УДК 005.004

В.Б. Задоров, О.О. Васильєв, І.В. Дерев'янко

Київський національний університет будівництва і архітектури, Київ

МОВА МОДЕЛЮВАННЯ БУДІВЕЛЬНИХ ТЕХНОЛОГІЙ «AIMO»

Розглянуто передумови створення мови моделювання будівельних технологій «AIMO», основні вимоги щодо її розробки, складові елементи. Наведено опис сутностей, конструктивів, ресурсів, технологій мови. Наведено опис мови в нотації Бекуса-Наура.

Ключові слова: системи підготовки і управління будівництвом; конструктивна ієрархія; технологічна ієрархія; мова програмування і моделювання будівельних технологій AIMO; компілятор; віртуальна машина; середовище розробки; простір імен; сутності; дочірні сутності; конструктиви; ресурси; технології; опис мови в нотації Бекуса-Наура

Рассматриваются предпосылки создания языка моделирования строительных технологий «АИМО», основные требования, предявляемые языку во время его разработки, составляющие элементы. Приведено описание сущностей, конструктивов, ресурсов, технологий языка. Дано описание языка в нотации Бекуса-Наура.

Ключевые слова: системы подготовки и управления строительством; конструктивная иерархия; технологическая иерархия; язык программирования и моделирования строительных технологий АИМО; компилятор; виртуальная машина; среда разработки; пространство имен; сущности; дочерние сущности; конструктивы; ресурсы; технологии; описание языка в нотации Бэкуса-Наура

Are considered prerequisites for the creation of modeling language construction technologies «AIMO», the principal requirements imposed language during its development, the constituent elements. A description of the entity, form factors, resources, and technologies of language. A description of the language in the notation of Backus-Naur form.

Keywords: system of preparation and construction management; structural hierarchy; technological hierarchy; the language of programming and simulation of building technologies AIMO; the compiler; virtual machine; the development environment; namespace; essence; subsidiary entities; frames; resources; technology; description of the language in notation Backus-Naur form

Обґрунтування ідеї створення мови моделювання будівельних технологій «AIMO»

На сьогодні існує широкий вибір інформаційних систем з розв'язанням задач підготовки і управління будівництвом. Ці системи можуть бути умовно розділені на системи з архітектурно-конструктивного моделювання будівель та споруд (ArchiCad, AutoCad, AllPlan, …), а також на системи з організаційно-технологічного та управлінського моделювання процесів створення будівель та споруд (MSProject, RillSoft, SpiderProject, …). Кожен з цих напрямів має чималу історію розвитку, орієнтуючись на потреби ринку і бізнес-інтереси.

Актуальною проблемою завжди було зведення всіх даних про проект воєдино з метою оптимізації процесів аналізу і управління на всіх стадіях життєвого циклу підготовки, створення і експлуатації будівель та споруд. Так, було розроблено ряд програмних продуктів (як доповнень до тих, що існують, так і нових), що дозволяють так чи інакше зв'язувати дані конструктивних і організаційно-технологічних моделей будівельних об'єктів. Основою стиковки таких моделей є перетинання конструктивної і організаційно-технологічної ієрархії. Одним з простих, але ефективних рішень, є зв'язок між вузлом конструктивної ієрархії (як деталізованого, так і високого рівня) і вузлом організаційно-технологічної ієрархії. Наприклад, таким чином може задаватися правило, що певне приміщення буде зведене після закінчення виконання якоїсь роботи. Це рішення може забезпечити побудову системи так званого «4D перегляду», що означає спостереження за станом зведення будівельного об'єкта в часі.

Не дивлячись на велику кількість програмних продуктів, що різною мірою виконують стиковку конструктивної і організаційно-технологічної ієрархії, проблема створення цілісної системи залишається невирішеною. Тут можуть бути виділені такі проблеми:

- Взаємний вплив конструктивної і організаційно-технологічної ієрархій. Так, спочатку технологія і організація процесів визначається структурою, яка має бути зведена, і в той же час структура може модифікуватися залежно від особливостей технології і виникаючих організаційних обставин (як зміни потоку ресурсів, термінів, або просто зміни у фінансуванні).
- Створення (або ж вичленення) блоків конструктивної і організаційно-технологічної складових, що тісно пов'язані, для подальшого використання і збереження у вигляді типового елемента або елемента нормативної бази. Гнучкість багатократного використання без дублювання інформації.
- Створення загальної ресурсної бази, тісно інтегрованої з об'ємами конструктивної і об'ємно-планувальної складових, з можливістю багатократного використання і параметризації.

