Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра прикладної математики

Реєстраційний № _____

На правах рукопису

«Допустити до захисту»
Зав. кафедрою, д.т.н., проф.
_____ Молчанов О. А.
«_____» _____ 2005 р.

АТЕСТАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Дослідження та розробка методів та засобів побудови систем управління неструктурованими інформаційними потоками»

зі спеціальності 8.08.02.02 «Прикладна математика»

Науковий керівник:
кандидат технічних наук, доцент
Маслянко Павло Павлович
_____ /підпис, дата/

Консультанти:
З організаційно-економічного розділу
Березовський К. В.
_____

Магістрант:
Сохацький Максим Еротейович
_____ /підпис, дата/

Київ 2005

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Факультет прикладної математики — Кафедра прикладної математики

Спеціальність 08.080202 — Прикладна математика

«Затверджую»
Зав. кафедрою, д.т.н., проф.
О. А. Молчанов
_____ (підпис)
«1» вересня 2001 р.

ЗАВДАННЯ

на атестаційну магістерську роботу

Сохацького Максима Еротейовича

(прізвище, ім'я, по батькові магістранта)

1. **Тема роботи:**
   «Дослідження та розробка методів та засобів побудови систем управління неструктурованими інформаційними потоками»
   затверджена наказом по університету від «_____» _____ 200___ р. №_____
2. **Термін здачі студентом закінченої роботи:** 1 червня 2005 р.
3. **Вихідні дані (характеристика об'єкта, умов дослідження та ін.):**
   _____
4. **Основні задачі дослідження:**
   _____
5. **Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):**
   _____

Консультанти по проекту, із зазначенням розділів проекту, що стосуються їх

Розділ	**Консультант**	**Підпис, дата**
Проектно-конструкторський	Маслянко П.П.	_____
Організаційно-економічний	Березовський К.В.	_____

Завдання видав: _____ Маслянко П.П.
Завдання прийняв: _____ Сохацький М.Е.

Дата видачі завдання: 1 вересня 2003 р.

Керівник: _____ Маслянко П.П.
Завдання прийняла до виконання: _____ Сохацький М.Е.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів магістерської роботи	**Термін виконання етапів роботи**	**Примітка**
1. Вивчення предметної області, огляд літератури	31.12.2003	
2. Постановка та формалізація задачі	01.03.2004	
3. Побудова математичної модельної бази	01.06.2004	
4. Розробка методів побудови компонент системи	01.09.2004	
5. Розробка інформаційного забезпечення	01.12.2004	
6. Тестування програмного забезпечення	01.03.2005	
7. Проведення експериментальних розрахунків	01.04.2005	
8. Апробація отриманих наукових і практичних результатів	20.04.2005	
9. Вирішення питань економіки	05.05.2005	
10. Виконання графічних документів	10.05.2005	
11. Оформлення пояснювальної записки	15.05.2005	
12. Подання АМР рецензенту	20.05.2005	
13. Подання АМР на попередній захист на кафедрі	25.06.2005	
14. Подання АМР на основний захист	01.06.2005	

Студент-магістрант: _____ Сохацький М.Е.
Керівник роботи: _____ Маслянко П.П.

Зміст

Анотація

Об'єкт дослідження: Організаційні структури та системи управління цими структурами. Управління організаційними структурами перебуває на межі таких дисциплін, як економіка, менеджмент, теорія управління, теорія систем. Дослідження організаційних структур ведуться давно, з того часу, коли почали з'являтися перші великі компанії зі складною організаційною структурою. Саме тоді виникла потреба в якісному управлінні, що спричинило появу питань моделювання та реінжинірингу.

Теорія управління організаційними структурами вирішує такі проблеми, як опис роботи підприємства, реінжиніринг бізнес-процесів підприємства, створення автоматизованих систем, що використовуються на підприємстві — автоматизація діловодства, логістики, фінансового управління, адміністративного управління.

Питання, що висвітлюються в роботі: здатність до адаптації, самоуправління, нові рівні контролю та аудиту, інтелектуалізація автоматизованих систем. Із середини 80-х років починає розвиватися така галузь, як управління процесами, що відбуваються в організаційних структурах. В англомовній літературі автоматизовані системи, які реалізують управління процесами, називаються «Workflow Systems» або системи управління потоками. У нас застосовується термін «управління процесами» або «управління неструктурованою інформацією».

Мета роботи: По-перше, актуалізація головних проблем, які виникають під час створення систем управління процесами; по-друге, створення архітектури та прототипу системи управління процесами, використовуючи досвід і помилки попередників. Також хотілося б, щоб ця робота була вступом до теорії управління процесами, що відбуваються в організаційних структурах.

Ціль роботи: Створення набору забезпечень — математичного, лінгвістичного, інформаційного, програмного — тобто засобів для побудови систем управління процесами.

Розроблені методи та моделі становлять єдину інфраструктуру для опису та розробки систем управління неструктурованою інформацією. Багато методів і моделей пройшли перевірку практикою і успішно використовуються в реальних системах [LRS, DSS]. Деякі методи є інноваційними прототипами і представлені в роботі вперше.

Розділ 1

Управління процесами

1.1 Вступ до предмета

Об'єктом дослідження цієї роботи є організаційні структури, а також системи управління цими структурами. Управління організаційними структурами перебуває на стику таких дисциплін, як економіка, менеджмент, теорія управління та теорія систем. Дослідження організаційних структур ведуться давно, з того часу, коли почали з'являтися перші великі компанії зі складними організаційними структурами. Саме тоді виникла потреба в якісному управлінні, що спричинило появу питань моделювання та реінжинірингу.

Теорія управління організаційними структурами вирішує такі проблеми, як опис роботи підприємства, реінжиніринг бізнес-процесів підприємства, створення автоматизованих систем, які використовуються на підприємстві — автоматизація діловодства, логістики, фінансового управління, адміністративного управління.

Питання, що піднімаються в роботі: здатність до адаптації, самоуправління, нові рівні контролю та аудиту, інтелектуалізація автоматизованих систем. З середини 80-х років почала розвиватися така галузь, як управління процесами, що відбуваються в організаційних структурах. В англомовній літературі автоматизовані системи, які реалізують управління процесами, називаються «Workflow Systems» або системи управління потоками. У нас застосовується термін «управління процесами» або «управління неструктурованою інформацією».

Цілі, поставлені автором роботи, такі: по-перше, актуалізація основних проблем, які виникають під час створення систем управління процесами; по-друге, створення архітектури та прототипу системи управління процесами, використовуючи досвід і помилки попередників. Також хотілося б, щоб ця робота слугувала введенням у теорію управління процесами, що протікають в організаційних структурах.

1.2 Структури та процеси

У вивченні організаційних структур, їх моделюванні та застосуванні використовуються знання з різних дисциплін, таких як соціологія, психологія, але переважно це, звісно, математика, економіка, менеджмент і діловодство.

Першою прикладною галуззю організації процесів були системи автоматизації, де моделювалися автоматизовані пристрої та їхні вузли. Тут головним чином сформувався базовий апарат математичного забезпечення.

Друга, більш актуальна для нас, галузь дослідження організації процесів — це адміністративне управління організаційними структурами. Основні цілі тут — отримання моделей, життєздатних та ефективних з точки зору економічних показників, а також методів їх застосування на практиці.

Багато робіт після 70-х років порушували питання побудови функціональної бази та способів опису організаційних структур і процесів. Більшість досягнень були втілені в життя і зараз є стандартом де-факто на підприємствах, особливо в управлінні великими компаніями. Вони знайшли розвиток у вигляді стандартів побудови автоматизованих систем для підприємств. Дослідження орієнтувалися на розв'язання економічних та оптимізаційних завдань: скорочення циклу виробництва, оптимізацію організаційних структур, побудову динамічних, легко адаптованих моделей. Моделювання бізнес-процесів — це основа досліджень у галузі організаційних структур.

Управління процесами іноді називають обробкою неструктурованої інформації, підкреслюючи характеристику наповнення, оскільки фактично невідомо, чим маніпулює процес і від чого він залежить.

Існуючі мови та стандарти можна поділити на два множини. Перші, такі як SADT, IDEF, UML, були створені в 80–90-х роках минулого століття в рамках теорії інформаційних систем і застосовувалися для опису організаційних структур. Другі, такі як XPDL, BP4L, створювалися з урахуванням попереднього досвіду і описують бізнес-процеси. Фактично теорія інформаційних систем мігрує до теорії управління процесами. Застосування систем управління процесами підвищує значущість існуючих інформаційних систем і допомагає підприємствам здійснити перехід до процесно-орієнтованого способу управління. Процесно-орієнтований спосіб організації виробництва має низку переваг над проблемно-орієнтованим, основні з яких — це гнучкість структур, незалежність від окремих вузлів (відділів, людей), самодокументованість (приклад: процес управління CMMI).

Таким чином, спостерігається зміщення дослідницької складової від способів опису організаційних структур та їхніх зв'язків у бік процесів, які протікають у цих структурах. Таким чином відбувається доповнення теорії управління організаційними структурами теорією управління процесами.

1.3 Самоорганізація

Впровадження процесно-орієнтованого опису організаційних структур — лише перший крок на шляху до управління. Для отримання всіх переваг необхідно мати постійну підтримку системи, контроль над бізнес-процесом і можливість його адаптації.

В управлінні процесами необхідно вирішувати такі завдання, як проєктування, впровадження, розширення, контроль процесу, а також адаптація процесу відповідно до результатів контролю.

Не завжди моделі, створені аналітиками, приживаються в реальному середовищі, тому системи управління процесами повинні мати здатність до самоорганізації, адаптації та стабілізації в процесі функціонування. Тому системи управління процесами охоплюють повний цикл проєктування, розгортання та функціонування організаційних структур:

- **Проєктування процесу.** Аналітики проєктують або переглядають наявну модель роботи організаційної структури, описуючи процеси, що в ній відбуваються. На цьому етапі модель описується та оптимізується. У процесі визначення моделі використовуються такі методи моделювання, як мережі Петрі та подійно-керовані ланцюжки правил. Більшість підходів фокусуються саме на цій стадії.
- **Впровадження нового процесу та вимірювання ефективності нової моделі.** У процесі впровадження нового процесу в організаційній структурі структура видозмінюється відповідно до вимог процесу. Нові вузли (люди) навчаються тих частин процесу, які змінилися. Метрики ефективності (переважно пропускна здатність) у процесі роботи нової впровадженої структури збираються для подальшого аналізу.
- **Контроль і маніпуляція результатами.** На основі даних, зібраних на попередній стадії, а також на основі економічних показників, аналізуються та профілюються окремі вузли процесу, порівнюючи прогнози, які були зроблені під час зміни процесу, з тим, що фактично отримано. Крім того, на основі профілювання обираються вузли, де можлива подальша оптимізація.
- **Визначення необхідних змін.** Аналітична стадія реінжинірингу. Тут ухвалюється стратегічне рішення щодо подальшого вдосконалення системи. Створюються варіанти перепланування і, на конкурентній основі, обирається план-кандидат для реінжинірингу.

Цей цикл у літературі називається циклом Демінга. Для реалізації повного циклу управління процесом модель цієї системи має бути максимально гнучкою, динамічною та пристосованою до такого роду змін. Це неможливо здійснити, застосовуючи стандарти SADT, IDEF, які були створені для розв'язання завдань, що виникають лише на першому етапі життєвого циклу.

Цей цикл наочно ілюструє охоплення завдань, які адресують системи управління процесами. Одним із найважливіших завдань управління процесами на практиці є впровадження та підтримка системи, створюючи інфраструктуру (середовище) для інформаційної системи. Після цього система в автоматичному режимі збирає всю інформацію, необхідну для контролю та управління.

Основна причина того, що в більшості випадків це вдається з великими труднощами, — це, по-перше, непрозорість інтеграційної інфраструктури (практична складова). По-друге, це незавершеність набору індикаторів для вимірювання продуктивності наявної схеми поточного процесу (теоретична складова). Ця робота адресована практичній складовій проблеми створення методів для побудови систем управління процесами (або систем управління неструктурованою інформацією).

Процесно-орієнтоване управління організаційними структурами дає змогу організаціям точніше контролювати виробничий процес і дозволяє контролювати не лише економічні показники, а й показники продуктивності самого процесу, даючи змогу подивитися на процес ізсередини. Лише вдосконалення підходів і наявність заверненої теорії управління процесами дасть змогу перейти до процесно-орієнтованої моделі управління організаційними структурами.

