УДК 519.6

Т.В. Аксьонова

ПРОГРАМНА ТЕХНОЛОГІЯ ДЛЯ ПРОВЕДЕННЯ ІМІТАЦІЙНИХ ЕКСПЕРИМЕНТІВ З МАТЕМАТИЧНИМИ МОДЕЛЯМИ ФІЗІОЛОГІЧНИХ СИСТЕМ

Проводиться аналіз існуючих систем, на підставі чого доводиться необхідність розробки гнучкої системи, орієнтованої на фізіолога-дослідника. Наводиться опис розробленої системи, що дозволяє фізіологам будувати віртуальні організми з набору блоків, що постійно розширюється.

Вступ

Особливістю моделювання фізіологічних систем є необхідність враховувати горизонтальний та вертикальний напрямки взаємодії підсистем організму, більше того, процеси є різномасштабними у часі [1]. Отже, при розробці програмного засобу потрібно враховувати, що він працюватиме із великими моделями, що ймовірно міститимуть деякі підмоделі та будуть взаємодіяти між собою.

Серед існуючих програмних засобів потрібно згадати розроблений в медичному центрі університету Міссісіпі біологічний симулятор «HumMod» [2], що дозволяє проводити лабораторні дослідження для більш ніж 30 стандартних ситуацій, таких як голод, знаходження на великій висоті, нестача кисню та інші. Система є досить потужною, дозволяє працювати з інтегрованою моделлю організму людини, що складається з під моделей органів та рецепторів. Головним недоліком даного симулятора є те, що він поставляється із запрограмованими значеннями параметрів та фіксованими зв'язками, які вже неможливо змінити, отже експерименти обмежуються стандартним набором.

У межах проекту Physiome розроблено моделюючий комплекс JSim [3 – 5]. Цей комплекс орієнтований для роботи з математичними моделями фізіологічних систем, причому моделі можуть бути у вигляді як звичайних, так і диференційних рівнянь, також підтримуються інтеграли. Всі моделі мають відповідати відкритому стандарту, що з одного боку полегшує роботу з моделями інших розробників, а з іншого – ускладнює роботу фізіолога, оскільки мова MML, що використовується для побудови моделей, схожа на мову програмування і є інтуїтивною лише для розробників програмного забезпечення.

Спробою зробити процес роботи з моделями більш дружнім для користувача є інтегрована у Matlab система Simulink [6], та побудована на його основі біологічно-орієнтована система BioUML [7, 8]. На відміну від попередніх систем тут користувач має змогу самостійно будувати моделі на підставі графічних діаграм, причому можливим є використання стандартних моделей та збереження моделей в базі даних для подальшого використання. Крім того, BioUML підтримує роботу із ієрархічними моделями та дозволяє використання підмоделей. До недоліків слід віднести те, що використання BioUML є досить складним і користувач потребує спеціальних навичок.

Першою спробою вирішення проблем, що все ще стоять перед фізіологом-дослідником, була розробка спеціалізованої програми [9 – 11], що дозволяла створювати системи з моделей та підмоделей, працювати з моделями, що мають групи параметрів, та мала високу ефективність обчислень. Недоліком даного дослідницького інструменту була жорсткість системи: всі зв'язки задавалися на етапі програмування, а отже, для повторного використання моделей необхідним було втручання програміста. Наступним кроком розвитку системи стала побудова такого дослідницького середовища, в якому фізіолог-дослідник має змогу самостійно будувати віртуальний організм із існуючих модулів.

Розроблена комп'ютерна технологія дозволяє:

реалізувати моделі, які описуються співвідношеннями «вхід-вихід» або «вхід-стан-вихід», що відповідає прямим алгебраїчним залежностям та диференційним рівнянням у формі Коші;

– створювати конфігурації таких моделей (фіксовані сукупності моделей, що працюють разом та дозволяють подавати на вхід моделі вихід іншої моделі);

– в будь-який час додавати нові моделі до системи;

– ефективно виконувати обчислення за рахунок використання паралельно працюючих потоків;

– змінювати з інтерфейсу та зберігати параметри та вхідні дані для кожної з моделей;

– зберігати результати експериментів у різних текстових та графічних форматах;

– гнучко налаштовувати інтерфейс користувача.

Структура програмного забезпечення

Згідно наведених вимог було розроблено структуру програмного комплексу для моделювання, показану на рис. 1. Щодо цієї архітектури, моделюючий комплекс є трирівневим. Кожен рівень є максимально автономним і реалізує відведену йому задачу. Детальний опис кожного з рівнів наведено у наступних пунктах, при цьому використано такі терміни:

модель – це певний клас, що реалізує базовий для всіх моделей інтерфейс IModel та містить у собі опис певного об'єкта (фізіологічної системи, органу, механізму, зовнішнього середовища і т. д.);

модельна зборка – модель скомпільована у dll зборку з унікальним іменем;

конфігурація – зафіксована і поіменована сукупність моделей, зв'язків між ними, їх параметрів та вхідних даних.