Всі наведені проблеми мають бути вирішені для всіх етапів підготовки та зведення будівельного об'єкта, проте насамперед це актуально на стадії підготовки. Так, це б дозволило оцінювати такі показники на етапі інвестування:

- вартості і терміни залежно від варійованих властивостей конструктивних моделей будівель чи споруд;
- вартості, терміни, видозмінені характеристики конструктивних моделей залежно від варійованих критеріїв оптимізації, графіків постачань, фінансування на передпроектній фазі життєвого циклу будівель та споруд;
- аналіз вартостей залежно від варійованих джерел ресурсів, що поставляються, та використовуваних нормативних баз.

Виходячи з описаних вище проблем, авторами пропонується мова програмування і моделювання AIMO. В основі AIMO лежить ідея цілісної системи конструктивного і організаційно-технологічного моделювання. Це дозволяє будувати складні логічні залежності між компонентами моделей, і, таким чином, вирішити вказані вище проблеми [2; 3; 5].

Чому саме мова? Однією з основних цілей системи, що розробляється, є максимально можлива гнучкість завдання сумісної логіки моделей будівельних конструкцій і процесів. Будь-які візуальні редактори не можуть забезпечити подібну гнучкість, за виключенням, можливо, систем візуального програмування. Проте не треба виключати наявність візуальних оболонок для мови: для спрощення роботи передбачені візуальні інтерфейси (конструктори) для спрощеного завдання і редагування елементів конструктивних і організаційно-технологічних ієрархій [1; 7].

Мова моделювання: так чи інакше, шляхом кодування формується модель будівлі з точки зору як конструкцій і об'ємно-планувальних рішень, так і технології і організації зведення. Відбувається моделювання вузлів конструкцій і зв'язків між ними дерева організаційно-технологічних процесів.

Мова програмування: на кожному етапі моделювання є можливість формувати логічні правила, формули, цикли для опису зв'язків між елементами моделі і визначення кількісних характеристик.

Альтернативи серед існуючих рішень. З точки зору інструментів наочної області авторами не було виявлено систем, які б дозволяли виконувати комплексне моделювання систем в будівництві для вирішення наведених проблем. Розглядався варіант використання наявних мов програмування загального призначення (наприклад, C#, або C++), проте при цьому зустрічалися такі обмеження:

- система, що розробляється, стає залежною від можливостей використаної мови програмування; таким чином неможливо обмежити синтаксис тими можливостями, які необхідні для вирішення проблеми;
- як наслідок, суттєво зростає мінімальна кваліфікація фахівців, що використовують систему;
- необхідність використання компіляторів мов програмування для роботи системи, і тим самим значне ускладнення процесу;
- багато рішень, використаних в системі, не було реалізовано в мовах програмування загального призначення.

Таким чином подальший пошук шляхів вирішення проблеми цілісного конструктивного і організаційно-технологічного моделювання в будівництві є актуальним.

Тому до основних вимог щодо мови моделювання будівельних технологій «AIMO» під час її розробки, можна віднести:

- можливість формування основних сутностей: структурних і організаційно-технологічних ієрархій об'єктів необмеженої (в розумних межах) глибини;
- опис допоміжних сутностей для структурних і організаційно-технологічних ієрархій (наприклад, ресурсів);
- систематизацію сутностей за технологією «просторів імен», вживаних у мовах програмування, наприклад, C#;
- гнучкість описуваних ієрархій: можливості мінімізації дублювання описуваної логіки шляхом

багатократного використання компонентів вказаних ієрархій з можливістю параметризації;

- підтримку широкого спектру можливостей програмування логіки роботи ієрархій, таких як: варіювання показників структурної ієрархії залежно від вхідних параметрів, що її описують [4; 6]; логіку роботи організаційно-технологічних моделей, що залежить від параметрів структурної ієрархій, яку використовують технологія і організація будівництва;
- підтримки послідовних, паралельних, альтернативних робіт необмеженої вкладеності і додаткових можливостей, тобто потужної системи опису технології і організації будівництва;
- програмування використання ресурсів з врахуванням пріоритетів, альтернатив, а також доступних потоків ресурсів.