1.4 Аспекти управління

Слід зазначити, що процесно-орієнтована модель не замінює модель, орієнтовану на організаційну структуру, а розширює та доповнює її. Місце процесно-орієнтованої моделі контролю перебуває в центрі таких протилежностей, як контроль у бізнесі та виробництві, контроль кількості та якості.

З точки зору управління та контролю, теорія систем управління процесами охоплює такі аспекти:

- **Управління показниками.** Управління простими показниками продуктивності процесу в реальних умовах. Наприклад, як покращити час, витрачений на обробку заявок вузлами. Головний постулат тут такий: якщо певний показник неможливо виміряти, то неможливо його контролювати, а отже, неможливо управляти.

- **Управління потоком заявок.** Перенаправлення потоку, виключення вузлів залежно від пропускної здатності потоку, усі основні засоби впливу на потік. Тут зачіпаються дані, які беруть участь у потоці, а також правила, що входять у процес. Потрібно забезпечити потужні та гнучкі механізми перенаправлення потоку.

- **Приклад 1:** У CMMI є рівень, який вимагає розрахунку ризиків. Управління ризиками — це прогнозування. Наприклад, програміст по понеділках і після свят мало пише (комітить) коду в репозиторій,

отже, система автоматично прогнозує виконання обсягів роботи, використовуючи показники пропускної здатності.

- **Приклад 2:** Один співробітник іде у відпустку такого числа, інший — пізніше, тому немає сенсу надсилати заявки першому співробітнику, якщо він не встигне їх завершити і піде у відпустку.

- **Контроль системних процесів.** Системне програмування в управлінні процесами. Розробникам необхідно знати, що в критичних вузлах має бути логіка отримання аудиторської інформації — статистики проходження потоку. Особливо це важливо на стику різних систем управління процесами. Зазвичай це називається відлагоджувальною інформацією процесу.

- **Контроль у процесі предметної області.** Прикладне програмування систем управління процесами. Розстановка необхідних вимірювальних точок, впровадження бізнес-правил для створення індикаторів, для моніторингу потоку правилами, зміна потоку. Аналітики, розробники процесів повинні мати можливість додавати все це в опис процесу, створюючи таким чином власні показники.

- **Сховище даних.** Банки знань, управлінської та статистичної інформації переважно використовують інформацію, отриману від транзакційних систем зберігання. Цей тип даних являє собою бізнес-об'єкти предметної області, або, як ще кажуть, операційні дані компанії. Самі правила системи та структури разом із керуючою інформацією зберігаються в одному зі сховищ даних. Сховище даних має бути версіонованим, відмовостійким тощо.

1.5 Опис роботи та цілі

Мета цієї роботи — створення набору забезпечень — математичного, лінгвістичного, інформаційного, програмного, — тобто засобів для побудови систем управління процесами.

Розроблені методи та моделі становлять єдину інфраструктуру для опису та розробки систем управління неструктурованою інформацією. Багато методів і моделей пройшли перевірку на практиці та успішно використовуються в реальних системах [LRS, DSS]. Багато методів є інноваційними прототипами і представлені в роботі вперше. Частина дослідницької роботи була виконана в рамках інших дослідницьких комерційних проєктів [NovaLIS].

1.6 Структура роботи

Ця книга складається з п'яти розділів.

- **Розділ 1. Управління процесами.** Вступ до теорії управління системами обробки неструктурованої інформації (системами управління процесами або системами управління потоками). Опис цілей роботи.
- **Розділ 2. Огляд літератури.** Огляд наявних підходів. Набір базових математичних методів. Огляд недоліків. Огляд різних забезпечень, які існують у галузі цього типу систем.
- **Розділ 3. Проєктування.** Методи та моделі, визначені в рамках цієї роботи. Створення вимог, яким має відповідати система. Розв'язання цих завдань. Розробка прототипу та філософії.
- **Розділ 4. Реалізація.** Технічні аспекти реалізації, стандартів, алгоритмів і технологій, за допомогою яких створювався набір засобів для побудови систем управління процесами.
- **Розділ 5. Випробування та аналіз.** Висновки, огляд наявних систем управління процесами. Що хотілося і що вийшло. Аналіз результатів. Висновок.

Розділ 2

Огляд літератури

2.1 Забезпечення

При створенні автоматизованих систем, систем документообігу та систем планування насамперед розрізняють різні види забезпечень. У цьому розділі ми спробуємо систематизувати різноманітну теоретичну та історичну інформацію щодо видів забезпечень.

- **Математичне.** Математичне забезпечення (МЗ) — це методи та алгоритми розрахунків для прийняття рішень, які використовуються в інформаційних технологіях та автоматизованих системах. Наприклад, у цій роботі застосовувався алгоритм пошуку шляху розв'язання на просторі предикатів. Також використовувався аксіоматичний підхід під час опису мови опису процесів.

- **Лінгвістичне.** Лінгвістичне забезпечення (ЛЗ) — це сукупність мов для реалізації та експлуатації автоматизованих систем. Сюди насамперед входять мова предметної області, на якій описується робота підприємства, а також мови, на яких базується основна мова. У цій роботі як єдина мова був обраний LISP, який максимально підходить для створення багатошарових наборів мовних конструкцій для оперування над бізнес-об'єктами. За допомогою ECA бізнес-правил, написаних на LISP, задаються всілякі шляхи проходження документів.

- **Інформаційне.** Інформаційне забезпечення (ІЗ) — це простір даних, якими оперує система документообігу. За основу при розробці системи зберігання документів бралася модель Document Object Model (XML DOM), а також внутрішня розширена структура документа ILS з можливістю конвертації в реляційну базу даних або вертикальну модель.

- **Програмне.** Програмне забезпечення (ПЗ) — це сукупність програмних модулів, необхідних для функціонування системи.

Головним чином, це сервер застосунків підсистеми документообігу та сервер зберігання документів.

- **Організаційне.** Організаційне забезпечення (ОЗ) — це сукупність людей або програм, які беруть участь у системі документообігу, доповнюючи інформаційну систему даними, необхідними для її функціонування.
- **Технічне.** Технічне забезпечення (ТЗ) — це сукупність апаратних засобів, засобів зв'язку та обладнання, необхідного для функціонування системи.

Усі ці види забезпечень можна поділити на дві області: предметне забезпечення та технічне забезпечення. Мета цього розділу — зібрати воєдино всі знання, які використовуються при описі системи управління процесами.

2.2 Предметна область

Кожна компанія в широкому сенсі є постачальником товарів або послуг. Для досягнення цього мають бути впроваджені різні логістичні процеси для виробництва, зберігання, доставки товарів або послуг. Логістичні процеси пов'язані з фінансовими процесами, що протікають у системі, які спрямовані на збільшення грошових коштів. Логістичні процеси та процеси руху фінансових потоків формують операційну систему компанії. Мета комерційних підприємств — виробництво матеріального або нематеріального продукту (у державних установах мета інша — управління соціальними структурами). Традиційно виконання завдань в операційній системі підприємства, прийняття рішень здійснюється різними учасниками процесу. Тому координація активностей учасників процесу має бути такою, щоб відповідати головній меті підприємства.

Складні системи

Для опису організаційних структур використовують теорію складних систем. Теорія складних систем оперує системами в цілому, тобто їхніми структурами та поведінкою. Ця теорія складається з набору напрямів, започаткованих у 40-х роках XX століття. Один із напрямів — біологічні системи, саме з цього напряму з'явилося поняття кібернетики. Одночасно з цим Шенноном була розроблена математична база для опису доставки повідомлень в інформаційних системах.

Вступ до теорії складних систем Одне з визначень складної системи звучить так: система — це набір елементів, зв'язків між ними та середовища, якщо система відкрита. Системи можуть бути класифіковані за різними наборами атрибутів, такими як відкритість, складність,

динамічність тощо. Якщо існують відносини (зв'язки) між елементами системи (або підсистеми) та її середовищем, то така система називається відкритою.

Стан системи — це знімок елементів і зв'язків між ними в певний момент часу, а також набір атрибутів, асоційованих із кожним елементом системи або її відношенням. Те, як часто змінюється стан системи, визначає її динамічність. Динамічні системи змінюють свій стан залежно від вхідних даних.

Зазвичай системи розглядаються в певних зрізах:

- **Ієрархічна будова системи.** Тут стоїть завдання розбити систему на композицію підсистем. Під час побудови систем, орієнтуючись на це завдання, розрізняють проєктування зверху вниз і знизу вгору.
- **Функціональна будова системи.** Тут головним атрибутом є поведінка елемента або підсистеми. Елемент представляється як чорна скринька, у якій є входи та виходи. Статичні системи завжди на однакових вхідних даних продукують однакові вихідні дані. Функціональний аналіз намагається пояснити процеси, що відбуваються в системі, та її поведінку з позиції динаміки.
- **Структурна будова системи.** Тут виділяються зв'язки між елементами системи. Структурний аналіз займається виділенням кластерів (куди входить множина елементів), усередині яких відносини сильніші, ніж відносини з іншими кластерами.

Рис. 2.1: Різні погляди на систему

Місце для рисунка: Різні погляди на систему

Небезпеки системного підходу Організаційні структури часто сприймаються та описуються як структури та системи. Але насправді це набагато більше, ніж те, що може описати теорія систем. Системний підхід — це лише один із поглядів на організаційні структури. Проблема полягає в описі загальної поведінки системи. У теорії мета та мотивація поведінки кожної частини системи проектуються на цілі та поведінку всієї системи. Але узагальнення поведінки учасників процесу до верхнього рівня часто є проблематичним.

Організації

Компанії як соціотехнічні конгломерати Організації та комерційні компанії являють собою соціотехнічні організаційні структури, які керуються певними цілями, мають певне призначення і є відкритими, тобто мають відкриті зв'язки для клієнтів або інших організаційних структур.

Робота всіх організаційних структур побудована на використанні певного процесу — вказівок того, як має виконуватися робота. Ми дамо визначення процесу більш точно нижче. Поки що поговоримо про показники ефективності бізнес-процесів. Групи показників, що характеризують бізнес-процес, можна класифікувати так: показники виконання, вартості, ефективності, якості, спостережуваності, керованості. Правильний набір показників, управління ними та правильне реагування на них у процесі виконання бізнес-процесу — запорука ефективності бізнес-процесу.

Управлінська та операційна частини компанії Базовий поділ, який було зроблено під час аналізу роботи організаційних структур, — це відокремлення операційної частини (поточні дані, повсякденна робота) підприємства від частини прийняття рішень (управлінська частина). Їх так і називають: система прийняття рішень (СПР) і автоматична система обробки даних (АСОД).

Рис. 2.2: Системний погляд на організаційну структуру

Місце для рисунка: Системний погляд на організаційну структуру

Процес

Визначення процесу Основні процеси, які відбуваються в операційній системі компанії, — це фінансові процеси, процеси логістики тощо. Ці процеси описують підприємство в динаміці, як саме відбувається виробництво товарів і послуг. Детально описується робота відділів, ланцюжок від постачальників сировини або заявок до виробництва товару чи послуги та їх відвантаження клієнтам, а також залучення та обслуговування клієнтів.

Головна ідея процесно-орієнтованого опису організаційних структур — це акцент на послідовності виконання робіт, що беруть участь у процесі. Існує багато визначень процесу:

1. Процес — це набір активностей, які мають входи та виходи.
2. Процес — це структурований, упорядкований, вимірюваний набір активностей, яким подаються вхідні дані, вони виконують над ними роботу та передають роботу далі іншим активностям. Це чітко визначає, як виконується робота, а не те, над чим виконуються маніпуляції.

Усі визначення зводяться до схеми «отримав-опрацював-передав далі». Деякі визначення накладають певні обмеження на динаміку процесу та його впорядкованість. У системах обробки знань процеси визначаються як набори правил або предикатів.

Бізнес-процеси — це певна категорія процесів. Бізнес-процес визначається як описана на високому рівні схема роботи підсистеми підприємства для отримання максимальної ефективності за тими чи іншими показниками, керуючись цілями підприємства.

Рис. 2.3: Спосіб завдання процесу — діаграма потоків

Місце для рисунка: Діаграма потоків

Для схематичного відображення процесу використовується діаграма потоку робіт. Потік робіт — засіб відображення процесу — описує набір активностей та їхню структуру, відносини між ними, і визначає учасників процесу, які братимуть участь у процесі та матимуть доступ до системи, яка все це обробляє, тобто системи управління процесами. Прототипом для сучасних стандартів опису процесів були такі стандарти, як SADT Data Flow Diagram, IDEF 3. Зараз це XPDL, B4PL, YAWL.

Діаграма потоків поєднує в собі всі погляди на систему: ієрархічний (у ролі активностей виступають підпроцеси), функціональний (визначаються правила руху інформації через активності, у активностей є входи та виходи) і структурний.