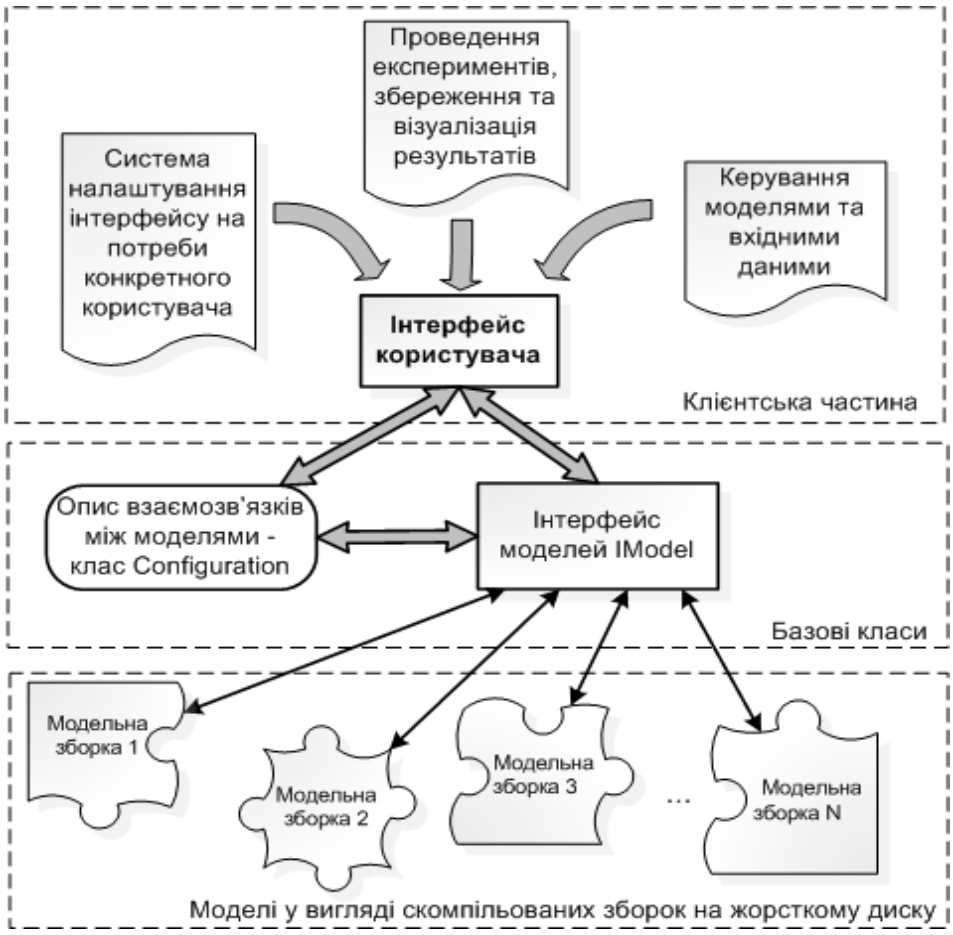

Рис. 1. Архітектура програмного забезпечення

Опис клієнтської частини

Почнемо детальний аналіз архітектури з клієнтської частини, яка є репрезентативною частиною усього комплексу. Головне вікно системи (рис. 2) можна поділити на 6 частин:

– титульна частина – показує назву та версію системи, а також поточну конфігурацію моделей;

– стрічка панелі інструментів - виконана у стилі MS Office 2007 як найбільш зручний та перспективний варіант, що дозволяє зекономити дорогоцінний простір робочої області та полегшити для користувача пошук потрібного йому інструментарію серед різноманіття існуючих. Містить системне меню (закладка Main) та меню для вибраної у дереві моделі (закладка з назвою, що збігається з ім'ям моделі). Детальний опис кожної кнопки меню міститься у керівництві користувача, що розповсюджується разом з даною програмою та може бути відкрите за допомого кнопки Help головного меню;

– ліва панель – містить дерево моделей та панель налаштувань експерименту;

– центральна частина – слугує для виводу результатів обчислень у вигляді графіків або таблиці. Крім того, кожна модель може мати свою закладку на центральній частині для покращення візуалізації результатів або виконання додаткових налаштувань;

– права панель – для вибраної в дереві моделі містить таблицю для завдання параметрів (закладка Parameters), таблицю для завдання початкового стану моделі (закладка Init values), та панель налаштувань даної моделі (закладка Settings);

– панель стану – показує хід виконання обчислень.