Запропонована мова АИМО складається з компілятору, віртуальної машини та візуального середовища розробки.

Компілятор перетворює вихідні тексти (коди) у об'єктну структуру. Видає повідомлення про помилки, якщо такі зустрічаються.

Віртуальна машина, або засіб моделювання, виконує моделювання організаційно-технологічної ієрархії. Має візуальний інтерфейс для задавання вхідних параметрів і перегляду результатів моделювання, а саме:

- задавання параметрів конструктиву, який моделюється;
- опис параметрів використовуваних ресурсів, вартостей, пріоритетів, потоку постачання;
- показ результатів моделювання у вигляді діаграм Ганта, мережевих графіків (наскільки це може бути можливим), графіків використання ресурсів, кошторису.

Віртуальна машина мови може бути замінена зовнішнім засобом моделювання у випадку, якщо потрібна особлива система моделювання. Наприклад, можуть бути застосовані мережі Петрі.

Візуальне середовище розробки (IDE) [8]. Програма, що представляє можливість зручно програмувати на мові AIMO. Після компіляції надає можливість переглядати у вигляді дерева новоутворені ієрархічні структури, переглядати їх опис (атрибути).

Опис мови «AIMO»

Загальні правила

Початковий код визначається групою файлів, які підлягають компіляції. Як правило, вони знаходяться в одній папці або ж в підпапках. Розділення на файли є умовним, тобто всі файли рівноправні і визначають певну частину структури.

Мова чутлива до регістру.

Простори імен

Простір імен (англ. *namespace*) — певна множина, під якою розуміють модель, абстрактне сховище або оточення, створене для логічного групування унікальних ідентифікаторів (тобто імен). Ідентифікатор, визначений в просторі імен, асоціюється з цим простором. Один і той же ідентифікатор може бути незалежно визначений в кількох просторах. Таким чином, значення, пов'язане з ідентифікатором, визначеним в одному просторі імен, може мати (або не мати) таке ж значення, як і такий самий ідентифікатор, визначений в іншому просторі.

Простір імен визначається блоком інструкцій:

```
namespace knauf{
    [ ... ]
},
```

де *knauf*- назва простору імен, *[...]* – його зміст.

У мові AIMO простори можуть бути вкладеними, наприклад:

```
namespace knauf{
namespace rs_karkas{
resource profil_napr_t1
{
    default
    {
        Description = "Профіль направляючий ПН 50/40, ТУ 1121-004-04001508-2003";
        Cost = 240.0;
    }
}
}
}
```

Опис сутностей

Загальний формат опису сутності:

```
Тип_сутностіім'я_сутності
{
inparam тип_вхідного_параметру_1 ім'я_вхідного_параметру_1;

[Опис дочірніх сутностей]
[Опис екземплярів сутностей]
    Тип_змінної_1 ім'я_змінної_1;
    Тип_змінної_2 ім'я_змінної_2;

    default
    {
        ім'я_змінної_1 = значення_за_умовчанням_ім'я_змінної_1;
    }
}
```

У мові є можливість задавання вхідних параметрів для сутностей, змінних (полів), а також їх значень за замовчуванням. Окрім цього, однією з основних можливостей є опис дочірніх сутностей і екземплярів сутностей. Зупинимося на цьому детальніше.

Опис дочірніх сутностей

Наводиться опис дочірньої сутності, формат вищеописаний. Описана сутність буде дочірньою у ієрархії.