Усі процеси операційної системи підприємства можна назвати одним словом — інформаційний процес, який відображає рух документів та інших даних компанії по відділах компанії.

Таксономія процесу

- **Агенти.** Учасники процесу називаються агентами. Їхнє завдання — виконати ті роботи, які вони мають виконати зараз, тобто роботи, закріплені за ними системою. Агентами можуть бути як люди, так і технічні ресурси або програми. Процес роботи відбувається так: агент отримує дані від системи, виконує над ними певну дію та відправляє дані назад у систему.

- **Повідомлення.** Набір даних, об'єднаних в одну порцію, називається повідомленням. Це елементарна одиниця потоку, роботи якої рухаються по процесу. Повідомлення — це ті дані, над якими агенти виконують маніпуляції. Повідомлення може являти собою як один документ, так і набір документів, пов'язаних якимись зв'язками або непов'язаних.

- **Рівень абстракції.** Опис процесу описує послідовність робіт, які мають виконуватися для досягнення цілей процесу. Однак у реальності процеси в системі являють собою реальні дані (повідомлення), які рухаються по процесу. Набір даних, який обробляється за певною схемою (процесом), називається екземпляром процесу.

- **Область видимості процесу.** Усі процеси можна поділити на внутрішні, які описують роботу відділів компанії, або зовнішні, які використовуються для взаємодії з іншими компаніями, що використовують або не використовують процесно-орієнтований підхід.
- **Рівень процесу.** Процеси, які відбуваються в операційній системі, називаються операційними, процеси в управлінській частині підприємства називаються стратегічними. Стратегічні процеси відбуваються значно рідше.

Функціональна частина процесу Окрім опису процесу та самих даних, які беруть участь у процесі, до процесу ще належить певний набір правил і навіть знань. Оскільки бізнес-правила, які діють у компанії, впливають на бізнес-процес, експертні системи слід відносити до опису бізнес-процесу, оскільки правила не абстрактні, а цілком конкретні, вони мають бізнес-орієнтовану мотивацію і працюють на просторі операційних даних, які беруть участь у процесі.

Рішення будь-якої проблеми, яку можна розв'язати протягом 10–30 хвилин телефоном вашим експертом, може бути розроблено у вигляді експертної системи (Феєрбах).

Для управлінської системи підприємства необхідно приймати рішення в автоматичному режимі на основі тих даних, які зберігаються в процесі, поточних активностях, стану організаційного забезпечення тощо. У більшості випадків спостерігається прогалина в інтеграційній складовій систем управління процесами та експертними системами.

Автори вважають, що простір даних процесу та простір даних правил є частинами однієї сутності — того, що описує процес. Процес — це не лише послідовність переходів, які вказують, за якою схемою мають оброблятися дані, процес — це також набір правил подій і дій, що ініціюються, коли виникають події, а також до процесу (управлінського) належать набір знань або фактів, на основі яких будуються динамічні правила (продукції).

Зазвичай модель, коли виникла подія — потрібно обробити ці дані за цим алгоритмом, називають системою Event Condition Action (ECA). Цю модель ще називають активною або реакційною моделлю [Active DBMS, Reactive Rules].

Друга схема: маючи певний набір правил (фактів), визначити той механізм дій, який потрібно виконати. [Це класичний приклад експертної системи, такої як NxBRE, CLIPS].

Розглянемо ці механізми детальніше.

ECA Як показує досвід, пов'язувати підсистему ECA з підсистемою опису процесів недоцільно. Однак у загальному випадку модель ECA має бути частиною опису процесу. Крім того, правила в ECA можуть динамічно додаватися в систему, тому необхідно подбати про персистентність правил ECA.

Ім'я, для якого визначено список асоційованих із ним обробників подій. Передбачається, що під час виникнення події обробнику буде доступний певний набір даних. З подіями асоційовані обробники, які здебільшого складаються з двох частин: предиката та дії. Одна й та сама подія може оброблятися багатьма обробниками. Виклик обробників можна спрощено представити як: якщо (умова), то дія. Дії можуть викликати інші події тощо.

Обробник складається з предикативної та діючої частини. Умова — це предикат, при істинному значенні якого виконується дія. Але можуть бути обробники, у яких немає предикативної частини.

Дія — це тіло обробника події — правило проходження, в якому прихована логіка проходження документа, супровідні дії, введення документів на ходу. Саме дія визначає варіанти маршрутів у системі.

Системи на правилах Розглянемо системи, які будують продукції, або, як їх ще називають, експертні системи. В основі експертних систем також лежать схожі правила виду: якщо (умова), то (значення), а не як в ECA: якщо (умова), то (дія).

Системи, засновані на правилах, складаються з таких частин:

- База даних правил.
- База даних фактів.
- Інтерпретатор або машина логічного виведення.

У системах, заснованих на правилах, правила — це база даних зв'язків, які побудовані на просторі фактів. База даних фактів — це набір об'єктів, над якими застосовуються правила-предикати та будуються умовиводи або продукції. Завдання машини логічного виведення — знайти шлях із правил над простором фактів, який приведе до потрібного результату.

Існує кілька підходів для продукування умовиводів (або ланцюжків умовиводів), але зазвичай застосовується дедукція. Дедукція — це метод, який реалізується так: виходячи з набору вихідних даних, отримується цільовий умовивід. Його ще називають прямим логічним виведенням. Альтернативний (зворотний) метод — це зворотне логічне виведення, коли береться мета і шукається набір продукцій, які необхідно застосувати, щоб її досягти.

У певному сенсі прямий логічний вивід схожий на застосування реактивних правил. Тільки в експертних системах застосування правил обмежено метою, яку необхідно отримати, а реактивні правила самі по собі являють мету, яку потрібно виконати.

Важливий момент під час побудови систем, заснованих на правилах, — розв'язання колізій, коли існує кілька шляхів логічного виведення. Правила, які визначають, як будуть розв'язуватися конфлікти, називаються мета-правилами.

Алгоритм Rete Потенційна проблема, яка виникає в експертних системах, — це кількість порівнянь, які необхідно зробити між фактами та правилами. Алгоритм Rete — це ефективний метод розв'язання цієї проблеми, який використовується в більшості експертних систем.

Rete — це спрямований, ациклічний, кореневий граф, або, як його ще називають, пошукове дерево. Кожен шлях від кореня до листя в дереві представляє ліву частину правила. Кожен вузол зберігає, який факт було обрано в цій ланці ланцюжка правил. Зі зміною фактів нові факти поширюються в дереві від кореня до листя, змінюючи інформацію, що зберігається у вузлах. Використовуючи це, система повинна тестувати лише ті правила, які стосуються нових фактів, замість того, щоб перевіряти кожен факт для кожного правила.

Більш детальну інформацію про пристрої та алгоритми в експертних системах див. у [Rules and Expert Systems].

2.3 Технічна область

2.4 Історія систем управління процесами

Щоб зрозуміти, що таке системи управління процесами сьогодні, потрібно знати, якими вони були вчора.

Системи автоматизації офісу Перша система автоматизації офісу з'явилася ще на платформі Xerox PARC наприкінці 70-х. Вона називалася Officetalk, тоді вже використовувалися віконні форми графічного середовища для опису завдань процесу. Визначення процесів мало потужну підтримку з боку лінгвістичного забезпечення — процеси задавалися Information Control Networks (ICN) — похідними від мереж Петрі. Основна ідея роботи — спільна робота в офісі з використанням примітивів INBOX, OUTBOX і форм обробки завдань на базі власного (першого у світі) графічного інтерфейсу.

Сьогодні останньою існуючою системою автоматизації офісу є Lotus Domino Server, яка досі користується великою популярністю в середніх і малих компаніях із простою структурою.

Інші технології

- **Системи документообігу.** Системи документообігу — це системи, що працюють паралельно системам управління процесами, використовуються для зберігання повідомлень, документів або ж для зберігання самої системи управління процесами.

- **Електронна пошта.** Інтернет-пошта є базовим механізмом нотифікації як у системах управління процесами, так і в інших системах. Сама пошта, поштові програми можуть бути організовані

для обробки процесів. Листи — це повідомлення, відповідь на лист — це виконана робота.

- **Бази даних.** Бази даних і засоби аналітичної обробки OLAP є перпендикулярними до систем документообігу. Це більш низькорівневий спосіб зберігання даних. Для систем управління процесами найбільш цікаві такі частини БД, як управління транзакціями та активні бази даних, у яких можна писати правила. З боку активних баз даних системи управління процесами можуть реалізовуватися у вигляді правил, тригерів в активних базах. Послідовність активностей може бути реалізована як виклик ланцюжка правил і спрацьовування нових правил. Ця область досліджень дуже цікава в контексті Workflow-систем.

2.5 Системи управління процесами (Workflow Systems)

Бум систем управління процесами припав на початок 90-х років. За одними оцінками, зростання доходів від реалізації систем управління процесами лише у 2003 році зросло на 30%. Системи застосування продуктів на базі систем управління процесами різноманітні.

По-перше, в альянсі з системами документообігу системи управління процесами зайняли нішу, яка довго залишалася незайнятою: це автоматизація розподіленого офісного праці.

По-друге, це більш досконалі системи на підприємствах. Вони розширили існуючі ERP-системи додатковими модулями, внесли новий рівень масштабованості в сервери застосунків. Наприклад, сьогодні SAP NetWeaver, інфраструктурний модуль, містить три компоненти автоматизації процесів: крос-компонентний BPM для інтеграції із зовнішніми системами, Business Workflow для автоматизації робочих місць людей у рамках системи та Collaborative Workflow для автоматизації неструктурованих процесів взаємодії людей.

Якщо розглядати середні компанії, тут максимальна віддача від Workflow-систем спостерігається в галузі автоматизації паперового діловодства.

Визначення Відповідно до визначень у другому параграфі, ми визначили потік робіт як засіб опису процесу, який визначає формальний механізм опису взаємодії та координації між активностями, застосунками, учасниками процесу. Цей механізм може контролюватися інформаційною системою, яка називається системою управління процесами.

Відповідно до стандарту WfMC, система управління процесами визначається як система, яка визначає, створює та управляє процесами, їхнім типом і екземплярами за допомогою набору програмного забезпечення, що базується на одному або кількох двигунах, які здатні

інтерпретувати процес, взаємодіяти з учасниками процесу, а також із застосунками.

Модель процесу, яка задається потоком робіт, визначається WfMC так: представлення бізнес-процесу у формі, яка підтримує автоматизовані маніпуляції, такі як моделювання або розширення з боку системи. Модель представляє собою мережу активностей, їхніх взаємовідносин, прямого або непрямого критерію, який дозволяє визначити початок і кінець процесу, а також набір даних у вигляді даних, визначених програмами або учасниками процесу.

Ці визначення звужують клас процесів, які можна описати. Ці процеси називають стан-орієнтованими, де ключову роль відіграють активності, до яких прив'язані учасники процесу, а не набір бізнес-правил. Стан-орієнтоване описання процесу підходить для повністю автоматичних систем, таких як операційна система підприємства. У другому параграфі цієї глави наводилися інші підходи для опису процесу, які особливо важливі для управлінської системи підприємства.

Анатомія системи З яких компонентів зазвичай складаються системи управління процесами. Ядро системи управління процесами як об'єкт системи являє собою сервіс, що реалізує наступний інтерфейс:

- Отримати список завдань із черги для поточної активності. Завдання являють собою повідомлення. Їх отримують учасники процесу.
- Завершити обробку завдання. Користувач може скасувати завдання, перенаправити його іншому користувачу системи, зберегти оброблену заявку в системі для подальшого проходження завдання тощо.
- Користувач може породити новий процес, який залучить у своє виконання виконання нових завдань.

Практично всі системи управління процесами, за винятком чисто реактивних систем, таких як Skelta, реалізують цей інтерфейс черги повідомлень для активності.

- Управління користувачами.
- Управління аудиторською інформацією.
- Підтримка дизайнера процесів.

Інші важливі аспекти представлені в наступному розділі, при формуванні вимог і цілей конкретної системи.

Вбудовувані та автономні системи Залежно від виробничого контексту, роль систем управління процесами може бути прихованою або неголовною. Системне рішення може бути побудовано так, що система управління процесами є центральною самостійною системою, або навпаки, коли система управління процесами — це сполучна ланка для інших систем підприємства, і зв'язок між існуючими інформаційними системами.

