ОСОБЛИВОСТІ ВИКОРИСТАННЯ UML ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СУДНОВИХ ЕЛЕКТРОЕНЕРГЕТИЧНИХ СИСТЕМ

УДК 004.051

ДОРОГАНЬ Ольга Ігорівна
к.т.н., асистент кафедри теоретичної електротехніки та електронних систем,
Національний університет кораблебудування ім. адм. Макарова.
Наукові інтереси: інформаційне забезпечення електроенергетичних систем, імітаційне моделювання в електроенергетиці.

УШКАРЕНКО Олександр Олегович
к.т.н., доцент кафедри теоретичної електротехніки та електронних систем,
Національний університет кораблебудування ім. адм. Макарова.
Наукові інтереси: автоматизація електроенергетичних систем, інформаційне забезпечення електроенергетичних систем.

ВСТУП

Завданням програмного забезпечення верхнього рівня є дистанційний контроль і управління електроенергетичною системою. В роботі [1] наведено обґрунтування необхідності модернізації систем збору, передачі та відображення даних про стан електроенергетичної установки. Сучасні системи керування судновими електроенергетичними системами (СЕЕС) є розподільними та можуть розглядатися на трьох рівнях [2]: апаратура та програмне забезпечення верхнього рівня (SCADA-програма), які містять автоматизоване робоче місце (АРМ) оператора; територіально розподілені контролери автоматизації та комунікаційна інфраструктура. Розробці програмного забезпечення для керування енергосистемами присвячена робота [3]. Розвиток технологій програмної інженерії обумовив зміну вимог до АРМ оператора [4], тому наведені в роботі [3] принципи потребують перегляду. В даній роботі пропонується використання об'єктно-орієнтованого програмування з явним виділенням станів [5-8] для створення програмного забезпечення верхнього рівня системи керування СЕЕС. Для повної автоматизації СЕЕС необхідно формалізувати процес управління складною системою з великим числом елементів, кожен з яких може мати кілька станів [9, 10]. Актуальність розвитку даного напрямку обумовлена необхідністю підвищення надійності та ефективності оперативного управління судновими електроенергетичними установками, зменшення витрат часу на аналіз оперативної ситуації черговим персоналом. Особливої актуальності це завдання набуває в умовах проведення реконструкцій на діючих електроенергетичних установках.

Мета роботи полягає у визначенні компонентів програмного забезпечення автоматизованого робочого місця оператора суднової електроенергетичної системи та взаємозв'язків між ними з використанням уніфікованої мови моделювання UML, а також розглянути принципи опису їх поведінки за допомогою UML діаграм активності та діаграм станів.

ОСНОВНИЙ МАТЕРІАЛ

Особливостями систем керування СЕЕС є можливості контролю та керування генераторними агрегатами та навантаженням на нижньому рівні системи; відображення топології суднової електроенергетичної системи та параметрів, що контролюються, на верхньому рівні. Завданнями програмного забезпечення є контроль та керування СЕЕС. Вирішення цих задач досягається виконанням програмою наступних функцій: забезпечення засобами для складання топології схеми; відображення параметрів поточного стану СЕЕС та передача команд оператора до контролерів автоматизації в режимі реального часу.

Таким чином, можна виділити два режими роботи проектованого середовища:

– режим редагування (складання мнемосхеми (структурної схеми) СЕЕС та налагодження мережевих властивостей її компонентів);

– режим роботи в мережі (обмін даними з контролерами автоматизації та відображення отриманих параметрів).

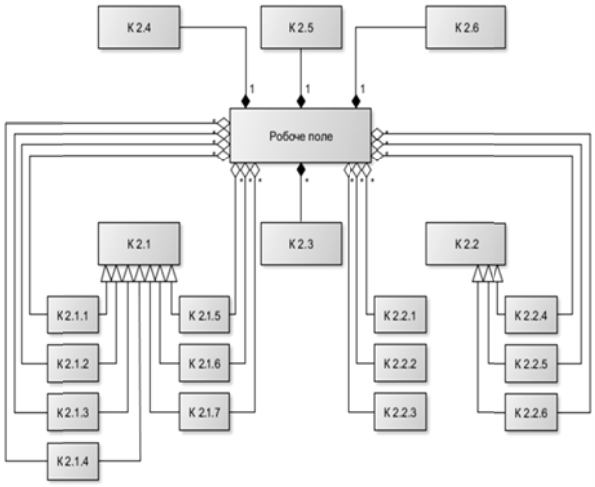

Рис. 1. Діаграма класів програмного забезпечення

Таблиця 1.

Класи, які реалізують поведінку компонентів

Компонент	Назва класу
Базовий клас з можливістю зв'язку	К 2.1
Дизель-генератор	К 2.1.1
Дизель	К 2.1.2
Генератор	К 2.1.3
Асинхронний. Двигун 1	К 2.1.4
Асинхронний двигун 2	К 2.1.5
Автомат	К 2.1.6
Трансформатор	К 2.1.7
Базовий клас з можливістю масштаб.	К 2.2
Текстовий надпис	К 2.2.1
Кнопка	К 2.2.2
Шкальний індикатор	К 2.2.3
Світлодіод 1	К 2.2.4
Світлодіод 2	К 2.2.5
Стрілочний індикатор	К 2.2.6
Шина	К 2.3
Діалог пар. Генератора	К 2.4
Діалог параметрів дизеля	К 2.5
Діалог процесу обміну	К 2.6

Складання схеми СЕЕС відбувається шляхом переміщення компонентів з Бібліотеки на Робоче поле та з'єднання їх в необхідному порядку. Під компонентами розуміються графічні символи та позначення, що відповідають фізичним (наприклад, генератор) або віртуальним (наприклад, індикатор) елементам СЕЕС. На рис. 1 наведена діаграма класів (розшифровка позначень класів приведена у табл. 1), що відображає взаємозв'язки між класами, які реалізують поведінку компонентів та допоміжних.