Опис екземплярів сутностей

Описує підсутність, яка створюватиметься в той момент, коли створюватиметься поточна сутність. У ієрархії екземплярів сутностей, описувана підсутність буде дочірньою відносно поточної сутності.

```
instance ім'я_сутності
{
param ім'я_параметру = значення_параметру;
}
```

Окрему увагу треба приділити опису параметрів: при створенні екземпляра сутності є можливість задати значення її вхідних параметрів. Як значення можуть застосовуватися вхідні параметри, змінні (поля) поточної сутності.

Опис конструктивів

Виконується згідно з правилами, описаними в загальному описі сутностей.

```
construction gvl
{
    float WallWidth;
    float WallHeight;

    default
    {
            WallWidth = 4.5;
            WallHeight = 6.0;
    }
}
```

Опис ресурсів

За замовчуванням є два поля, які можуть змінюватися користувачем: *Description(string)* – опис ресурсів і *Cost(float)* – вартість за умовчанням.

Наприклад:

```
resource profil_napr_t1
{
    default
    {
            Description = "Профіль направляючий ПН 50/40, ТУ 1121-004-04001508-2003";
            Cost = 240.0;
    }
}
```

Опис технологій

```
technology MainTech
{
inparam gvl TheGvl;
instance UstroistvoKarkasa
{
param TheGvl = TheGvl;
}
default
{
TechType = 1;
}
}
```

За замовчуванням є поле, яке може змінюватися користувачем: *TechType* – тип взаємодії дочірніх робіт: послідовний, паралельний чи альтернативний.

Всі дочірні екземпляри сутностей технологій вважаються дочірніми роботами.

Блок *resourceusage*у опису технології використовується для задавання використовуваних роботою ресурсів. Наприклад,

```
resourceusage
{
    //By square
    SetResourceUnitCount(TheGvl.WallWidth * TheGvl.WallHeight);
    //Materials
    if (TheGvl.WallWidth <= 4.0)
    {
            rs_karkas.profil_napr_t1 = 1.51;
            rs_karkas.profil_stoy = 2.04;
            rs_karkas.lenta1 = 1.26;
    }
    else if (TheGvl.WallWidth <= 6)
    {
            rs_karkas.profil_napr_t1 = 1.58;
            rs_karkas.profil_stoy = 2.54;
            rs_karkas.lenta1 = 1.17;
            rs_karkas.ugolok1 = 0.14;
    }
    else if (TheGvl.WallWidth <= 9)
    {
            rs_karkas.profil_napr_t1 = 1.61;
            rs_karkas.profil_stoy = 2.6;
            rs_karkas.lenta1 = 1.07;
            rs_karkas.ugolok1 = 0.2;
    }
    else if (TheGvl.WallWidth <= 16.2)
    {
            rs_karkas.profil_napr_t1 = 1.59;
            rs_karkas.profil_stoy = 2.44;
            rs_karkas.profil_napr_t2 = 0.15;
            rs_karkas.lenta1 = 0.94;
            rs_karkas.ugolok1 = 0.16;
    }
    //L'udi
    SetResourceOr(2);
    SetResourceOrStart(0);
    rs_rabochiye.rabochiy_r3 = 0.41;
    SetResourceOrStart(1);
    rs_rabochiye.rabochiy_r4 = 0.21;

    SetResourceOrEnd();
}
```

У наведеному коді слід зупинитися на деяких конструкціях.

- Функція *Setresourceunitcount* дозволяє задати кількість одиниць для підрахунку використовуваних ресурсів. Наприклад, у метрах квадратних або метрах погонних.
- Конструкція *rs_karkas.profil_napr_t1 = 1.51;* визначає, що ресурс *profil* з простору імен *rs_karkas*буде використаний в кількості 1.51 на одиницю виміру.
- *Setresourceor, Setresourceorstart, Setresourceorend.* Функції для задавання альтернатив у використанні ресурсів. Перша функція повідомляє систему, що далі будуть перелічені альтернативи; друга викликається перед задаванням кожної з альтернатив, і остання – коли перелічення альтернатив завершено.

Опис граматики в нотації Бекуса - Наура

Одним з найпоширеніших способів опису формальних мов є опис в нотації Бекуса – Наура (БНФ). Нотація БНФ є набором «продукцій», кожна з яких відповідає зразку:

<символ> ::= <вираз, що містить символи>, де вираз, що містить символи - це послідовність або послідовності символів, розділених вертикальною рискою |, що повністю перелічують можливий вибір символів з лівої частини формули.