Рис. 2.4: Самостійна система управління процесами

Місце для рисунка: Самостійна система управління процесами

Більшість систем управління процесами 80-х і 90-х років були самостійними або автономними. Вбудовувані ж системи більше потрібні там, де вже є якась інфраструктура, наприклад, ERP-системи. За допомогою вбудовуваної системи можна розв'язати більшість інтеграційних та інших проблем переходу до процесно-орієнтованого підходу.

Класичний приклад впровадження — коли на підприємствах, де є ERP, CRM і системи управління контентом, система управління процесами використовується для координації існуючих застосунків, формалізації робочих місць, управління потоком повідомлень (документами).

Рис. 2.5: Вбудовування як засіб інтеграції

Місце для рисунка: Вбудовування як засіб інтеграції

Далі перейдемо до розгляду стандартів, які є на ринку систем управління процесами, які вже згадувалися в цьому розділі.

Стандарти

Workflow Management Coalition У цьому розділі розглядається головним чином стандарт WfMC, а також зачіпається теорія мереж Петрі, яка слугує основою для State-Based Workflow систем. Консорціум WfMC публікує не один стандарт, а набір стандартів, які стосуються різних аспектів побудови Workflow-систем. Усі стандарти іменуються інтерфейсами. Існує п'ять інтерфейсів:

- **Інтерфейс 1** відповідає за стандарт опису бізнес-процесу. Фактично він визначає мову опису, яка називається XPDL і є підмножиною XML.
- **Інтерфейси 2/3** визначають стандарт для інтерфейсу прикладного програмування, доступу до об'єктів, якими оперує система.
- **Інтерфейс 4** визначає рівні сумісності та взаємодії з іншими системами, які повинні підтримувати системи управління процесами.

- **Інтерфейс 5** визначає набір системних індикаторів, які стандартно мають бути в будь-якій системі управління процесами.

Нижче ми коротко розглянемо, що являє собою набір стандартів WfMC, хочемо лише сказати, що за більш детальною інформацією слід звернутися до джерел [1, 2, 3].

Інтерфейс 1 XPDL визначає, із чого і як будується та описується процес. Основні об'єкти, які беруть участь в описі: Packages, Process, Activity, Transition, Participants, Resources, Application, Data Field, Environment.

Інтерфейси 2/3 Інтерфейси 2 і 3 визначають інтерфейс прикладного програмування як набір COM/CORBA-об'єктів, так і C API.

Інтерфейс 4 Інтерфейс 4 визначає рівні сумісності та взаємодії з зовнішніми системами управління процесами:

- Рівень 1 — Без взаємодії.
- Рівень 2 — Співіснування.
- Рівень 3 — Унікальні шлюзи.
- Рівень 4 — Обмежена спільна підмножина API.
- Рівень 5 — Повний Workflow API.
- Рівень 6 — Спільні формати визначень.
- Рівень 7 — Сумісність протоколів.
- Рівень 8 — Утилітарний вигляд і відчуття.

Інтерфейс 5 Інтерфейс 5 визначає набір індикаторів, який фактично описує рівень контролю, можливого на системному рівні. Передбачається, що набір цих індикаторів буде розширюватися виробниками систем:

- Create/Start Process/Subprocess Instance Audit Data
- Change Process Instance State Audit Data
- Change Activity Instance State Audit Data
- Assign Activity Instance Attributes Audit Data
- Change WorkItem State Audit Data
- Assign/Reassign WorkItem Audit Data
- Assign Work Item Attribute Audit Data
- Source Workflow Engine/Request/Operation/Response Audit Data
- Target Workflow Engine/Request/Operation/Response Audit Data

Object Management Group На додаток до WfMC, OMG адресує частину того, що стосується систем управління процесами. OMG була заснована в 1989 році низкою провідних компаній із розробки програмного забезпечення і була націлена на формалізацію принципів побудови програмного забезпечення. До її складу входять три комітети:

- Комітет із базових стандартів (Platform Technology).
- Комітет зі стандартизації в галузі певних видів бізнесу або доменів (Domain Technology).
- Головний координаційний комітет.

В основному OMG зосереджена на CORBA-орієнтованих стандартах. Вона визначає сервіси, які реалізуються в системах управління підприємствами.

Веб-сервіси Стандартизація протоколів нижніх рівнів взаємодії зовнішнього/внутрішнього в системах управління процесами охоплює такі галузі, як кодування, визначення маршрутів і транспортування даних (HTTP, SOAP, XML, WSDL). Увесь цей набір технологій зараз просувається під лейблом «веб-сервіси» основними гравцями на ринку ПЗ.

Протягом 90-х основний акцент робився на стандартизацію всередині компаній, зараз він змістився на взаємодію між організаціями. Системи, які постійно перебувають в онлайні, надають доступ до своїх даних і дають можливість проводити транзакції або інші види взаємодії зі своїми даними, уже почали називати сервісами. Основний відкритий протокол, за яким передаються дані, — текстовий, тому сервіси називаються веб-сервісами.

Рис. 2.6: Міжорганізаційна взаємодія

Місце для рисунка: Міжорганізаційна взаємодія

Стандарти, які будуть описані в цьому параграфі, стосуються саме формату міжорганізаційних повідомлень.

Класифікація мов опису процесів

Мови опису процесів умовно можна поділити на графічно-описові та текстово-описові. Графічно-описові мови задаються графами XML, які описують ці графи, зазвичай ці мови застосовуються в бізнес-сфері, щоб аналітику було зручно описувати процес. Наглядне відображення процесу в графічній формі...

Розділ 3

Проєктування

3.1 Модель процесів

Під моделлю даних тут розумітимемо все, що стосується саме управління процесом як сутністю. Функціональна частина описана в моделі подій. Як відбувається робота з системою, описано в моделі сервісів.

3.2 Базові поняття

Існує низка моделей [Сети Петри, Комерційні стандарти опису процесів], які описують процесно-орієнтоване виробництво. Багато з них уже давно стандартизовано, але, тим не менш, досі не існує повної в математичному плані системи. Останньою спробою формалізації моделей опису процесів є методологія YAWL [http://is.tm.tue.nl], де зібрано досвід проєктування процесно-орієнтованих застосунків.

Усі існуючі системи так чи інакше є підмножиною над множиною систем, які можуть бути задані мережами Петрі. Однак використовувати мережі Петрі [http://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions] незручно, тому ми опишемо набір визначень [An alternative way to analyze Workflow Graphs] і закономірностей (теорем [Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques]) у нашій математичній моделі, які базуються на мережах Петрі та є математично-лінгвістичним апаратом для визначення процесів.

Під час конструювання синтаксису мови визначення процесів автори систем вільно використовують той чи інший синтаксис, зручний для їхнього власного розпізнавання. Синтаксис, який використовуємо ми, жодним чином не претендує на наш стандарт. Ба більше, необхідна ізоляція рівня опису процесу від самої системи, щоб мати можливість працювати з різними стандартами опису процесів на одному рівні. Це, по-перше, дасть змогу абстрагуватися від систем і допоможе на стадії інтеграції з іншими системами, де процеси описуються інакше. По-друге, це вирішить

проблему імпорту-експорту описів процесів, оскільки для системи не буде визначеного стандарту-фаворита.

Цей підхід у складних системах часто називається мікроядерним. На базі деякої узагальненої моделі будуються інші моделі та реалізуються обгортки для вже наявних моделей. Прикладом є організація захищених підсистем у мікроядерній архітектурі Windows NT [Inside Windows NT]. Уперше така модель була реалізована в мікроядерній операційній системі Mach [Mach].

Автори під час розробки моделей визначення процесів стикалися з проблемою імпорту/експорту та взаємодії з іншими підсистемами опису процесу. Саме мікроядерний підхід є найдоцільнішим під час реалізації прошарку, який відповідає за визначення процесу, де різні синтаксиси опису процесу реалізуються як драйвери.

Процес Процес являє собою ініційовану системою послідовність проводок, у межах якої елементарними сутностями є повідомлення. З процесом асоціюється контекст даних — набір документів, які беруть участь у процесі, та оточення процесу, яке може містити як набір змінних, так і набір правил.

Процес поділяється на дві частини: клас або тип, який описує певний клас процесів (наприклад, відвантаження товару чи реєстрація земельної власності), і друга частина — це екземпляр процесу певного класу, коли, використовуючи певний тип процесу, фактично ініціюється реальний процес із реальними даними, де дані рухатимуться відповідно до опису процесу, на основі якого створено екземпляр.

Для кожної системи існує свій спосіб завдання процесу. Тому для підтримки синтаксису певної системи необхідно постачати драйвер, який може розпізнавати мову завдання процесу цієї системи. Цей драйвер повинен використовувати мета-мову описів процесу, яка в загальному випадку має бути спільною для всіх процесно-орієнтованих систем. Фактично для всіх структурних систем такою мовою є мережа Петрі, яка в нетривіальних випадках досить громіздка.

Повідомлення Повідомлення є пакетом даних, якими маніпулюють правила. Повідомлення — це елементарна сутність, над якою виконується маніпуляція людиною. Дані, які містить повідомлення, завжди перебувають у контексті певного запущеного процесу. Фактично система нічого не знає про дані, які в ній зберігаються.

Рис. 3.1: Опис процесу та його екземпляри

Місце для рисунка: Опис процесу та його екземпляри

Дані в системі можуть бути активними, наприклад, впливати на логіку системи за допомогою активних елементів — правил, що інтегруються в

систему, які представлені у вигляді скриптів. Навіть у статичних стан-орієнтованих системах для прийняття того чи іншого рішення необхідно орієнтуватися на дані, які зберігаються в повідомленні.

У системах, заснованих на фактах-правилах (Пролог, системи зворотного відстеження) [NxBRE, CLIPS], самі повідомлення є банком знань, на основі якого будуються шляхи проходження повідомлень. У будь-якому випадку повідомлення завжди потрібно розглядати в контексті функцій, які виконують операції над повідомленнями.

Активності Активності — це те, що пов'язує модель даних із моделлю опису процесу. Активність або стан пов'язує учасника процесу (користувача-агента) і включає його в модель маршрутів і обчислень шляхів, роблячи систему активною. Зверніть увагу, що стани можуть бути як статичними, так і динамічними, вони можуть обчислюватися імперативно (за послідовністю проходження правил у стан-орієнтованих системах), а також зворотним шляхом (якщо будувати шлях на основі набору знань).

Набір класів станів описується мета-мовою опису процесів, яка в загальному випадку має підходити для всіх стан-орієнтованих систем.

Перехід Перехід або Transition — це елемент, який пов'язує два стани. У статичних системах переходи — це частина опису процесу, крім того, під час модифікації екземпляра процесу переходи можуть реорганізовуватися в екземплярах. У динамічних системах переходи можуть створюватися динамічно, що фактично стирає межу між описом процесу та його екземпляром.

У факт-орієнтованих системах поняття переходу відсутнє, оскільки переходи обчислюються динамічно або неявно описані в ECA-правилах. Послідовність переходів становить шлях, виведений на наборі правил, які застосовуються над областю даних, що містяться в повідомленнях.

Патерни

Існує цілий набір класів активностей. Наприклад, активності, де потік даних розщеплюється, або активності, у яких, навпаки, потік даних сходиться, виконуються логічні операції над умовами, які застосовуються до даних у потоці тощо. Більшість стандартів опису процесів розрізняються саме за потужністю набору типів активностей.

Як було сказано у вступі до цього розділу, усі мови опису процесів у своїй основі так чи інакше використовують мережі Петрі, тому логічно було б будувати системи на базі мови, заснованої на мережах Петрі, яка не звужує їхньої потужності.

Мережі Петрі та патерни Значна робота з удосконалення принципів опису процесів була проведена в Університеті технологій Ейндховена,

Нідерланди. За основу бралися всі існуючі мови опису, а також стандарт WfMC. Оскільки в основі будь-яких процесів лежать мережі Петрі, було поглиблено вивчено зв'язок цих понять і проведено роботу з формалізації математичного забезпечення визначення процесів. В університеті ведеться робота над створенням академічної системи управління процесами YAWL.

