потрібні атрибути в заданому порядку |
| Атрибут є зовнішнім ключем для зв'язку 1:1 | Вивести додаткові атрибути зв'язаного концепту-таблиці |
| Атрибут є зовнішнім ключем для зв'язку 1::n | Вивести значення атрибута зв'язаного концепту-таблиці, при активації значення надати можливість вибору із списку можливих значень |
| | Те ж саме, але на список значень накладена умова |

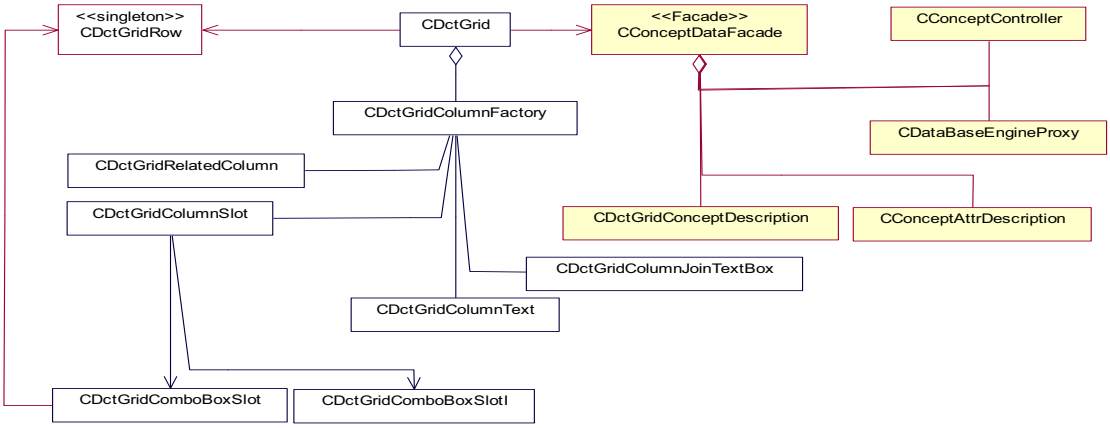


Рисунок 4 – Взаємодія класу CDctGrid з класами для роботи з мета даними


ISSN 1562-9945

Подальше вдосконалення підходу може привести до досить повного опису процесу побудови інтерфейсу користувача на базі описів, що зберігаються в одному або ряді xml-файлів, та їхнього динамічного відтворення в процесі роботи програми. Варто відзначити, що використання підходу розглядається в контексті розробки ГІС, що безумовно накладає свої обмеження як на елементи, так і на мову опису моделі; однак у рамках підходу, вибраного для проектування і реалізації, передбачається його подальше вдосконалення.

Також були розроблені засоби, що дозволяють шляхом зміни моделі предметної області, яка представлена у вигляді метаінформації, впливати на інтерфейс користувача, скорочуючи або зовсім не піддаючи при цьому змінам код програми. Тому реалізовані наступні можливості: приховання або відкриття тих або інших атрибутів від користувача; введення обмежень на значення та кількість значень атрибута; імплікація атрибутів; валідація, збереження, зміна, видалення відображуваних даних; автоматизація зв'язку з довідником; установка значень за умовчанням.

Слід зазначити, що розроблені елементи не дозволяють повністю вирішити питання рутинного програмування при побудові інтерфейсу користувача, однак істотно скорочують час роботи програміста при переробці коду, адаптації підсистем. Крім цього використання даного підходу дозволяє полегшити перехід системи з одного на інший клас візуальних елементів.

В цілому, поведінка елементів може бути описана в такий спосіб:

1. Контейнер візуальних елементів (форма або Grid) є пов'язаним з певною сутністю/групою сутностей, атрибути яких він відображає.

2. В процесі активації контейнер по унікальному імені концепту, з яким він зв'язаний, одержує список (ArrayList) елементів, що відсортовані за порядком відображення атрибутів, який було описаному в моделі. Розмір і позиція елемента в полі контейнера можуть бути представлені як грані (мета атрибутів) опису атрибута та бути задані безпосередньо в моделі.

3. Для реалізації спеціальних можливостей, які пов'язані з відображенням елементів на формі, передбачений XML-файл

конфігурації форми, що описує такі властивості елементів як розмір, позиція на формі.

4. Набір елементів відображається у відповідності до властивостей, що задані в списку елементів та файлі конфігурації.

5. Після зміни значень атрибутів і ініціалізації процедури збереження відбувається перевірка даних, що були введені, згідно значень відповідних граней в описі атрибута.

Але поряд з описаними перевагами зазначеного підходу слід відзначити наступні недоліки: для побудови складних форм, особливо з використанням технології прив'язки елементів до певної області форми (Dock), що дозволяє їм пропорційно змінювати розміри, необхідно будувати досить складну оболонку розробника, повторюючи при цьому вже розроблене візуальне середовище; інтерпретація потребує значно більше часу ніж виконання коду, особливо, якщо форма буде певної складності; наявність додаткових файлів з метаінформацією про форми може викликати невдоволення з боку користувачів; вигода від динамічної інтерпретації у цьому контексті незначна. Тобто такі ситуації, щоб не виходячи з програмного додатку змінювати зміст форми, змінювати метаінформацію і структуру БД, практично не виникають.