Далі:

< — лівий обмежувач виразу;

> — правий обмежувач виразу;

::= — визначене як;

| — або;

{…} – повторення блоку від 0 до N разів;

[…] – блок може бути відсутнім або присутнім лише раз.

Ці символи є символами метамови, вони не визначені у мові, котру описують. Решта описаних символів належать до «абетки» описуваної мови.

Далі наведено опис мови AIMO в нотації Бекуса – Наура:

<Програма> ::= <Простір_імен>

<Простір_імен> ::=
<Namespace><Ідентифікатор><Початок_блок>
{<Простір_імен> | <Конструкція> |
<Технологія> | <Ресурс>} <Кінець_блок>

<Конструкція> ::= <Construction>
<Ідентифікатор><Початок_блок>
{<Конструкція> | <Змінна> |
<Вхідний_параметр> | <Екземпляр> }
[<Блок_за_умовчанням>] <Кінець_блок>

<Технологія> ::= <Technology>
<Ідентифікатор><Початок_блок>
[<Блок_використання_ресурсів>]
{<Технологія> | <Вхідний_параметр> |
<Екземпляр>} [<Блок_за_умовчанням>]
<Кінець_блок>

<Ресурс> ::= <Resource><Ідентифікатор>
<Початок_блок><Блок_за_умовчанням>
<Кінець_блок>

<Блок_використання_ресурсів> ::=
<Resourceusage><Блок_код>

<Екземпляр> ::=
<Instance><Ідентифікатор_типу>[
<Ідентифікатор>] [<Кінець_виразу>|
<Блок_код>]

<Блок_за_умовчанням> ::= <Default>
<Блок_код>

<Блок_код> ::= <Початок_блок> {
<Блок_IFELSE> |
<Блок_FOR> | <Блок_WHILE> | <Вираз_кін> }
<Кінець_блок>

<Блок_IF> ::= <If><Дужка_поч><Вираз>
<Дужка_кін><Блок_код>

<Блок_ELSE> :: <Else><Блок_IFELSE>|
<Блок_код>

<Блок_IFELSE> ::= <Блок_IF>
[<Блок_ELSE>]

<Блок_FOR> ::= <For><Дужка_поч>
<Вираз_кін><Вираз_кін><Вираз>
<Дужка_кін><Блок_код>

<Блок_WHILE> ::= <While><Дужка_поч>
<Вираз><Дужка_кін><Блок_код>

<Вираз> ::= [<Param>]
<Ідентифікатор_змінної><Унарна_Операція>
| [<Param>] <Ідентифікатор_змінної>
<Бінарна_Операція><Вираз>| [<Дужка_поч>]
<Ідентифікатор_змінної>|<Функція>|
<Строка><Бінарна_Операція><Вираз>
[<Дужка_кін>] | <Функція>

<Функція> ::= Ідентифікатор><Дужка_поч>
<Вираз><Дужка_кін>

<Строка> ::= Строка_поч><Символи>
<Строка_поч>

<Вираз_кін> ::= <Вираз><Кінець_виразу>

<Змінна> ::= <Тип_Змінної>
<Ідентифікатор_змінної><Кінець_виразу>

<Вхідний_параметр> ::= <Inparam><Змінна>

<Тип_Змінної> ::= <Ідентифікатор> |
<Тип_строка> | <Тип_ціле> | <Тип_дійсне>

<Ідентифікатор_змінної> ::=
<Ідентифікатор>[<Початок_масив>
<Число_без_знак><Кінець_масив>]

<Ідентифікатор_типу> ::=<Ідентифікатор> |{
<Ідентифікатор><Крапка>
<Ідентифікатор_типу>}

<Число_без_знак> := <Цифра>|<Цифра>
<Число_без_знак>

<Ідентифікатор> ::= <Лат_Буква> {
<Лат_Буква> | <Цифра><Кінець_масив> }

<Унарна_Операція> ::= ++|--|+|-|!