Мова опису YAWL складається з набору патернів активностей, на яких будуються процеси. Усього 21 активність:

- **Базові патерни управління:**
  - Sequence
  - Parallel Split
  - Synchronization
  - Exclusive Choice
  - Simple Merge
- **Розширені розгалуження та синхронізація:**
  - Multiple Choice
  - Synchronizing Merge
  - Multiple Merge
  - Discriminator
  - N-out-of-M Join
- **Структурні патерни:**
  - Arbitrary Cycles
  - Implicit Termination
- **Патерни, що залучають кілька екземплярів процесів:**
  - MI without synchronization
  - MI with a priori known design time knowledge
  - MI with a priori known runtime knowledge
  - MI with no a priori runtime knowledge
- **Стан-орієнтовані патерни:**
  - Deferred Choice
  - Interleaved Parallel Routing
  - Milestone
- **Патерни скасування:**
  - Cancel Activity
  - Cancel Case

Визначення Мережі Петрі. Мережа Петрі — це трійка P, T, F:

1. $P = \{p_1, p_2, \ldots, p_n\}$ — набір станів.
2. $T = \{t_1, t_2, \ldots, t_n\}$ — набір переходів.
3. $F \subseteq P \times T \cup T \times P$ — набір ділянок, де пов'язуються стани та переходи.
4. $P \cup T \neq \emptyset, P \cap T = \emptyset$. Використовуємо символіку $\bullet t$ для позначення всіх станів, що входять до даного переходу, а також $\bullet p$ для позначення всіх вхідних переходів для стану p. Аналогічно $t\bullet, p\bullet$.

Мережа потоків. Мережа Петрі $\{P, T, F\}$ називається мережею потоків (Workflow Net) тоді і тільки тоді, коли:

1. Мережа Петрі має два спеціальні стани (вхідний i та вихідний o), причому у вхідного немає вхідних переходів, а у вихідного — вихідних.
2. Кожен елемент мережі перебуває на шляху між i та o.

Звучність. Опис процесу, змодельованого за допомогою мережі Петрі $\{P, T, F\}$, є звучним тоді і тільки тоді, коли:

1. Із будь-якого стану M, до якого можна потрапити зі стану i, можна потрапити до стану o.
2. Стан o — це єдиний стан, до якого можна потрапити зі стану i принаймні з одним маркером.
3. У мережі немає мертвих переходів.

Звучність мережі описує коректність моделі.

Простий послідовний перехід.

$$Seq_{branch}(P, T, I, O) = [P_0][t_k P_k]^*,$$

де $P = \{p_1, p_2, \ldots, p_n\}$ — набір станів, $T = \{t_1, t_2, \ldots, t_n\}$ — набір переходів. Для будь-якого i від 0 до $n-1$, $I(p_i, t_{i+1}) = 1$, і для будь-якого i від 1 до n, $O(p_i, t_i) = 0$. Для всіх інших $I(p_i, t_j) = 0$ та $O(p_i, t_j) = 0$. $P_0(t_1)$ називається початковим станом (переходом), а $p_n(t_n)$ — кінцевим станом (переходом).

Простий умовний перехід. Ця сутність утворена шляхом поєднання кількох простих послідовних переходів, які ділять початковий і кінцевий стан:

$$Cond_{branch}(P, T, I, O) = [p_s] \left[\bigcup_{k=1}^{m} t_{k,0} Seq_{branch}(P_k, T_k, I_k, O_k) t_{k,n_k} \right] [p_e]$$

Простий паралельний перехід. Ця сутність утворена шляхом поєднання кількох простих послідовних переходів, які ділять одні й ті самі початкові та кінцеві переходи:

$$Par_{branch}(P,T,I,O) = [p_s][t_s]\left[\bigcup_{k=1}^{m} t_{k,0} Seq_{branch}(P_k,T_k,I_k,O_k) t_{k,n_k}\right][t_e][p_e]$$

Порівняння з іншими мовами Зазвичай у роботах із патернами, які застосовуються в системах управління процесами, наводять порівняння патернової мови опису з існуючими до цього мовами та стандартами. Як видно, далеко не всі наявні засоби опису процесів мають повний набір примітивів.

Патерн	XPDL	BP4	BPML	XLNG	WSFL	WSCI	
1 (seq)	+	+	+	+	+	+	
2 (par-spl)	+	+	+	+	+	+	
3 (synch)	+	+	+	+	+	+	
4 (ex-ch)	+	+	+	+	+	+	
5 (simple-m)	+	+	+	+	+	+	
6 (m-choice)	+	-	+	-	-	+	-
7 (sync-m)	+	-	+	-	-	+	-
8 (multi-m)	-	-	-	+/-	-	-	+/-
9 (disc)	-	-	-	-	-	-	-
10 (arb-c)	+	-	-	-	-	-	-
11 (impl-t)	+	-	+	+	-	+	+
12 (mi-no-s)	+	-	+	+	+	+	+
13 (mi-dt)	+	+	+	+	+	+	+
14 (mi-rt)	-	+	-	-	-	-	-
15 (mi-no)	-	-	-	-	-	-	-
16 (def-c)	-	+	+	+	+	-	+
17 (int-par)	-	-	+/-	-	-	-	-
18 (milest)	-	-	-	-	-	-	-
19 (can-a)	-	+	+	+	+	+	+
20 (can-c)	-	+	+	+	+	+	+

3.3 Модель подій

Це так звана ECA — Event Condition Action модель [ECA Rule Integration into an OODBMS]. Ця модель є незалежним елементом загальної системи, який є ортогональним до моделі workflow, що будується на похідній моделі від мереж Петрі.

Як показує досвід, пов'язувати підсистему ECA з підсистемою опису процесів недоцільно. Однак у загальному випадку модель ECA має бути частиною опису процесу [A Framework for Optimizing Distributed Workflow Executions]. Крім того, правила в ECA можуть динамічно додаватися в систему, тому необхідно подбати про персистентність правил ECA.

Прикладами використання ECA-правил під час моделювання процесів є [Event-based Distributed Workflow Execution with EVE, XDoC-WFMS: A Framework for Document Centric Workflow Management System].

Подія Ім'я, для якого визначено список асоційованих із ним обробників подій. Передбачається, що під час виникнення події обробнику буде доступний певний набір даних.

Обробник події З подіями асоційовані обробники, які здебільшого складаються з двох частин: предиката та дії. Одна й та сама подія може оброблятися багатьма обробниками. Виклик обробників можна спрощено представити як: якщо (умова), то {дія}. Дії можуть викликати інші події тощо.

Умова Обробник складається з предикативної та діючої частини. Умова — це предикат, при істинному значенні якого виконується дія. Але можуть бути обробники, у яких немає предикативної частини.

Дія Дія — це тіло обробника події — правило проходження, у якому прихована логіка проходження документа, супровідні дії, введення документів на ходу. Саме дія визначає варіанти маршрутів у системі.

Застосування Можуть розглядатися різні варіанти, як саме використовувати ECA-модель у системах управління процесами. Один із підходів, запропонований Андрієм Пилипцем, полягає в маршрутизації повідомлень замість конструювання процесу на мережах Петрі.

Рис. 3.2: Реалізація маршрутизації повідомлень за допомогою ECA-моделі

Місце для рисунка: Реалізація маршрутизації повідомлень за допомогою ECA-моделі

Розкажемо по порядку, як включається модель ECA в опис процесу: поряд із контекстом процесу (тобто даними, асоційованими з ним) у модель процесу включаються функції-правила, які реагують на певні події в системі та можуть втручатися в потік повідомлень, а також взаємодіяти на мета-рівні з системою.

Для опису процесу вже недостатньо накладати бізнес-правила в активності процесу (як це можна робити в багатьох розширених мовах опису процесів), необхідно включити функціональну частину в опис процесу, яка має бути персистентною та входити в екземпляр процесу. Оскільки самі ECA-правила містять необхідну інформацію для переходу від активності до активності, потреба в моделюванні процесу на основі мереж Петрі відпадає.

Прикладами таких систем можуть бути [xTier, Skelta]. Також можна розглядати такий варіант: залишити модель опису процесу на базі мереж Петрі як базовий скелет опису процесу і підтримувати також ті самі ECA-правила, не зменшуючи їхньої потужності. Це дасть змогу розділити процес на дві частини — структурну та функціональну. Опис структурної частини забезпечить можливість інтеграції з класичними структурними описами процесів.

3.4 Модель сервісу

Ядро системи управління процесами як об'єкт системи являє собою сервіс, що реалізує наступний інтерфейс:

Отримання завдання Отримати список завдань із черги для поточної активності. Завдання являють собою повідомлення. Ніде не сказано, який саме вигляд мають повідомлення. Загалом це серіалізовані об'єкти невідомої структури, які можуть бути розпізнані (або не розпізнані) певними модулями підтримки даних.

Завершення завдання Завершити обробку завдання. Користувач може скасувати завдання, перенаправити його іншому користувачу системи, зберегти оброблену заявку в системі для подальшого проходження завдання тощо. Весь спектр ми називаємо завершенням завдання.

Редагування завдання Редагування завдань. Наприклад, немає потреби в отриманні завдання та зміні його керуючої атрибутики: активності чи поточного учасника. Необхідно просто змінити версію даних, внести зміни в дані, для цього система управління процесами використовується як система зберігання повідомлень.

Ініціація екземпляра процесу Користувач може породити новий процес, який залучить у своє виконання виконання нових завдань.

Це був описаний базовий інтерфейс. Але існує ще інтерфейс управління базою даних процесів:

Моделювання типу процесу Для функціонування системи в ній має бути принаймні один процес, для якого є опис. Як би не задавався процес, має бути можливість його завдання в системі.

Управління учасниками процесу До цієї частини входить визначення набору користувачів-учасників процесу та застосунків, асоційованих із ними.

3.5 Методи управління потоком

Подієвий підхід

З точки зору користувача, йому доступний набір завдань, призначених для нього. Активуючи та отримуючи конкретне завдання, він виконує над ним дію та відправляє його назад у систему. Перед вибором завдання та перед відправленням назад у систему система автоматично активує ланцюжок правил, які

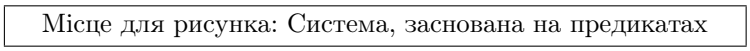
Рис. 3.3: Імперативний документообіг

виконуються в контексті певного процесу та певного завдання. Після цього система передає завдання іншим користувачам, яких необхідно пройти відповідно до заданої моделі.

3.6 Система, заснована на предикатах

З точки зору документообігу, система визначається набором описових правил, які вказують, що потрібно для досягнення певних цілей. Під час обчислення кінцевої мети система вираховує шлях правил, який потрібно пройти для її досягнення, дорогою відправляючи повідомлення користувачу для введення інформації, підпису, генерації резолюцій.

Рис. 3.4: Система, заснована на предикатах



3.7 Модель безпосередніх переходів

Граф проходження документів у такій моделі задається статично, у ньому заздалегідь вказуються всі можливі переходи в системі від стану до стану. Це класичний підхід до опису процесів на основі мереж Петрі. WfMC описує саме цей підхід до створення workflow-систем. Усі продукти, описані в розділі 5, належать до цього класу систем, за винятком, хіба що, Skelta, яка спочатку орієнтована на executional business process model (як це зараз називають).

3.8 Змішана система

Система, заснована на правилах, активованих подіями, і система розв'язання загальних завдань являють собою діаметрально протилежні підходи до побудови шляху проводки документа.

У разі прямого проходження завдання система бере на себе функції маршрутизатора напрямків до інших користувачів, даючи змогу останнім поповнювати систему необхідною інформацією для подальшого проходження.

У разі предикативного проходження система має можливість вирахувати шлях проходження завдання від користувача до користувача, як це робиться в логічних системах, таких як PROLOG, GPS-системах.

3.9 Вимоги та цілі

Вимоги до системи сховища

- **Підтримка багатьох цільових СУБД.** Система сховища має забезпечувати можливість взаємодії з основними СУБД, що виступають сховищами документів.
- **Динамічне визначення класів документів.** Документ як об'єкт системи документообігу описує структуру зберігання та відображення, які мають динамічно створюватися адміністраторами системи.
- **Комплексність даних.** Документи мають підтримувати основні типи даних поряд зі складними типами, як-от масиви, зображення, файли, складні об'єкти (карти, схеми, плани), зовнішні щодо цієї системи дані.
- **Ієрархічна структура документів.** Об'єкт документа часто може мати дуже складну структуру, наприклад, файл, папку або навіть цілу полицю (набір томів).
- **Каталогізація документів.** Документ може бути віднесений до певної категорії на основі його класу, даних або рішення людини. Введення «папки проєкту» також є елементом каталогізації. Документ може належати до кількох категорій одночасно.

Вимоги до системи управління процесами

- **Послідовності обробки документів.** Послідовність документів у заданому порядку визначає певний бізнес-процес, який має початок і кінець. Стадії, через які проходять документи, безпосередньо пов'язані з робочими місцями користувачів, які їх обробляють.
- **Динамічне визначення бізнес-правил.** Усі бізнес-правила системи мають бути здатними до динамічного визначення та бути виділеними в окремих частинах системи.
- **Робоче місце користувача.** Базове робоче місце має включати мінімум, необхідний для роботи з системою. Крім того, мають

бути визначені специфікації на розширювані модулі системи для постачальників — третіх осіб, а також для динамічної публікації (оновлення) компонентів системи.