Виходячи з оцінки недоліків було прийнято рішення про зміну підходу щодо побудови інтерфейсу, не лишаючи при цьому зв'язку з метаінформацією. За цим підходом форми будуються в звичайному середовищі Visual Studio IDE, але постачальником елементів, що пов'язані з даними на засадах заданої метаінформації, є контур GUIFactory. Типовий сценарій роботи програміста за заданою методикою можна описати наступним чином: програміст будує нову форму і розташовує головний елемент TableLayoutPanel, що дозволяє зв'язаним з ним елементам змінювати свій розмір пропорційно до змін розмірів форми; програміст розміщує елементи TableLayoutPanel або Panel в ячейках агрегуючого елементу TableLayoutPanel більш високого рівня; назва (атрибут Name) відповідного елементу Panel має бути такою, як і відповідний атрибут з префіксом "p". Далі форма в процесі ініціалізації запрошує набір елементів у контуру GUIFactory та розташовує їх відповідно до заданої розмітки.

 ISSN 1562-9945

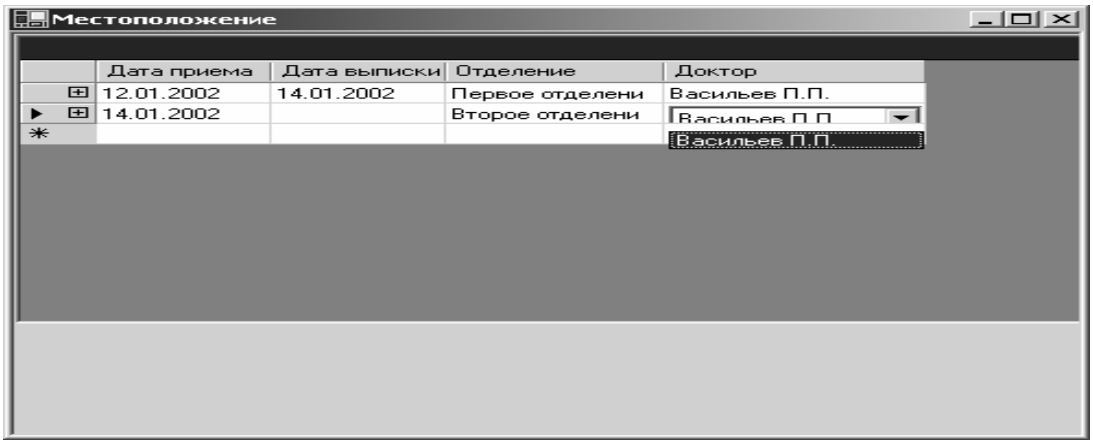


Рисунок 5 – Форма введення місцезнаходження пацієнта

Використання елемента CDctGrid показано на рис. 5. Так по моделі концепту «переведення» будується наступна форма, що включає в себе елемент CDctGrid, який відображає список переведень даного пацієнта. При цьому бізнес-правило «лікар може бути тільки із призначеного відділення», описане в моделі як «імплікація атрибутів», відображається на інтерфейс таким чином, що без вибору відділення не може бути задане значення атрибута «лікар», а при призначенні відділення, значення атрибута «лікар» може бути задано тільки зі списку лікарів, що працюють у призначеному відділенні. Визначним, з погляду складності організації, може служити концепт «операції», в якому всі поля є зовнішніми ключами, що пов'язують даний концепт з іншими концептами. Так «етап лікування», «тип операції» відносяться до довідника обмежень, хірург і асистенти відносяться до лікарів, спеціалізація яких – хірургія; анестезіолог – із спеціалізацією анестезіологія.

**Висновки.** Використання фреймової парадигми є ефективним засобом опису метаінформації інформаційної системи, що забезпечує гнучкість ІС і дозволяє уникнути недоліків об'єктно-орієнтованої моделі пов'язаних із відсутністю поняття «грані» в об'єктна-орієнтованих системах. Представлена модель метаінформації може ефективно застосовуватися при побудові інтерфейсу користувача.

## ЛІТЕРАТУРА

1. Hallvard Trжtteberg. Model-based User Interface Design.// Information Systems Group Department of Computer and Information Sciences Faculty of Information Technology, Mathematics and

Electrical Engineering Norwegian University of Science and Technology. 2002. P.204.
2. F. Bodart, A.-M. Hennebert, J.-M., Leheureux, J. Vanderdonckt, A Model-based Approach to Presentation: A Continuum from Task Analysis to Prototype, Chapter 6, in "Interactive Systems: Design, Specification and Verification", F. Paterno (ed.), Focus on Computer Graphics Series, Springer-Verlag , Berlin, 1995, pp. 77-94
3. MDA home page. – Режим доступу: http://www.omg.org/mda/.
4. OMG presentations and papers. http://www.omg.org/mda/presentations.htm.
5. The list of papers of Joseph W. Yoder. http: //www.joeyoder.com/papers/.
6. В.В.Грибова, А.С. Клещев. Концепция разработки пользовательского интерфейса на основе онтологий. Вестник ДВО РАН, №6.2005.-с.123-128
7. Литвинов А.А. Использование метаданных при проектировании и реализации медицинской информационной системы // Системные технологии. Рег. меж вуз. сб. научн. работ. – Вып. 6(35). – Днепропетровск, 2004. – С. 33–39.
8. Karp P.D. The Design Space of Frame Knowledge Representation Systems // Technical Report 520, SRI International Artificial Intelligence Center.–1992.– 49 p.
9. Литвинов А.А. Использование метаинформации при реализации пользовательского интерфейса медицинской госпитальной системы // Математическое и программное обеспечение интеллектуальных систем: Международная научно–практическая конференция: Сб. тез. – Днепропетровск, 2005. – С. 106–107.
10. Литвинов А.А. Использование метаинформации при построении аналитической подсистемы медицинской госпитальной системы // Системные технологии. Региональный межвузовский сборник научных работ. – Вып. 6(41). – Днепропетровск, 2005. – С. 105–114.

Одержано 23.11.2008р.