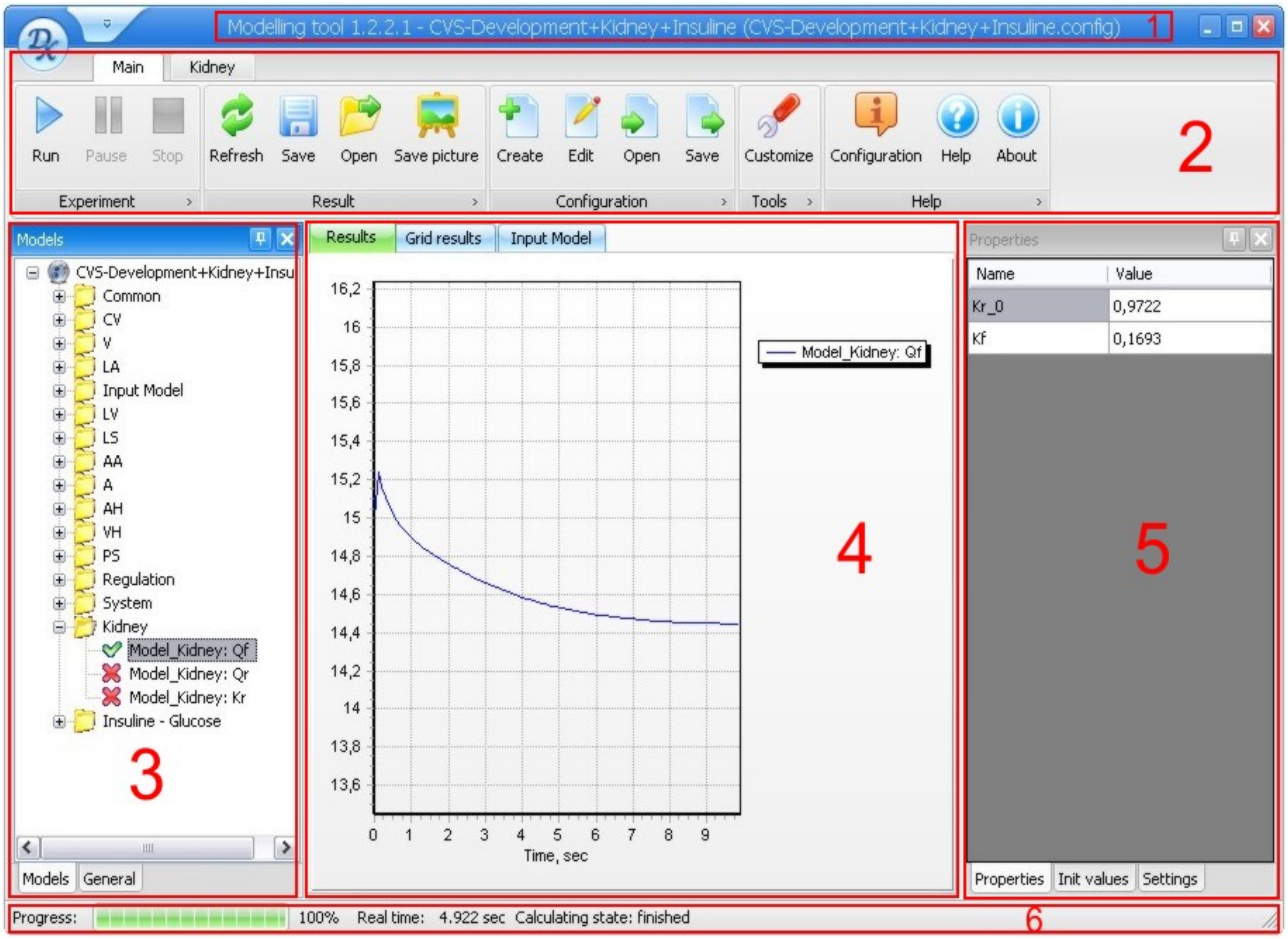

Рис. 2. Головне вікно системи

Цей поділ є досить умовним, так як робоча область головного вікна в повному обсязі підтримує технологію drag-n-drop, тому всі її дочірні вікна користувач може розмістити у будь-якому порядку, «вмонтувати» одне в одне, або взагалі відокремити від головного вікна. Більше того, використовуючи форму налаштувань системи користувач може у будь-який час сховати чи знов показати дані панелі.

Для більшої зручності користувача система автоматично підлаштовується під нього за рахунок зберігання так званого «користувацького вибору». Це досягається через створення спеціального класу що містить такі дані як: поточна кольорова схема та шрифти, розмір та положення головного вікна, перелік видимих панелей та їх розташування, остання використовувана конфігурація, час експерименту, параметри підказок та графіка результатів. При закритті програми всі ці дані автоматично серіалізуються в xml файл з ім'ям default.prefs що зберігається в тому самому каталозі, що й виконавчий файл системи, а при запуску за наявності даного файлу вони з нього десеріалізуються після чого інтерфейс перебудовується згідно з останнім користувацьким вибором.

Дуже важливою частиною головного вікна є дерево моделей (рис. 3), що дозволяє користувачу бачити з якими саме моделями він нині працює, вибирати модель для налаштувань, вибирати які змінні яких моделей буде показано на графіки результатів та навіть змінювати ім'я та опис моделей та їх змінних. Дерево моделей розташовано на закладці Models лівої панелі (частина 3 головного вікна).

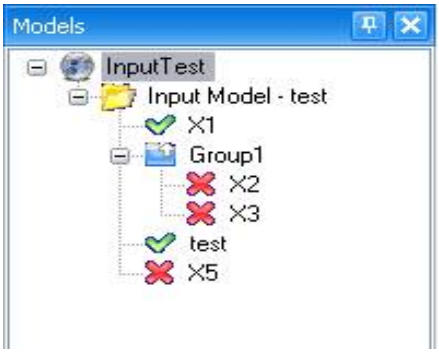

Рис. 3. Дерево моделей

Для різних елементів у дереві використано наступні позначки:

– конфігурація;

– модель; при наведенні миші на вузол такого типу спливаюча підказка містить опис моделі;

– група змінних моделі;

– змінна моделі, що буде відображена на графіку результатів;

– змінна моделі, що не буде відображена на графіку результатів;

– результат обчислень відкритий з .result файлу.

Для переводу змінної моделі зі стану «буде відображена на графіку» до стану «не буде відображена на графіку» користувачеві необхідно лише клікнути лівою клавішею миші по вузлу даної змінної.

З головного меню системи може бути відкрито вікно керування конфігураціями (рис. 4), що дозволяє створювати нові та редагувати існуючі конфігурації моделей.

Дане вікно системи можна поділити на 5 частин:

– титульна частина – показує назву конфігурації, що редагується;

– меню – дозволяє зберігати конфігурацію у файл, добавляти чи виділяти моделі, відкривати граф конфігурації;

– описова частина – містить назву та детальний опис конфігурації;

– перелік всіх моделей конфігурації;

– перелік всіх зв'язків конфігурації – виконаний у вигляді таблиці, де для кожної вхідної змінної кожної моделі створено рядок, у якому в стовпці Link має бути встановлено відповідне вихідне значення деякої іншої моделі (зв'язки встановлюються та змінюються за допомогою кліку мишею по потрібній клітині з стовпчика Link) або на вхід моделі може бути прив'язана константа.

Більш наочно конфігурація представляється за допомогою форми, що містить направлений граф конфігурації (рис. 5), вершинами якого є моделі, що містить у собі дана конфігурація. Вершини графа поєднані дугою, якщо між моделями конфігурації є зв'язок.