Вимоги до підсистеми захисту

- **Підсистема захисту.** Система має підтримувати щонайменше механізм захисту C2, який описується в термінології списків прав доступу ACL. Загалом необхідно підтримувати організаційну модель MAC.

- **Авторизація.** Система має підтримувати різні механізми авторизації користувача (пароль, відбиток пальця, сітківка ока).

- **Електронний підпис.** Для визначення справжності документа в захищеній системі необхідно підтримувати відомі механізми електронного підпису даних. Це не дасть змоги особі, яка має доступ до документа (наприклад, адміністратору системи), змінити документ, що не відповідає сфері впливу її посадових повноважень.

- **Захист від несанкціонованого використання.** Обмеження на кількість одночасно використовуваних клієнтських застосунків, кількість класів документів та інші обмеження мають регулюватися сервером ліцензій.

Публікація даних

- **Розвинена система звітів.** Розвинена система електронного документообігу має забезпечувати набір механізмів для відображення статистичної інформації та поточної інформації про стан системи. Сюди входять генератори звітів для друку, системи публічного доступу до даних.

- **Зовнішній доступ до системи.** Необхідно забезпечити доступ до системи з будь-якої точки планети, використовуючи будь-який пристрій.

Адміністрування

- **Дизайнер форм.** Визначені складні класи документів, редагування яких у базовій схемі нетривіальне, мають включати визначені користувачем редактори та правила відображення.

- **Адміністратор системи.** Розширене робоче місце користувача-адміністратора має включати механізми створення класів документів, визначення послідовностей проходження документів, дизайнер форм, розширені статистичні звіти.

- **Архівування.** Механізм резервного копіювання має визначатися часовим зрізом, форматом резервних файлів. Крім того, сюди входить як імпорт-експорт усієї бази (або її частини), так і окремих документів або набору документів.

Інші вимоги

- **Багатомовна підтримка.** Система має підтримувати лівосторонні, правосторонні мовні схеми, ієрогліфічне та алфавітне письмо. Мови мають встановлюватися опціонально, прозоро для системи. Усі символьні дані в системі мають бути закодовані в UNICODE.

- **Набір компонентів для програмування.** Для власних потреб клієнтів система має надавати бібліотеки для програмування, які забезпечують доступ до ключових функцій системи. Крім того, деякі частини системи (контрольні елементи інтерфейсу користувача) мають дистриб'ютися окремо.

Цілі

Підсистеми або модулі документообігу в складі автоматизованих систем мають відповідати таким вимогам:

Завдання моделі роботи підприємства Встановлюючи та конфігуруючи систему документообігу на великих підприємствах, неможливо повністю заздалегідь визначити модель її функціонування після впровадження системи документообігу з урахуванням реінжинірингу.

Під час створення автоматизованих систем зазвичай спочатку проводять детальний аналіз роботи підприємства з урахуванням таких методологій аналізу та проєктування, як IDEF, SADT, UML. Визначаються сценарії та варіанти використання системи, виділяється набір робочих місць, описуються шляхи проходження документа, набір бізнес-правил, за якими працює підприємство.

Монолітні системи, побудовані таким чином, дуже чутливі до різного роду змін, внесених після впровадження системи, які зазвичай зачіпають усі архітектурні аспекти: зміна сценаріїв роботи, зміна робочих місць, зміна набору бізнес-правил.

Таким чином, система документообігу не повинна залежати від конкретної предметної області, як-от, наприклад, торговельні підприємства, державні установи, виробничі чи наукові підприємства.

У системах такого класу (SAP/R3) сценарій роботи підприємства задається під час конфігурації системи, описується модель його роботи, використовуючи базові примітиви відомих методологій аналізу та проєктування.

Окреме визначення бізнес-правил Один із головних принципів побудови вдалих автоматизованих систем — це відокремлення набору бізнес-правил від програмного забезпечення. Статично спроєктована система, коли логіку роботи підприємства неможливо змінити, завжди пов'язана з нерозв'язними проблемами. Під час розробки будь-яких автоматизованих систем, у яких модель роботи підприємства наперед визначена, намагаються створити власне лінгвістичне забезпечення для опису бізнес-правил (1С Бухгалтерія).

Гнучкість щодо сховища даних Абстракція від сховища даних, тобто можливість одночасного використання кількох засобів для зберігання інформаційного забезпечення, відіграє першочергову роль у розширюваності та гнучкості системи.

Практично всі сучасні автоматизовані системи можуть зберігати дані на серверах від різних виробників, які часто дуже сильно відрізняються. Ба більше, існує кілька типів СУБД, які мають принципові відмінності в підходах до опису та зберігання даних: реляційні (Oracle, MS SQL), об'єктні (Cache) тощо. Абстракція від типу зберігання даних — ключ до добре спроєктованої системи. Це реалізується за допомогою виділеного сервера застосунків, який обслуговує зберігання даних.

Можливість зміни моделі в процесі роботи Часто в процесі роботи підприємства виникають складнощі під час зміни моделі його роботи, що гальмує розвиток підприємства під час переходу на якісно вищі рівні роботи. Задана модель роботи, визначена на етапі впровадження автоматизованої системи, має бути здатною динамічно змінюватися.

Масштабованість Масштабованість передбачає роботу кількох серверів документообігу в межах одного підприємства. Територіально розподілені підприємства потребують підтримки роботи кількох пов'язаних систем документообігу, що підтримують реплікацію даних, взаємодію та залежності робочих місць, розташованих у різних системах. Крім того, можливість взаємодіяти із зовнішніми щодо цієї системи документообігу системами, наприклад, 1С, Парус тощо.

Відмовостійкість Відмовостійкість визначає механізми, що запобігають втраті даних у системі. Насамперед, це транзакційне виконання елементарних операцій у системі документообігу під час переходу документів із однієї стадії на іншу.

Розділ 4

Реалізація

4.1 Сервер застосунків

Хост Завданням хоста є створення оточення для модулів системи. Він має працювати як служба Windows і як звичайний консольний застосунок.

Сервер Сервер виконує всю роботу з завантаження та вивантаження плагінів, створення та заповнення базового оточення.

Модулі Усі компоненти системи є плагінами, які можна динамічно завантажувати та вивантажувати з хоста. За це відповідає сервер, який надає передбачені для цього сервіси. Як модулі можуть бути організовані зовнішні компоненти для взаємодії з іншими системами або зовнішнім середовищем.

Користувачі Один модуль є спільним для системи зберігання та системи управління процесами — це модуль управління користувачами — UserManager, який являє собою підсистему управління захистом документів і розмежуванням доступу до активностей процесів за допомогою списків прав і політик, керованих адміністратором системи.

У межах сервера застосунків працюють як система сховища, так і система управління процесами.

Сховище

Як було зазначено, сховище є невід'ємною частиною системи управління процесами, яка забезпечує персистентне зберігання повідомлень — інформаційних потоків, а також самих описів процесів і екземплярів процесів. Фактично без потужної системи сховища немає сенсу реалізовувати систему управління процесами. За основу в нашій системі управління процесами було взято систему зберігання документів компанії ILS — DSS.

Джерела даних

- **Вертикальна база даних.** Типи полів, які можуть бути записані в базі даних, зберігаються в окремих таблицях. Під час створення нового документа до нього приєднуються дані з таблиць — сховищ типів. Завдяки такій моделі можна на ходу створювати нові типи документів. Цю модель даних ще називають вертикальною або об'єктною.
- **Традиційна плоска модель даних.** Традиційна реляційна модель — коли для зміни структури документа виконується команда ALTER TABLE.
- **XML-сховище на файловій системі.** Усі документи зберігаються на файловій системі у вигляді XML-файлів. Крім того, XML-сховище може бути проміжним, якщо потрібно конвертувати одне сховище в інше. Також, працюючи в офлайн-режимі відносно сервера даних, документи можна тимчасово зберігати в локальному XML-сховищі.

Доступ до послуг сервера системи сканування документів здійснюється за допомогою об'єктного API, який значною мірою базується на архітектурній моделі системи.

Архітектура сховища

Основне завдання системи сканування документів — слугувати засобом збору/реєстрації інформації та виступати сховищем. Система може використовуватися як самостійний продукт, але також може бути складовою іншої, більш спеціалізованої системи.

Система зберігання документів ILS підтримує зберігання та обробку настроюваної атрибутики з розвиненим набором типів полів (включаючи такі типи, як файли, посилання на документи, масиви), автоматичне індексування документів із застосуванням розпізнавання зображень і повнотекстового пошуку. Система дає змогу зберігати та обробляти кілька версій документа одночасно.

Сервіс сховища У контексті сервера застосунків працюють такі модулі, як об'єкт — сховище документів — Storage, що надає безпосередній доступ до документів і реалізує такі сервіси, як додавання, видалення, модифікація документів, пошук за документами, отримання метаінформації. Це один із головних модулів.

Сервіс метаопису сховища Модуль StorageDesigner дає змогу модифікувати метаінформацію. Метаінформація — це набір класів документів, властивостей класів, режимів роботи класів, типів властивостей тощо. Агент StorageDesigner дозволяє створювати нові класи, додавати властивості до класів, визначати режими роботи з класами.

Документи Інтерфейс (IDocument) реалізує основний механізм роботи з документами, тобто підтримку версій, установлення атрибутів-властивостей документа, як скалярних, так і векторних, відкат змін (Undo), збереження та підтвердження змін (Save, Commit), видалення документа (Delete).

Класи Основоположним поняттям у системі документів є поняття класу документа. Клас документа — це набір полів або властивостей, притаманний цьому класу документа. Цей набір полів (клас) являє собою фізичні атрибути документа, які реально зберігаються в сховищі даних.

Форми Поняття режиму роботи (форми) є додатковим, але не менш фундаментальним поняттям стосовно класу документа. Це набір полів документа (з визначеними правами доступу), їхня послідовність у відображенні, назви самих полів, а також додаткові атрибути, що обмежують використання документів, як-от права доступу або обмеження на читання/запис/створення документів.

Доступ до документа завжди здійснюється на основі режиму роботи. Це вказується в методі CreateDocument другим параметром. Не маючи режиму роботи для цього класу документа або не маючи до нього доступу, ви фактично не зможете працювати з конкретним документом.

Типи даних Властивості документів, як і їхні відображення, які підтримуються, можна дізнатися за допомогою StorageDesigner. Окрім простих типів даних, таких як Int, String, Date, Money, є складні типи даних.

Табл. 4.1: Типи даних

Прості	Складні
Int	File
String	Image
Money	Object
Date	Vector
Enum	

Типи даних File та Image являють собою бінарні образи змінної довжини. Тип Vector являє собою розріджений масив, який може містити як елементи простих типів, так і елементи типів File, Image та Object. Тип вектор векторів системою не підтримується. Тип Object являє собою посилання на документ, що дає змогу пов'язувати документи підпорядкованими двосторонніми зв'язками.

Пошук Підтримка пошуку здійснюється за допомогою інтерфейсу ICondition. Метод класу Storage NewSearchCondition повертає об'єкт, що

підтримує інтерфейс ICondition, за допомогою якого можна конструювати певні критерії пошуку. Далі, задавши всі необхідні критерії пошуку, потрібно викликати функцію Search, передавши їй сконструйований об'єкт.

4.2 Користувацький інтерфейс

Бізнес-об'єкт — документ Бізнес-об'єкт являє собою документ, що зберігається на сервері документів. Документ складається з полів, які мають певний тип, зареєстрований у системі. У конфігурації за замовчуванням є такі типи даних: ціле значення, дійсне значення, рядок, булеве значення, об'єкт, список-словник, файл, зображення, грошове значення, дата. Крім того, у кожного поля, окрім визначення його типу, є низка інших атрибутів: ознака ключового поля, ознака векторного поля.

Редактори Для кожного зареєстрованого типу даних існує свій редактор, який дає змогу редагувати цей тип. Редактор складається з набору контрольних елементів, які доступні менеджеру компонування для розміщення.

Реєстрація нових редакторів Можна створити новий тип даних у системі, визначивши для нього власний редактор даних, а також перевизначити редактори даних для наявних типів даних.

Менеджер компонування Для кожного класу документа можна визначити кілька розташувань редакторів за допомогою скрипта. Скрипт компонування пов'язує редактори з формою, іншими словами, він визначає розташування редакторів на формі.

