УДК 004.94

В. А. Позняк
Л. Б. Ліщинська

РОЛЬ НЕЧІТКИХ МНОЖИН У ПРОЄКТУВАННІ ІНСТРУМЕНТАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація
Розглянуто важливість використання нечітких множин у проєктуванні інструментального програмного забезпечення. Проаналізовано переваги та можливості моделювання неоднозначних концепцій та вимог користувачів . Запропоновано методологію використання нечітких множин для підвищення ефективності та адаптивності систем проєктування інструментального програмного забезпечення.
Ключові слова: Нечіткі множини, проєктування, інструментальне програмне забезпечення.

Abstract
The importance of using fuzzy sets in the design of tool software is considered. The advantages and possibilities of modeling ambiguous concepts and user requirements are analyzed. A methodology for using fuzzy sets to improve the efficiency and adaptability of tool software design systems is proposed.
Keywords: Fuzzy sets, design, tool software.

Вступ

У сучасному світі розробка програмного забезпечення є важливою складовою технологічного прогресу. Проєктування інструментального програмного забезпечення, що сприяє розробці та управлінню програмами, стає ключовим аспектом цього процесу. У той же час, зростаюча складність завдань розробки вимагає ефективних методів роботи з нечіткістю та неоднозначністю, які часто властиві реальним даним у цій галузі. Застосування нечітких множин у проєктуванні інструментального програмного забезпечення допомагає збалансувати неоднозначність та створити більш гнучкі та адаптивні системи, які можуть ефективно працювати в умовах невизначеності.

Основна частина

Проєктування інструментального програмного забезпечення - це процес розробки програмних інструментів для полегшення та підтримки роботи з іншими програмними продуктами. Це включає в себе створення середовищ розробки, систем управління версіями, засобів тестування та інших інструментів [1].

Нечіткі множини - це математичний інструмент, що дозволяє моделювати нечіткі або неоднозначні концепції, які важко або неможливо точно визначити.

При аналізі вимог користувачів до програмного продукту можуть виникати ситуації, коли вимоги є нечіткими або неоднозначними. Використання нечітких множин дозволяє представити ці вимоги у вигляді ступенів належності до певних категорій чи характеристик.

Для прикладу було обрано ситуацію: розробники вирішують, яку мову програмування обрати для проєктування нового інструменту для обробки даних. На вибір є дві мови: Java та Python.

Щоб розпочати оцінювати мови програмування, було визначено критерії, за якими буде проводитися порівняння:

1. Швидкість розробки (Speed of Development) - час, необхідний для розробки програмного продукту.

2. Продуктивність (Productivity) - кількість функціоналу, який може бути реалізований за одиницю часу.

3. Надійність (Reliability) - ймовірність того, що програмний продукт буде працювати без збоїв.

Для вибору мови програмування було обрано показники:

1. Загальна ефективність (Overall Efficiency) - показник, що враховує не лише окремі критерії, але й їх взаємозв'язок. В контексті прикладу з порівнянням мов програмування Java та Python для проекту обробки даних, цей показник допомагає зрозуміти загальну ефективність кожної мови, враховуючи їхню швидкість розробки, продуктивність та надійність.

2. Ступінь досягнення мети (Degree of Goal Achievement) - величина, яка відображає, наскільки мова програмування відповідає поставленим цілям розробки. У наведеному прикладі, це ступінь, до якого кожна мова задовольняє вимоги проекту обробки даних.

3. Функція корисності (Utility Function) - математична функція, яка визначає, наскільки корисною є мова програмування для конкретного проєкту. Функція корисності оцінює ефективність кожної мови з урахуванням важливих критеріїв, таких як швидкість розробки, продуктивність та надійність. Вона допомагає приймати остаточне обґрунтоване рішення щодо вибору мови програмування для конкретного проєкту, оскільки враховує не лише окремі критерії, а й їх взаємозв'язок [2].

Повне розв’язання прикладу з вибору мови програмування:

Крок 1: Визначення ваг кожного критерію

Нехай ваги кожного критерію такі:

- Швидкість розробки (Speed of Development): wSD = 0.4
- Продуктивність (Productivity): wP = 0.3
- Надійність (Reliability): wR = 0.3

Крок 2: Введення даних

Для Java:

- Швидкість розробки: SDJava = 10 годин
- Продуктивність: PJava = 1500 рядків коду
- Надійність: RJava = 0.9

Для Python:

- Швидкість розробки: SDPython = 8 годин
- Продуктивність: PPython = 2000 рядків коду
- Надійність: RPython = 0.85

Крок 3: Обчислення кожного критерію

- Швидкість розробки: Для Java: SDJava = 10 годин, Для Python: SDPython = 8 годин
- Продуктивність: Для Java: PJava = 1500 рядків коду, Для Python: PPython = 2000 рядків коду
- Надійність: Для Java: 0.9RJava = 0.9, Для Python: RPython = 0.85

Крок 4: Обчислення загальної ефективності (Overall Efficiency)

Для Java: OEJava = wSD×SDJava+wP×PJava+wR×RJava

OEJava = 0.4×10+0.3×1500+0.3×0.9OEJava = 0.4×10+0.3×1500+0.3×0.9

OEJava = 4+450+0.27 = 454.27OEJava = 4+450+0.27 = 454.27

Для Python:

OEPython = wSD×SDPython+wP×PPython+wR×RPython = 0.4×8+0.3×2000+0.3×0.85

OEPython = 0.4×8+0.3×2000+0.3×0.85

OEPython = 3.2+600+0.255 = 603.455

Крок 5: Визначення ступеня досягнення мети (Degree of Goal Achievement)

Припущено, що ступінь досягнення мети для обох мов дорівнює 0.9.

Крок 6: Обчислення функції корисності (Utility Function)

Для Java: UJava = OEJava×DoAJava = 454.27×0.9 = 408.843

Для Python: UPython = OEPython×DoAPython = 603.455×0.9 = 543.1095

Крок 7: Визначення кращої мови програмування

Порівнюючи значення функції корисності, визначено, що мова програмування Python має більшу корисність для цього проєкту обробки даних порівняно з Java. Отже, у даному випадку, мова програмування Python виявилася більш ефективною з точки зору швидкості розробки, продуктивності та надійності.

Висновок

Використання нечітких множин у проєктуванні інструментального програмного забезпечення є потужним інструментом для розв'язання складних проблем та прийняття оптимальних рішень. Нечіткі множини дозволяють враховувати неоднозначність та нечіткість вхідних даних, а також враховувати експертні знання та досвід, що робить процес проєктування більш гнучким та придатним для реальних умов.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is software design ? [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://sea.ucar.edu/best-practices/design
2. Utility Function Definition, Example, and Calculation [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.investopedia.com/ask/answers/072915/what-utility-function-and-how-it-calculated

Позняк Вероніка Андріївна – студентка групи ПІ-23мз, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: hitechnic6740011@gmail.com

Ліщинська Людмила Броніславівна – д-р техн. наук, професор, професор кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: llb@vntu.edu.ua

Pozniak Veronika **–** student of group 1PI-23me, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: hitechnic6740011@gmail.com

Lishchynska Lyudmyla Bronislavivna – Dr. Sc. (Eng.), Full Professor, Professor of Program Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: llb@vntu.edu.ua