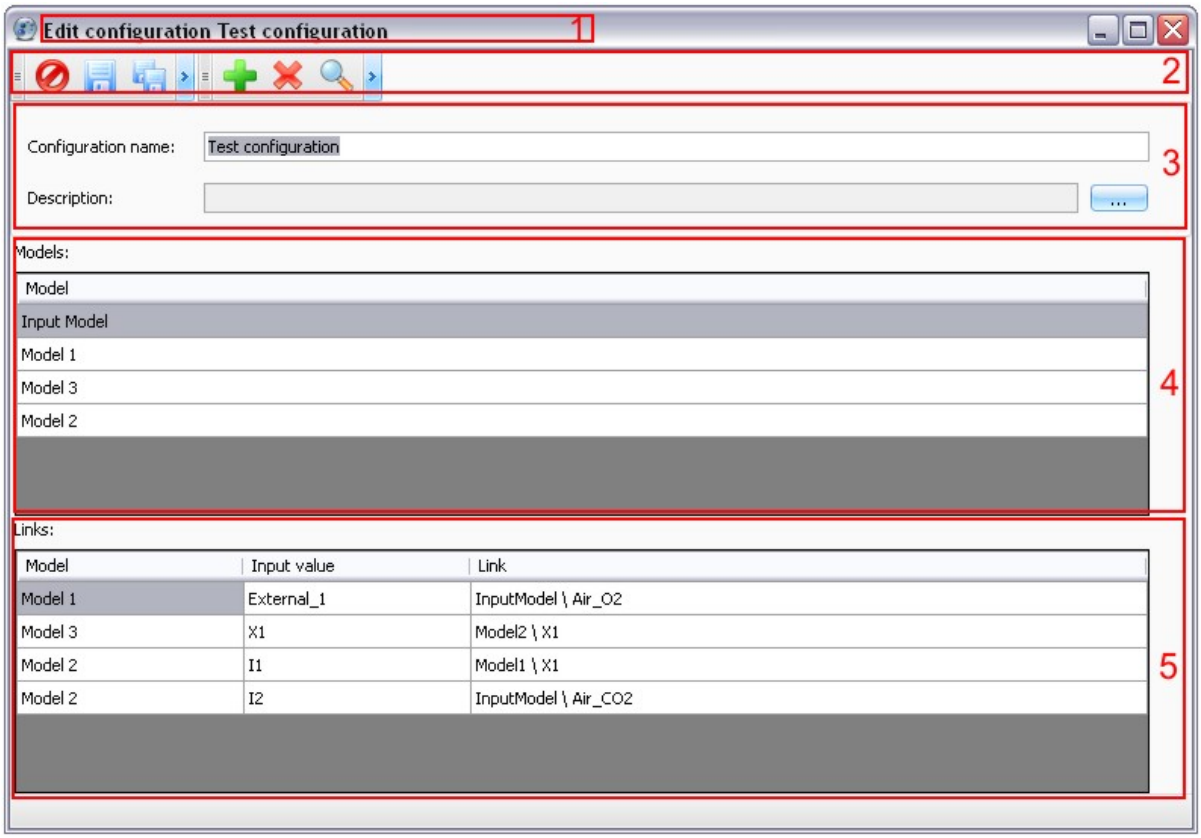

Рис. 4. Вікно керування конфігураціями

Дуга виходить з моделі, що має вихідну змінну, і входить у модель, що має вхідну змінну на яку приєднано дану вихідну змінну. Якщо конфігурація містить моделі, що не під'єднанні до жодної іншої моделі, то даний граф буде незв'язним.

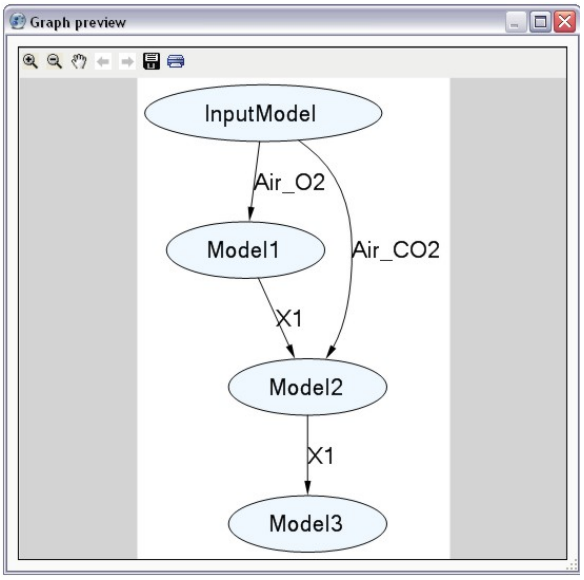

Рис. 5. Вікно графа конфігурації

З точки зору користувача, що проводить імітаційні експерименти в даній системі, найбільш часто використовуваним модулем клієнтської частини є підсистема виведення та збереження результатів. Як бачимо з рис. 6 дана підсистема містить у собі чотири компоненти, причому один з них не є постійно видимим, оскільки він надто громіздкий та не так часто вживаний як інші. Розглянемо ці компоненти більш детально:

– елементи меню для швидкого збереження результатів – дозволяють одним кліком миші зберегти зображені на графіку результатів дані або як рисунок у графічний файл формату .jpg або як набір розрахованих значень у текстовий файл формату .xlsx;

– графік результатів – відображає у вигляді двомірних кривих розраховані значення для всіх змінних моделей, що вибрано у дереві моделей. Для більшої зручності користувача система автоматично підбирає кольори так щоб за умови що таке можливо всі зображені криві мали різні контрастні кольори;

– редактор графіка результатів – дає користувачеві повний доступ до всіх налаштувань графіка результатів та можливих засобів збереження отриманих значень. Основними можливостями редактора є: налаштування написів на графіку; налаштування кольорів та позначок для кожної з зображених кривих; побудова нових кривих як функції існуючих (підтримуються основні математичні функції +, - , *, /, деякі статистичні – середнє, мода, медіана та спеціальні – брати більше чи менше значення); друк зображення на принтері; збереження розрахованих даних як картинки у одному з можливих графічних форматів (підтримуються .bmp, .jpg, .png, .gif, .tiff, .pdf); збереження розрахованих даних як набору значень з можливих текстових форматів (підтримуються .txt, .xml, .html, .xlsx);

– таблиця результатів (рис. 7) – дозволяє відслідковувати точні значення обраних змінних у фіксовані моменти часу та експортувати їх використовуючи буфер обміну або зберігати їх як .xlsx файли.