Форми Існує два класи форм — форми редагування або перегляду та форми пошуку, що дозволяють розв'язувати залежності (зв'язки типу документ-документ). Форма для типу документа — це те саме, що й редактор для типу даних. Оскільки дані (поля) групуються в документи, необхідно їхнє цілісне відображення — саме за це відповідають форми документів.

Системні форми — це форми, контрольні елементи яких розташовуються за допомогою скрипта менеджера компонування. Користувацькі форми являють собою панелі зі статично розташованими контрольними елементами. І ті, й інші форми зберігаються в бібліотеках, що розділяються. Але в системних формах ви можете керувати компонуванням під час виконання.

Реєстрація нових форм Так само, як і у випадку з редакторами, ви можете визначити власні форми для доступу до певних класів документів. Крім того, для нових класів ви можете використовувати власні форми або форми за замовчуванням.

4.3 Ядро — Система управління процесами

Рис. 4.1: Редагування процесу

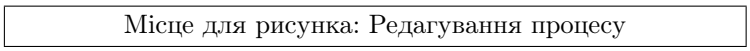


Рис. 4.2: Список процесів

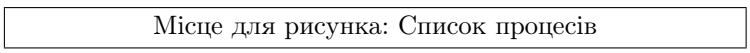


Рис. 4.3: Класи активностей

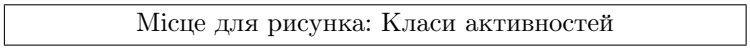


Рис. 4.4: Список переходів

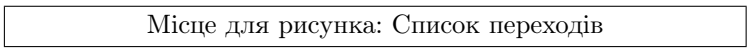


Рис. 4.5: Список екземплярів активностей

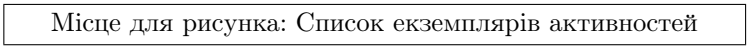


Рис. 4.6: Розблокування процесів

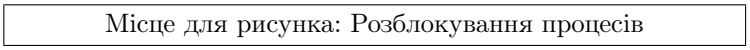


Рис. 4.7: Управління учасниками процесу

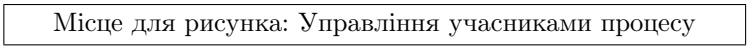


Рис. 4.8: Управління системою каталогу та словників

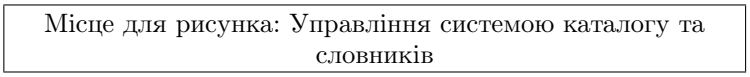


Рис. 4.9: Метадані інструментів

Рис. 4.10: Головна черга повідомлень для учасника

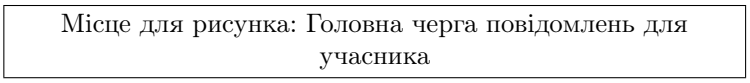


Рис. 4.11: Публічний доступ

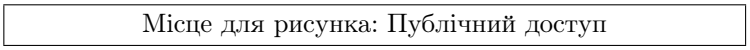


Рис. 4.12: Автоматично згенерований редактор для повідомлення типу «MO»

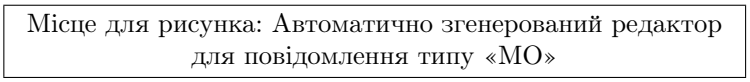


Рис. 4.13: Автоматично згенерований редактор для повідомлення типу «Заявник»

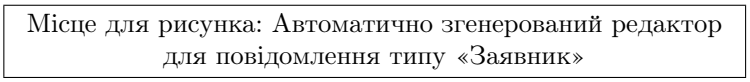
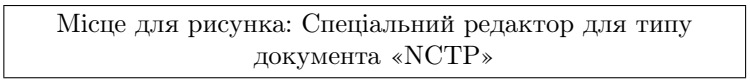


Рис. 4.14: Спеціальний редактор для типу документа «NCTP»



Розділ 5

Випробування та аналіз

5.1 Порівняння з існуючими системами

OpenWFE

Опис Серед продуктів із відкритим кодом ми вирішили розпочати огляд із JAVA-орієнтованої системи OpenWFE (`www.openwfe.org`). Для її розгортання не потрібні J2EE, сервери застосунків чи ORB-брокери. Крім того, вона надає бібліотеку доступу для .NET.

OpenWFE згадується з кількох причин: по-перше, це єдина система, яка врахувала зауваження [18] і реалізувала повний набір примітивів workflow. У наступних версіях планується реалізація імпорту/експорту XPDL. По-друге, вона досить проста і зрозуміла, тому може слугувати початковим посібником для систем подібного типу.

Підсистема опису процесів Як набір примітивів використовуються 20 примітивів, опис яких можна знайти на сайті `http://www.workflowpatterns.com`. Підтримка XPDL лише планується, але не можна забувати, що, як показано в пункті 1.6, патерни значно багатші за набір примітивів XPDL, тому реалізація підтримки XPDL не становить великої складності.

- **Ступінь відповідності примітивам із workflow patterns:** Усі
- **Модульність форматів:** Планується
- **Підтримка XPDL:** Ні
- **Підтримка різних систем зберігання:** Так (XML, MySQL, PostgreSQL)

Інтерфейс прикладного програмування Під час проєктування API автори системи не орієнтувалися на жоден стандарт, описаний WfMC.

Єдиною вимогою були простота, наочність і очевидність. У OpenWFE немає можливості взаємодії з іншими системами за допомогою загальноприйнятих стандартів. Система підтримує внутрішній текстовий протокол REST.

- **Підтримка базових примітивів:** Так
- **Підтримка інтерфейсів 2/3:** Ні
- **Ступінь відкритості:** Підтримує модулі (reactors)

Підсистема зберігання повідомлень Як система зберігання може використовуватися MySQL, PostgreSQL або звичайні XML-файли. Найстабільнішою підсистемою зберігання є XML Storage. Щодо структури повідомлень (об'єктів, які передаються від активності до активності), то тут можуть виступати будь-які об'єкти, сконструйовані з такого набору атрибутів, як основні скалярні типи, хеш-таблиці, списки, файли.

Додаткові можливості Існують дві бібліотеки зовнішнього доступу: Python і .NET. Як додаткові засоби, у цьому продукті приділено увагу модулям-агентам, які взаємодіють із сервером в автоматичному режимі (reactors). Ці модулі можна розробляти на .NET або Python.

Підсистема захисту Підсистема захисту являє собою спрощену систему користувачів і прав. Підтримка груп відсутня.

Клієнтські компоненти До комплекту постачання входять кілька утиліт:

- Повнофункціональний веб-клієнт,
- Консольний клієнт для перегляду сховищ, повідомлень і процесів, запущених у системі,
- Адміністратор, де встановлюються права та політики для користувачів.

Рис. 5.1: Редагування повідомлення (зліва) разом із історією (на правому плані)

Місце для рисунка: Редагування повідомлення

Рис. 5.2: Список завдань

Місце для рисунка: Список завдань

Рис. 5.3: Адміністратор системи

Місце для рисунка: Адміністратор системи

Рис. 5.4: Проєктування процесу

Місце для рисунка: Проєктування процесу

Висновок Загалом дотримання стандарту WfMC під час розробки цього продукту не було головною метою. Розробники в основному орієнтувалися на інтуїтивно зрозумілу систему класів, а під час розробки формату зберігання опису процесу використовувалися останні досягнення в цій галузі. Належну увагу приділено підсистемі захисту. API досить просте. Є бібліотека доступу для .NET із прикладами. Система поширюється у відкритих вихідних текстах, доступних на `www.openwfe.org`. Інтерфейси 4 і 5 не підтримуються і не плануються. Цей проєкт має навчально-ознайомлювальний характер із досить потужною функціональністю для початку реалізації більш функціональної системи.

Skelta

Опис Про комерційний продукт Skelta можна дізнатися на форумі `http://www.workflow-research.de/Forums`, який зазначено як джерело додаткової інформації на `www.wfmc.org`. Як зазначено в [20], існує два підходи до розробки workflow-систем: state-based systems і executional business processes. WfMC описує стан-орієнтований підхід до опису систем. Skelta використовує принципово іншу методологію проєктування, тому огляд цієї системи в контексті відповідності стандарту WfMC не зовсім коректний. Тут ми спочатку виділимо спільні частини та проведемо аналіз функціональної цілісності предметної області workflow-систем, а саме передачі повідомлень від активностей до активностей, процесів, ланцюжків проходження документа.

Зовні як state-based системи, так і executional business системи виглядають однаково. Тому для Skelta огляд проводитиметься в іншому ключі. Skelta — це веб-базована система, орієнтована на інтеграцію з іншими ASP.NET-системами. Окрім бібліотеки управління workflow, до складу продукту входить набір веб-контролів, які дозволяють:

- Переглядати список завдань, призначених користувачу,
- Вводити дані за вказаною задачею,
- Створювати нові workflow і керувати наявними.

Усі маніпуляції здійснюються через вікно інтернет-браузера. Також Skelta забезпечує функціональність для сповіщення користувача через різні засоби зв'язку: електронну пошту, месенджери тощо.

Skelta орієнтована на модель Executional Process. Це означає, що екземпляри workflow (workflow instance або Process) є програмними процесами, які складаються з Actions і виконуються на машині, де встановлена Skelta. Action — елемент програмної логіки, який виконується в наданому оточенні та реалізує необхідний програмний інтерфейс (`Workflow.NET.Engine.Interfaces.IActionRun`). Skelta дозволяє писати action на мовах .NET або VBScript .NET/JScript .NET.

Сервіси, які надає Skelta:

- Як WebService із деякими функціями управління workflow (Execute, Abort, Pause, Resume, AlertEngine), але без функціональності для отримання інформації про наявні workflow,
- Як бібліотека з розширеною функціональністю,
- Веб-контроли та класи для UI, які малюють workflow-діаграми (можуть використовуватися в GUI),
- Додаткові засоби взаємодії з електронною поштою,
- Набір базових Actions: Notification, Rule, Seek Approval, E-Mail Notification.

Обмеження Слід зазначити, що Skelta орієнтована виключно на використання в веб. Документація дуже слабка. Action'и запускаються на стороні сервера, тому в межах розподіленої системи потрібне додаткове створення механізмів взаємодії з Action. При цьому під час взаємодії на весь час роботи Activity будуть зайняті серверні ресурси, що може бути критично. Відмінний полігон для демонстрації інтеграційних можливостей .NET-інфраструктури: навіть таку систему можна інтегрувати з будь-чим, якщо добре попрацювати на C#!

Підсистема визначення процесів Система класів, які надають доступ до примітивів, за допомогою яких конструюється бізнес-процес, будується з використанням веб-клієнта. Документація на формат збереженого процесу відсутня. Персистентні (збережені) процеси зберігаються безпосередньо в реляційному форматі, який також не задокументовано.

- **Ступінь відповідності примітивам із workflow patterns:** Hi
- **Модульність форматів:** Hi
- **Підтримка XPDL:** Hi
- **Підтримка різних систем зберігання:** Так (Access, MS SQL)

Інтерфейс прикладного програмування Точки входу в обробники Action'ів задаються як посилання на відповідні Assemblies безпосередньо в XML-файлах. Сама система класів досить громіздка і неінтуїтивна. Вивчення системи може зайняти значний час. Значна частина класів системи орієнтована на розробку в ASP.NET-середовищі для веб.

- **Підтримка базових примітивів:** Ні
- **Підтримка інтерфейсів 2/3:** Ні
- **Ступінь відкритості:** Низька, нічого не задокументовано

Підсистема зберігання повідомлень Як система зберігання підтримуються лише Microsoft Access (Jet DB) і MS SQL.

Клієнтські компоненти До комплекту постачання входить набір ASPX-сторінок. Для майбутніх застосунків надається бібліотека, яка є досить низькорівневим шаром.

Додаткові можливості Як додаткові можливості можна відзначити різноманітні сповіщення (електронна пошта, сповіщення Action'ів). Крім того, можна писати власні обробники бізнес-правил на C#, компілюючи їх у бібліотеки та підключаючи безпосередньо до описів процесів, які в термінології Skelta називаються конфігураційними файлами.

Висновок Підбиваючи підсумки для такого неоднозначного продукту, зазначимо наступне. Призначення цієї системи — розробка веб-порталів із використанням технології ASP.NET. Skelta являє собою зачатки системи сповіщень, подібних до тих, що є в цілому класі Rule-Based систем (ECA). Але обробники подій не мають можливості персистентного зберігання, як у класичних ECA-системах. Крім того, сама бібліотека являє собою досить низький рівень workflow engine. Таким чином, для створення системи на базі Skelta доведеться розробляти додаткові інтерфейси та формати зберігання даних, не кажучи вже про засоби клієнтського доступу.