<Бінарна_Операція> ::= =|==|!=|>|>=|<|<=|+|-|*|/|**

<Символи> ::= {<Лат_Буква> | <Крапка> |
<Цифра> | <Спец_символи> }

<Спец_символи> ::= ~|!|@|#|$|%|^|&|*|(|)|_|+|=|-|||\|/|?|.|>|<|,|{|}| |:|;|'

<Лат_Буква> ::=
a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|

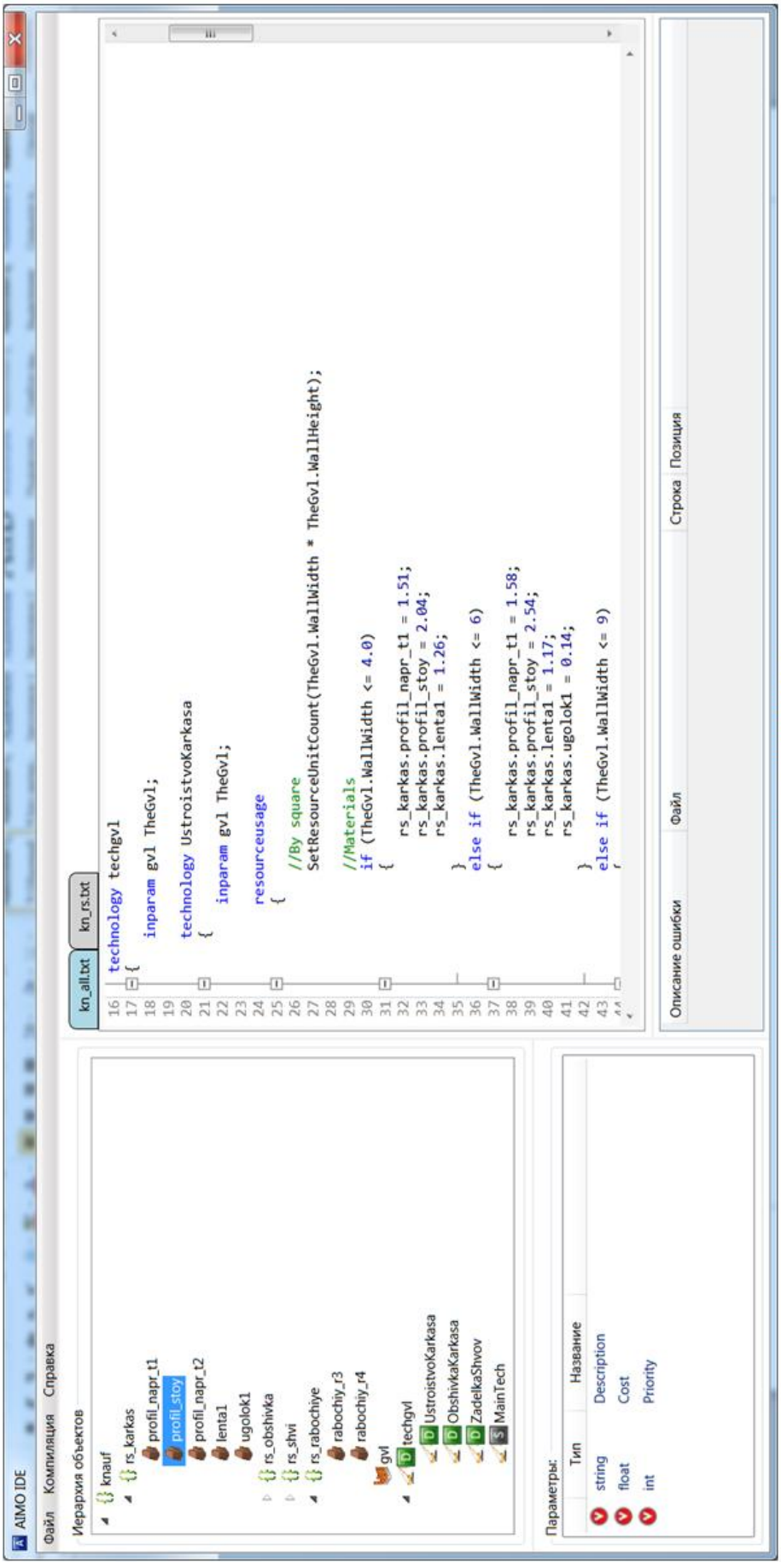

Рисунок. Головне вікно AIMO IDE

A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<Цифра> ::= 0|1|2|3|4|5|6|7|8|9

<Крапка> ::= .