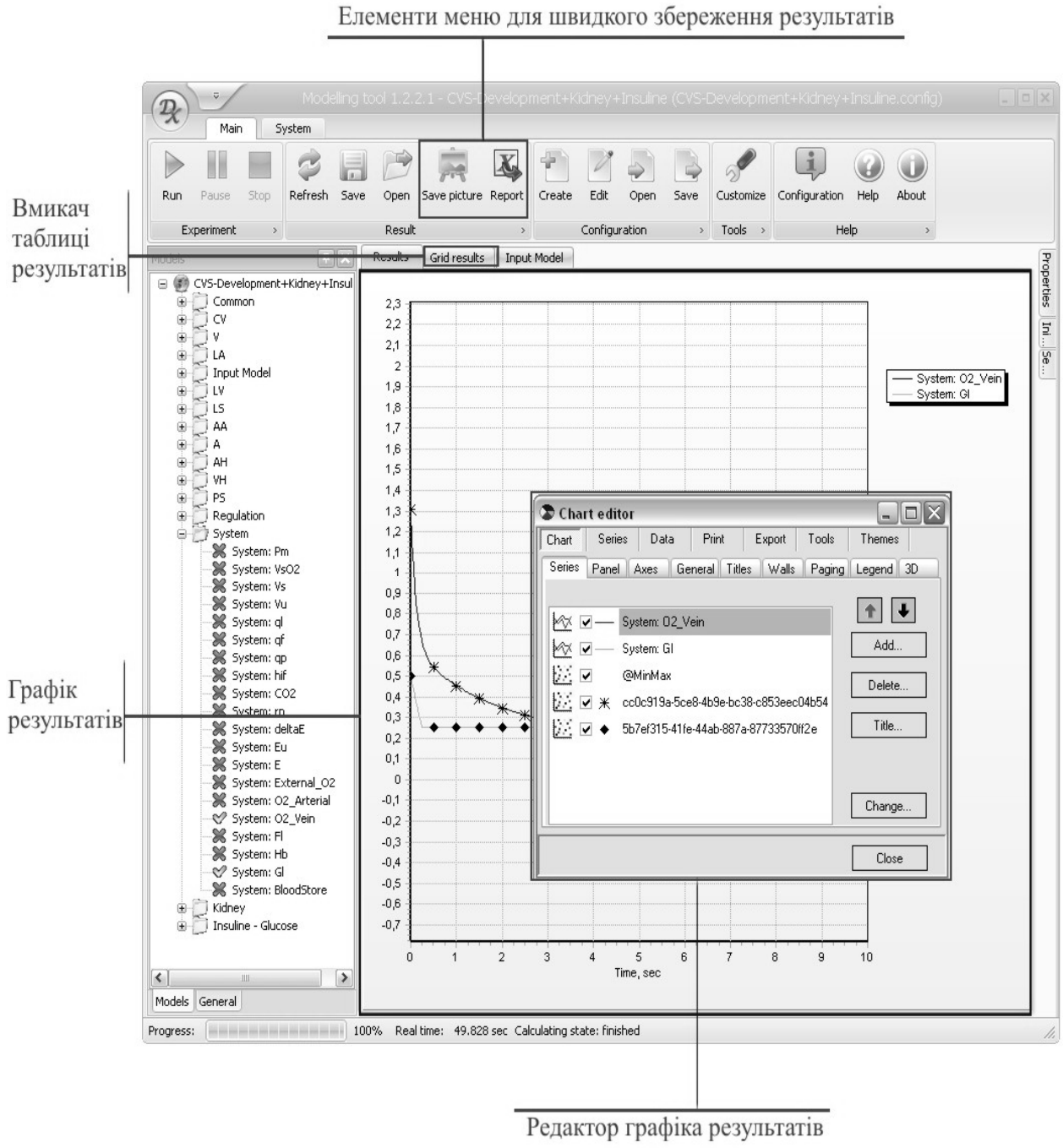

Рис. 6. Підсистема виведення та збереження результатів

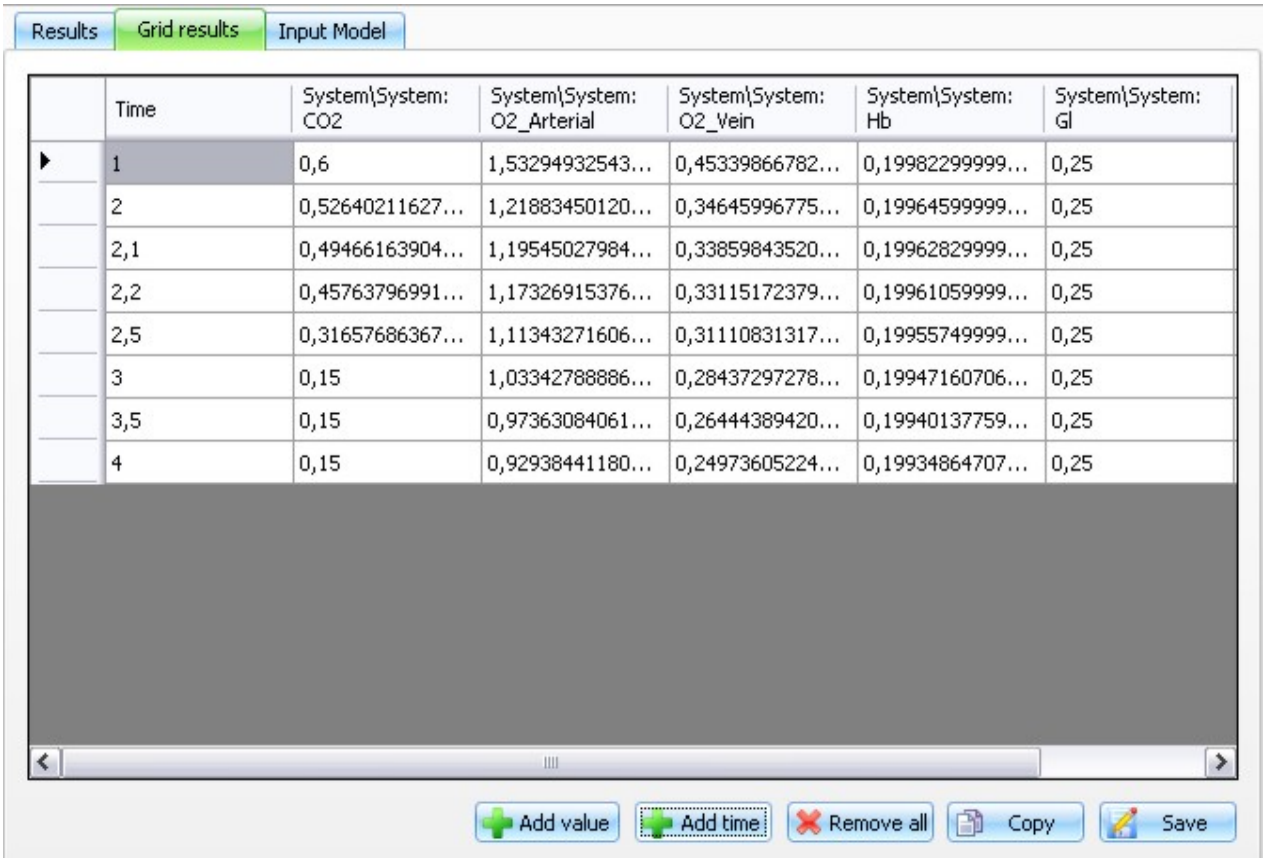

Рис. 7. Таблична форма виведення результатів

Опис базових класів

Основними базовими класами системи є:

1. Інтерфейс IXmlObject – містить методи для роботи з Xml. Є базовим інтерфейсом, оскільки всі об'єкти в системі мають зберігати та відновлювати свої параметри з .xml файлів. Даний підхід забезпечує уніфікацію методів для збереження та завантаження даних і дозволяє досвідченим користувачам редагувати значення прихованих параметрів системи.

2. Клас ObjectBase – реалізує інтерфейс IXmlObject і дозволяє швидко створювати класи, що вміють працювати з .xml файлами. Даний клас спадкується всіма іншими базовими класами системи.

3. Інтерфейс IModel – його мають реалізовувати всі моделі системи. Містить наступні групи методів та властивостей:

– для отримання загальної інформації про модель (внутрішнє ім'я, користувацьке ім'я, детальний опис);

– інтерфейсні методи (за потреби повертають елементи меню та головну панель для даної моделі);

– методи для налаштувань (списки змінних та параметрів);

– методи для розрахунків (шаг розрахунку, поточний шаг розрахунку, стан розрахунку, метод Calculate, що саме й виконує розрахунки для даної моделі);

– методи для роботи з Xml (збереження та відновлення з .xml файлів налаштувань моделі та результатів обчислень).

4. Клас Value – описує змінну моделі. Містить наступні групи методів та властивостей:

– загальна інформація про змінну моделі (внутрішнє ім'я, користувацьке ім'я, ім'я групи змінних, тип);

– для проведення розрахунків (мінімальне та максимальне значення, масив розрахованих значень);

– для зв'язків між моделями (ім'я моделі та ім'я змінної моделі до яких ця змінна прив'язана);

– методи для роботи з Xml (збереження та відновлення з .xml всіх параметрів класу).