Незважаючи на значний обсяг системи класів, як універсальний засіб обробки послідовностей документів Skelta — невдалий вибір, оскільки орієнтована на роботу в оточенні набору ASPX-сторінок, спрощуючи взаємодію між ними. Якщо ж такі особливості системи, як орієнтація на обробники подій і асоціації з Action'ами, дійсно потрібні системному інтегратору workflow-системи, ми рекомендуємо розглянути класичні Rule-Based системи workflow. Крім того, рекомендуємо звернутися до базової теорії ECA-систем.

Shark

Опис Shark — ще один представник продуктів із відкритим кодом. Написаний із використанням JAVA. Веб-сторінка проєкту: `http://shark.objectweb.org/`. У Shark заявлена підтримка основних інтерфейсів WfMC. Є повна підтримка XPDL (Interface 1) і AWSP (Interface 4), а також WfMC API. До складу системи входить CORBA Server. Управління робочими групами здійснюється через LDAP.

Архітектура

- **Клієнтські компоненти:**
  - CORBA Swing Admin/WorkList
  - POJO Client
- **Публічні компоненти:**
  - CORBA API
  - WfMC Instance API
  - Admin
- **Компоненти ядра:**
  - Event and Audit API
  - Authentication API
  - Transaction API
  - Process Locking
  - Logging API
  - Caching API
  - Security API
  - Integration API (Wf-XML 2.0)

Підсистема визначення процесів Як базовий формат використовується XPDL для опису бізнес-процесів. Модульність форматів не підтримується.

- **Ступінь відповідності примітивам із workflow patterns:** Hi
- **Модульність форматів:** Hi
- **Підтримка XPDL:** Так
- **Підтримка різних систем зберігання:** Так

Інтерфейс прикладного програмування Автори системи пішли далі стандартів WfMC. До складу інтерфейсу програмування входить різноманітна підтримка сповіщень тощо. За основу взято стандарт OMG, описаний у [23]. Основні класи для роботи клієнта виглядають так:

- WfActivityIterator
- WfAssignment
- WfAssignmentEventAudit
- WfAssignmentIterator
- WfCreateProcessEventAudit
- WfDataEventAudit
- WfEventAudit
- WfEventAuditIterator
- WfExecutionObject
- WfProcess
- WfProcessIterator
- WfProcessMgr
- WfRequester
- WfResource
- WfStateEventAudit

Інтерфейси ядра:

- CallbackUtilities
- ObjectFactory
- PersistenceInterface
- ToolAgentManager
- WfActivityInternal
- WfAssignmentEventAuditInternal
- WfAssignmentInternal
- WfCreateProcessEventAuditInternal
- WfDataEventAuditInternal

- WfEventAuditInternal
- WfExecutionObjectInternal
- WfProcessInternal
- WfProcessMgrInternal
- WfRequesterInternal
- WfResourceInternal
- WfStateEventAuditInternal

Таким чином, можна зазначити, що підтримка інтерфейсів 2/3 формально не дотримана, але загалом API досить схожий. З урахуванням підтримки інтерфейсів 1 і 4 можна сказати, що автори намагалися максимально відповідати вимогам WfMC. Питання формальної відповідності стандартам залишаються відкритими.

- **Підтримка базових примітивів:** Так
- **Підтримка інтерфейсів 2/3:** Ні, частково
- **Ступінь відкритості:** Висока

Підсистема зберігання повідомлень Як системи зберігання можуть використовуватися такі сховища:

- DB/2
- HypersonicSQL
- Informix
- MS SQL
- MySQL
- Oracle
- PostgreSQL
- Sybase

Клієнтські компоненти До комплекту постачання входить Java Swing-застосунок для роботи з WorkList. Для роботи з XPDL передбачається використання зовнішнього графічного редактора XPDL-файлів JaWE (`jawe.objectweb.org`), який входить до складу сімейства Enhydra.

Рис. 5.5: Створення процесу

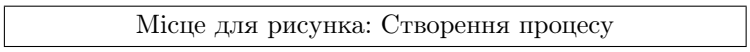
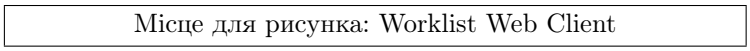


Рис. 5.6: Swing CORBA Client/Admin

Рис. 5.7: Worklist Web Client



Висновок Безумовно, головною метою під час розробки Shark було дотримання стандартів WfMC. Ба більше, ніша цього продукту — розподілені гетерогенні системи, де використовуються CORBA-сервіси. Для великих підприємств, де вже є спеціалісти з розгортання застосунків, що використовують CORBA, і які хочуть спробувати досить велику, потужну систему у вихідних кодах, Shark — вдалий вибір. Окрім WfMC, керівники проєкту орієнтуються також на стандарти інших організацій, таких як OMG (Object Management Group — `www.omg.org`).

Shark входить до складу ширшої системи Enhydra (`www.enhydra.org`), яку розробники порівнюють із Apache, але з урахуванням бізнес-вимог для електронного бізнесу. Також варто звернути увагу на `www.objectweb.org` — тут розташований консорціум, який курує цей проєкт. Список рішень, до складу яких входить workflow engine Shark, представлено на сторінці `solutions.objectweb.org`.

Таким чином, загалом Shark є частиною досить великого проєкту, який має солідних спонсорів. Ідеї, закладені в проєкт, можуть слугувати хорошим прикладом як реалізації, так і архітектури системи. У системі підтримуються всі основні інтерфейси WfMC, включаючи Wf-XML 2.0 (Interface 4).

K2.NET 2003

Опис K2 — єдиний представник на сьогодні state-based workflow-систем, написаних на .NET. Його API повністю сумісне з WfMC, за винятком хіба що перейменованих класів Transition → Line. API дуже доступне, наочне, і це попри дуже добру документацію. Ми вважаємо, що користуватися API буде досить зручно. За допомогою цієї системи можна створювати не лише примітивні процеси, але й основні примітиви, що використовують синхронізацію тощо.

Основним методом для доступу до engine є .NET-бібліотеки. Як об'єкти, що передаються як повідомлення за переходами, використовуються XML-об'єкти, тобто універсальні об'єкти. Для кожної активності ви можете написати власний код на C#, який виконається під час ініціації активності.

Система постачається з повним спектром компонентів: Server, DB, MMC Snap-in, Web Access, Process Designer Studio, який компілює процеси в .NET Assemblies.

Підсистема визначення процесів Як формат опису процесу використовується дещо модифікована версія XPDL, але значно більш функціональна. Оскільки в самому форматі можна зберігати код, написаний на C#, це забезпечує цілісність опису процесу.

- **Ступінь відповідності примітивам із workflow patterns:** Так
- **Модульність форматів:** Ні
- **Підтримка XPDL:** Не зовсім
- **Підтримка різних систем зберігання:** MS SQL

Інтерфейс прикладного програмування API поділяється на дві частини: одна являє собою засоби доступу в Run Time до Worklist і його елементів, інша надає Core-класи, з яких побудована система; ці класи використовуються в основному для проєктування процесів.

- **Підтримка базових примітивів:** Так
- **Підтримка інтерфейсів 2/3:** Так
- **Ступінь відкритості:** Достатньо потужна

Підсистема зберігання повідомлень Усі дані зберігаються на MS SQL сервері. Крім того, значну увагу приділено системі звітності, що вимагається інтерфейсом 5 стандарту WfMC.

Додаткові можливості Заявлена інтеграція з такими продуктами Microsoft:

- Microsoft Exchange Server (як система сповіщення)
- Microsoft Office
- Microsoft SharePoint Portal Server
- Microsoft Content Management Server
- Microsoft BizTalk Server

Захист Безпека на належному рівні.

Висновок Висновки цілком визначені: система красива, функціональна, зручна, відповідає класичному погляду на state-based системи, задовольняє основним вимогам NovaLIS Cadastral Office. Відкритим залишається питання ціни. Як можливий ресурс для інвестування (як виправдання можливо завищеної ціни) можна відзначити широкий спектр інтеграційних можливостей із продуктами Microsoft. Значну увагу приділено системі статистики, а також користувацьким утилітам. Продукт загалом досить завершений, із яким можна негайно розпочати роботу з мінімальними витратами на адаптацію до конкретних поставлених завдань.

x-Tier

Опис xTier — ще один представник Executional Business Process Oriented Workflow-систем. З цієї точки зору програмна модель найбільше нагадує Skelta. Під час створення розробниками була обрана SOA (Service Oriented Architecture).

Підсистема опису процесів Сам процес виконання являє собою ланцюжки правил, що виконуються, які задаються в XML-файлі. Кожне правило викликає відповідну збірку (.NET assembly). Набором таких правил у XML конфігурується процес.

- **Ступінь відповідності примітивам із workflow patterns:** Hi
- **Модульність форматів:** Hi
- **Підтримка XPDL:** Hi
- **Підтримка різних систем зберігання:** Oracle

Інтерфейс прикладного програмування Окрім API, що стосується Workflow-сервісу, у повній поставці продукту є ще набір сервісів: Cluster, Metadata, Workflow, Configuration, Cache, Email, JNDI, JMX, Logging, Jobs, Internationalization, License, Object Pool, Transaction.

- **Підтримка базових примітивів:** Hi
- **Підтримка інтерфейсів 2/3:** Hi
- **Ступінь відкритості:** Так, може запускати .NET assemblies

Підсистема зберігання повідомлень Завдяки абстрагуванню від даних, які приховуються за реалізаціями правил на C#, зникає потреба використовувати єдину систему зберігання, яка йде в поставці xTier — Object Pool.

Додаткові можливості Окрім Workflow-сервісу, до поставки входять також додаткові сервіси відповідно до ідеології SOA: транзакції, задачі, повідомлення, пул об'єктів, доставка пошти, інтернаціоналізація, кешування.

Захист Систему безпеки користувач повинен розробляти самостійно.

Висновок Система xTier побудована у вигляді мікроядра, на базі якого ви створюватимете свої системи. Сам модуль, який відповідає за Workflow, являє собою досить простий механізм виклику правил, що містяться в попередньо скомпільованих збірках. Інфраструктури розвиненої безпеки немає. Для деяких компонентів може знадобитися встановлення сервера бізнес-застосунків JBoss (JAVA). Сам xTier портований із Java на .NET.

Інші системи

Open Business Engine Open Business Engine (`http://www.openbusinessengine.org`). На стадії розробки, автори орієнтуються на WfMC. Продукт доступний із `www.sourceforge.net` лише через CVS.

NxBRE NxBRE (`http://nxbre.sf.net`) ми наводимо як приклад системи, що не належить до state-based систем. Це класична Rule-Based .NET-система, портована з JAVA, яка реалізує RuleML, а також підтримує власну внутрішню мову, що нагадує Prolog. Цю систему можна віднести до перших інтелектуальних систем прийняття рішень на основі правил, де транспортом можуть слугувати документи, що визначає клас цієї системи — Rule-Based Workflow Systems. Інші відомі системи в цій галузі — це Mandarax (метод резолюцій) і Drools (реалізує алгоритм Rete для пошуку знань).

jBpm jBpm, як і Enhydra Shark, входить до складу ширшої системи і призначений для роботи в середовищі JBoss, JAVA-операційного оточення для електронної комерції.

OfBiz Open For Business Engine (`www.ofbiz.org`). E-Commerce, орієнтований на роботу з клієнтами. Автори орієнтуються на ERP, CRM-моделі. Крім того, це Rule-Based система, що підтримує логічний вивід, тобто метод резолюцій (див. «дискретна математика»).

ЗВІТ

про впровадження результатів дипломної роботи
на здобуття ступеня магістра прикладної математики
Сохацького Максима Еротейовича
у компанії ІЛС – Україна

Результати роботи Максима Сохацького з тематики систем управління процесами та неструктурованою інформацією були використані в низці комерційних проєктів, науково-дослідницькій роботі, а також обговорювалися на відкритих семінарах компанії.

Насамперед це дослідження систем управління процесами, яке проводилося для компанії NovaLIS. Результатом цього проєкту став звіт про порівняльний аналіз існуючих систем управління процесами.

Методи, використані в роботі, а також окремі модулі системи, розроблені Максимом, впроваджено в низці наших продуктів: DSS (Система управління документами), LRS – система реєстрації земельної власності. Інсталяції цих систем охоплюють такі країни, як США, Ямайка, Азербайджан.

Також робота Максима обговорювалася на відкритих семінарах компанії і не є завершеною. Робота відкрита для подальших досліджень.

Права на всі вищезазначені матеріали, публікації, статті, рішення та продукти належать компанії ІЛС-Україна.

З повагою,

І. А. Попів
Генеральний директор