<Початок_блок> ::= {

<Кінець_блок> :: = }

<Namespace> ::= namespace

<Construction> ::= construction

<Technology> ::= technology

<Resource> ::= resource

<Inparam> ::= inparam

<Instance> ::= instance

<Default> ::= default

<Param> ::= param

<Resourceusage> ::= resourceusage

<For> ::= for

<If> ::= if

<Else> ::= else

<While> ::= while

<Кінець_виразу> ::= ;

<Початок_масив> ::= [

<Кінець_масив> ::=]

<Тип_строка>::= string

<Тип_ціле> ::= int

<Тип_дійсне> ::= float

<Дужка_поч> ::= (

<Дужка_кін> ::=)

<Строка_поч> ::= "

Опис алфавіту мови «AIMO»

До алфавіту мови можна віднести літери латинського алфавіту, ключові слова, арифметичні оператори, а також спеціальні символи.

Список ключових слів мови:

- namespace – опис простору імен;
- construction – опис конструктиву;
- technology – опис технології;
- resource – опис ресурсів;
- inparam – опис вхідних параметрів;
- instance – опис екземпляру конструкції;
- default – опис блоку коду за замовчуванням;
- param – опис параметрів;
- resourceusage – використання ресурсів;
- for – цикл з лічильником;
- while – цикл з умовою;
- if – умовний оператор;
- else - умовний оператор.

IDE (середовище розробки)

Головне вікно AIMO IDE показане на рисунку.

Висновки

1. Запропонована мова об'єднує можливості мов імітаційного моделювання складних процесів і універсальних мов програмування.

2. Проведені дослідження на прикладах деяких фрагментів конструктивних і організаційно-технологічних моделей будівельних об'єктів показують можливість застосування AIMO IDE для розробки інтегрованих інформаційних технологій в будівництві.

3. Запропонована мова моделювання будівельних технологій використовується в експериметальному програмному продукті PMB (the Patterns-maker of Building), що розробляється з використанням розвитку перспективної методології конфігураторів [5].

Список літератури

1. Вигерс Карл. Разработка требований к программному обеспечению / Пер. с англ. – М.:Издательско-торговый дом «Русская редакция», 2004. – 576 с.

2. Задоров В.Б. Системний аналіз об'єктів і процесів: технологічні основи: Навчальний посібник. – К.:КНУБА, 2003. – 276 с.

3. Задоров В.Б., Київська К.І. Концепція створення сучасних інформаційних систем маркетингу підприємств. Східно-Європейський журнал передових технологій, №3/2(39), 2009. – С. 18-25.

4. Задоров В.Б. Про один підхід до створення технології попереднього системного проектування КІС підприємств // Управління розвитком складних систем. – 2010. – Вип. 01. – С. 56 – 83.

5. Задоров В.Б., Васильєв О.О. Інтеграція інформаційних моделей в проектуванні та управлінні будівництва на основі узагальненого формату знань та даних// Управління розвитком складних систем. – 2011. – Вип. 05. – С. 52 – 60.

6. Калянов Г.Н., CASE – технологии. Консалтинг в автоматизации бизнес-процессов. – 3-е изд. – М.: Горячая линия –Телеком, 2002. – 320 с.

7. Леффингуэлл Дин, Уидриг Дон. Принципы работы с требованиями к программному обеспечению. Унифицированный подход.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 448 с.

8. Филлип Кратчен Введение в Rational Unified Process. Пер. с англ. Второе издание. – М.: Издательский дом "Вильямс", 2002. – 235 с.

Стаття надійшла до редколегії 26.03.2013

Рецензент: д-р техн. наук, проф. С.В. Цюцюра, Київський національний університет будівництва і архітектури, Київ.