5. Містить наступні групи методів та властивостей:

– загальна інформація про параметр моделі (внутрішнє ім'я, користувацьке ім'я);

– для проведення розрахунків (мінімальне та максимальне значення, поточне значення);

– методи для роботи з Xml (збереження та відновлення з .xml всіх параметрів класу).

6. Клас ModelBase – створений для спрощення і пришвидшення створення простих моделей. Він реалізовує всі методи інтерфейсу IModel наступним чином:

– там де це можливо вертає пусті значення;

– за замовчуванням створює елементи меню, що дозволяють зберігати та відновлювати з файлу налаштування моделі незалежно від конфігурації;

– знаходить всі задекларовані в класі змінні типів Value та Parameter і будує з них списки змінних та параметрів;

– контролює поточний шаг і стан розрахунку.

7. Клас Configuration – описує систему моделей. Даний клас виконує всі розрахунки та повідомляє інтерфейсну частину системи про стан виконання. Містить наступні групи методів та властивостей:

– загальна інформація про систему моделей (ім'я, опис, ім'я файлу в якому зберігаються дана конфігурація, перелік моделей);

– для проведення розрахунків (методи що стартують чи зупиняють обчислення та методи для синхронізації моделей);

– методи для роботи з Xml (збереження та відновлення з .xml всіх параметрів класу і всіх моделей);

– методи для збереження результатів розрахунків;

– методи для збереження та відновлення налаштувань дерева моделей для даної конфігурації.

8. Клас SavedConfiguration – описує збережені результати розрахунків. Використовується при аналізі проведених експериментів. Особливістю даного класу є те, що параметри не можна змінювати. Містить наступні групи методів та властивостей:

– загальна інформація про систему моделей (ім'я, опис, ім'я файлу в якому зберігаються дані, перелік моделей та розраховані значення моделей);

– методи для роботи з Xml (тільки відновлення з .xml всіх значень).

Структура файлів на жорсткому диску

Розроблена система є «гнучкою» у тому розумінні, що нові моделі можуть бути під'єднанні до неї у будь-який час. Це досягається за рахунок того, що система працює не з класами, а з заздалегідь скомпільованими модельними зборками, кожна з яких містить клас, що реалізує базовий інтерфейс IModel. При запуску системи вона шукає доступні модельні зборки та будує з них лист всіх доступних моделей, з якого далі необхідні моделі будуть додані до конфігурації.

Оскільки пошук модельних зборок по всій файловій системі комп'ютера займає забагато часу, то система бачить лише ті модельні зборки, що у вигляді .dll файлів лежать у каталозі моделей ModelsFolder, що визначається конфігураційним файлом системи (Modelling tool.exe.config) (рис. 8); якщо цей каталог не задано (по замовчуванню) – у тому каталозі, де знаходиться виконавчий файл системи Modelling tool.exe (системному каталозі). При завданні шляху до каталогу припускається використання як абсолютних шляхів, так і шляхів відносно системного каталогу.

Ще одним спеціальним каталогом використовуваним системою є каталог даних. У цьому каталозі зберігаються всі користувацькі файли – такі як файли конфігурацій, налаштування окремих моделей, збережені результати розрахунків. Шлях до каталогу даних налаштовується аналогічно шляху до каталогу моделей (ім'я у конфігураційному файлі системи – DataFolder). Оскільки зазвичай користувач має багато збережених файлів даних, то заради запобігання перенасичення системного каталогу необов'язковими файлами за замовчуванням каталог даних розміщено у підкаталозі Data каталогу з виконавчим файлом.

```
<applicationSettings>
    <Client.My.MySettings>
        <setting name="ModelsFolder" serializeAs="String">
            <value />
        </setting>
    </Client.My.MySettings>
</applicationSettings>
<userSettings>
    <Client.My.MySettings>
        <setting name="DataFolder" serializeAs="String">
            <value>Data\</value>
        </setting>
        <setting name="MaxChartValue" serializeAs="String">
            <value>100000</value>
        </setting>
    </Client.My.MySettings>
</userSettings>
```

Рис. 8. Фрагмент конфігураційного файлу системи

Алгоритм проведення обчислень

Обчислення ініціюються користувачем з головного меню головної форми системи. Після цього клієнтська програма очищає графік розрахунків, обнуляє дані на панелі стану розрахунків та викликає метод поточної конфігурації, що саме й виконує розрахунки. У процес розрахунків клієнтська програма не втручається (за умови, якщо користувач не захоче зупинити процес), оновлення інтерфейсу відбувається наступним чином: із початком розрахунків запускається спеціальний таймер і на кожен його тік клієнтська частина запитує у поточної конфігурації в якому стані знаходиться процес, до якого часу вже виконано розрахунки і за потреби оновлювати графіки результатів також запрошуються розраховані значення. Оскільки кожна модель розраховується незалежно і зі своїм кроком, то розрахованим часом вважається максимальний час, на який кожна модель конфігурації вже розрахувала всі свої змінні та саме цей час (і розраховані до нього значення) показуються користувачеві; розрахунки вважаються завершеними коли всі моделі завершать свої розрахунки. Коли конфігурація повідомляє що розрахунки закінчено таймер вимикається і оновлення інтерфейсу зупиняється.

Конфігурація виконує розрахунки наступним чином: для кожної моделі виконує підготовчі дії (встановлює час експерименту, виділяє пам'ять та ін.), створює окремий потік і в цьому потоці запускає процес обчислень моделі. Перед обчисленням кожного наступного кроку для кожної моделі конфігурація перевіряє, чи має модель вхідні дані. Якщо ні, то модель просто розраховує наступний крок, якщо ж так, то конфігурація запрошує необхідні вхідні дані у моделі, з якою пов’язана дана модель (якщо ці дані ще не готові – обчислення даної моделі буде призупинено поки необхідні вхідні дані не надійдуть). Крім того, конфігурація відслідковує всі виключні ситуації, що виникають під час обчислень, і якщо для будь-якої моделі така ситуація виникає, то розрахунок всієї конфігурації примусово зупиняється і користувачеві показується відповідне повідомлення.

Процес проведення обчислень схематично показано на рис. 9. На даній схемі не зафарбовані прямокутники відповідають діям користувача, зафарбовані у косу клітинку – діям клієнтської частини системи, зафарбовані у крапочку – діям підсистеми оновлення інтерфейсу, повністю зафарбовані – операції що виконуються у конфігурації, зафарбовані у клітинку – операції що виконуються у моделях.

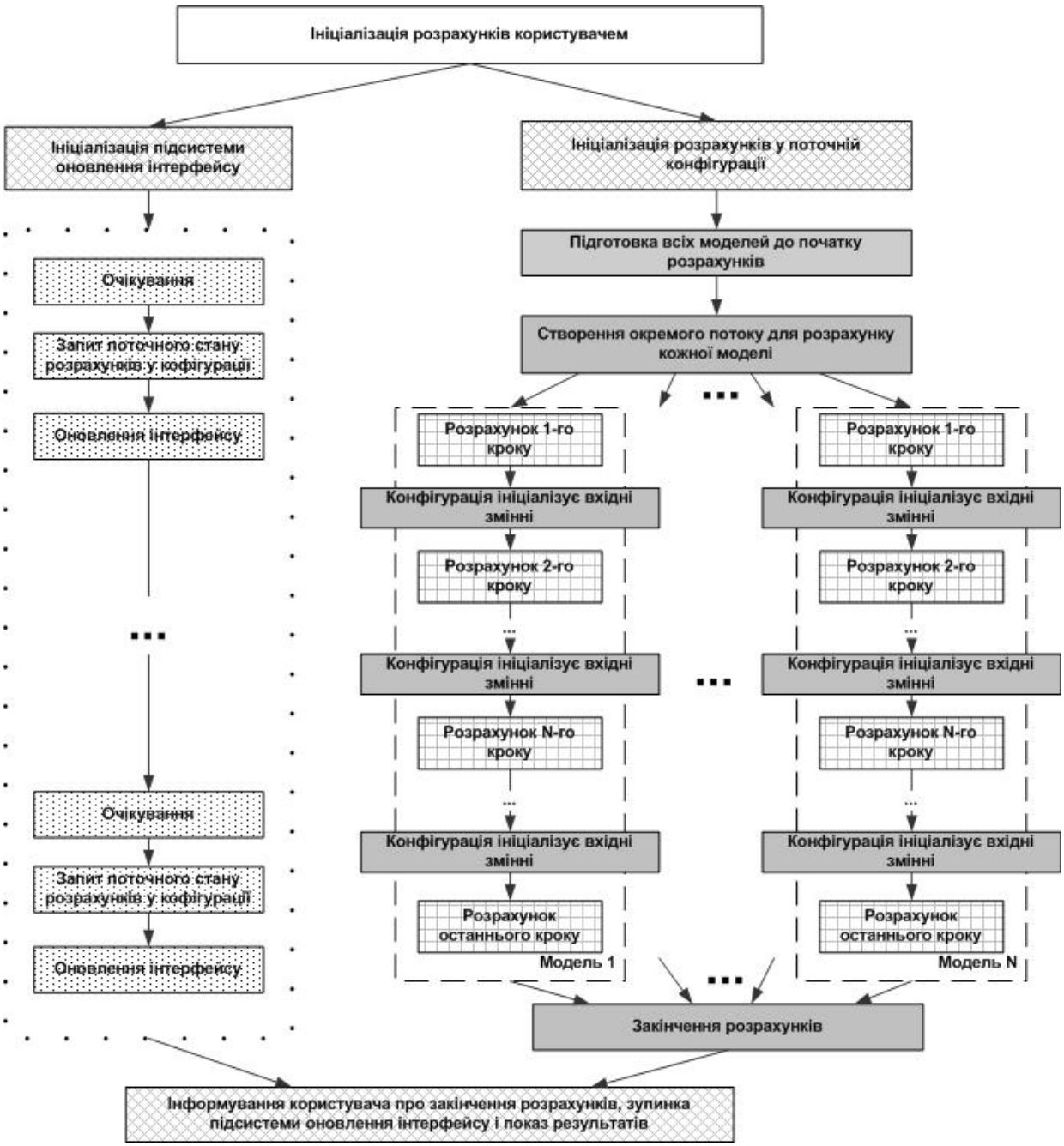

Рис. 9. Схема проведення обчислень

Висновки

Розроблена програмна технологія дозволяє робити декомпозицію моделей складних фізіологічних систем на під моделі, що під час виконання експериментів пов'язуються між собою за допомогою файлів конфігурацій. Це дозволяє спростити процес налаштування та візуалізації, а також збільшує можливості з повторного використання моделей. Дана система використана при побудові віртуального організму, що складався з наступних частин: серцево-судинна система, нирки, лімфатична система, рідкі середовища, термо- та баро- регулятори [12], що показало її зручність для проведення практичних досліджень.

1. *Григорян Р.Д., Атоев К.Л., Лиссов П.Н., Томин А.А.* Программно-моделирующий комплекс для теоретических исследований взаимодействия физиологических систем человека // Проблеми програмування. – 2006. – № 1. – С. 79–92.

2. *An* integrative mathematical model of human biomedicine [Електронний ресурс] http://hummod.org/
3. *Сайт* комплексу JSim проекту Physiome [Електронний ресурс] // http://nsr.bioeng.washington.edu.
4. *Hunter P.*, Robbins P., Noble D. The IUPS human physiome project // European Journal of Physiology. – 2002. – Vol.44. – P. 1–9.
5. *Bassingthwaighte J. B.* Strategies for the Physiome Project // Annals of Biomedical Engineering. – 2000. – Vol. 28. – P. 1043–1058.
6. *Simulink* Release Notes / [Електронний ресурс]: http://www.mathworks.com/access/helpdesk/help/pdf_doc/simulink/rn.pdf.
7. *Kolpakov F.A.* BioUML – framework for visal modeling and simulation of biological systems // International Conference on Bioinformatics of Genome Regulation and Structure-Systems Biology: proceedings. – Novosibirsk, 2002.
8. *Колпаков Ф.А.* BioUML – компьютерная программа для визуального моделирования биологических и других сложных систем: тезисы IV Всеросс. конф. молодых ученых по математическому моделированию и информационным технологиям (г. Красноярск, 3-5 ноября 2003) // Институт вычислительного моделирования СО РАН. – Красноярск, 2003. – С. 98.
9. *Аксенова Т.В.* Імітаційне моделювання процесів клітинного реагування на дефіцит енергії // СППМІ–2008.
10. *Аксенова Т.В.* Специализированное программное обеспечение для решения физиологических задач методом математического моделирования // Проблеми програмування. – 2008. – № 2–3. – С. 765–769.
11. *Григорян Р.Д., Аксенова Т.В.* Моделирование адаптивного реагирования организма на изменения в окружающей среде // Кибернетика и системный анализ. – 2008. – № 1. – С. 136–147.
12. *Григорян Р.Д*, *Лиссов П.Н., Аксенова Т.В., Мороз А.Г.* Специализированный программно – моделирующий комплекс “PHYSIOLRESP”. // Проблеми програмування. – 2009. – № 2. – С. 67–82.

Отримано 14.07.2011

Про автора:

Аксьонова Тетяна Валеріївна,
інженер-програміст.

Місце роботи автора:

Інститут програмних систем
НАН України,
03187, Київ,
проспект Академіка Глушкова, 40.
Тел.: 526 5169.
E-mail: akstanya@ukr.net