Кожен з компонентів та Робоче поле є сутністю зі складною поведінкою. Для розділення опису логіки їхньої поведінки від опису семантики кожної з дій при реалізації класів можна використати автоматний підхід, що в рамках об'єктно-орієнтованого програмування означає явне виділення станів та інкапсуляцію автомата в класі. Взаємодія об'єктів в системі здійснюється за допомогою повідомлень, серед яких можна виділити групи:

– повідомлення між Робочим полем та компонентами стосовно графічних дій (умовно позначені як множина дій $W = \{w1, w2\ldots\}$); ці дії реалізуються за допомогою базових класів («Базовий клас з можливістю масштабування» та «Базовий клас з можливістю створення зв'язку»), тому група повідомлень стосується лише базових класів та Робочого поля;

– повідомлення між Робочим полем та компонентами стосовно обміну даними (множина дій $D = \{d1, d2\}$); ці повідомлення стосуються компонентів, що передбачають обмін даними з контролерами автоматизації у режимі роботи в мережі («Дизель-генератор», «Генератор», «Автомат», «Кнопка управління», «Світлодіоди»);

– повідомлення між компонентами (множина дій $V = \{v1, v2\}$); група повідомлень стосується зміни кольору шин та пов'язаних з нею компонентів при зміні стану Автомату та відображення отриманих при обміні даними параметрів генератора за допомогою компонентів індикації.

Взаємодія між основними об'єктами програми графічно представлена на рис. 2 (розшифровка позначень класів приведена у табл. 1).

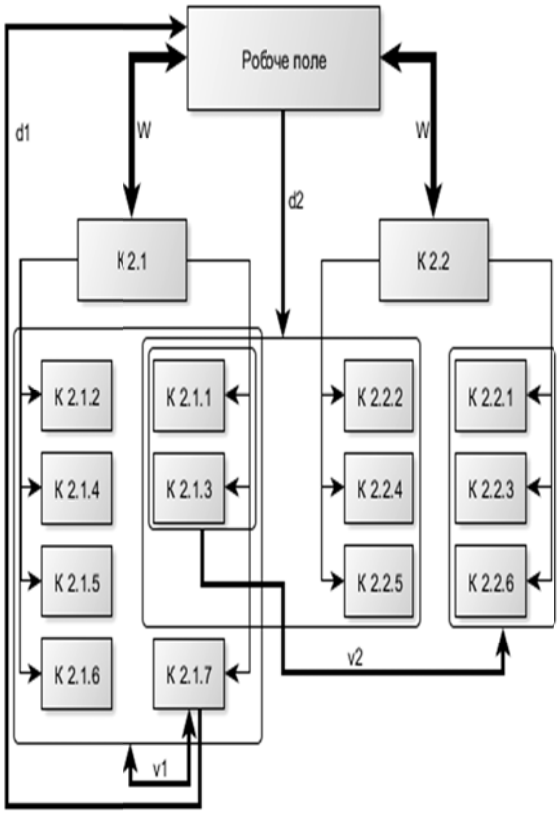

Рис. 2. Діаграма взаємодії між основними об'єктами програми

Для всіх елементів схеми (компонентів та Робочого поля) загальними є дії користувача з їх вікнами (переміщення курсору над об'єктом, натискання/відтискання лівої та правої кнопок миші). Ця множина дій позначена як $Xu = \{x1, x2\ldots\}$. Далі логіка роботи кожного з класів реалізована окремо. Оскільки робота програмного забезпечення відбувається в двох режимах, то в кожному класі реалізовано автомати для кожного з режимів окремо.

На рис. 3 наведено приклад діаграми станів автомату, що реалізує поведінку Робочого поля в режимі складання схеми СЕЕС.

В цьому режимі об'єкт «Робоче поле» має виконувати всі операції, пов'язані з шиною (створення, продовження, видалення повне та часткове, зміна кольору), створення компонентів та приймати участь при їх видаленні та переміщенні. Поведінка класу описана за допомогою автомату $A = \{Q, X, Y, f, g\}$, де:

– $Q = \{q0, q1, q2, q3, q4, q5, q6\}$ – множина станів автомату:

$q0$ – об'єкт має фокус (початковий стан);

$q1$ – створення компонента (поведінка в стані наведена на рис. 4). Створення компонента відбувається при його переміщенні з Бібліотеки компонентів на Робоче поле: при натисканні на компоненті в Бібліотеці в глобальну змінну записується його порядковий номер, який при відображені рамки на Робочому полі та створенні компонента дозволить його ідентифікувати; якщо переміщення не відбулось, глобальна змінна приймає нульове значення.

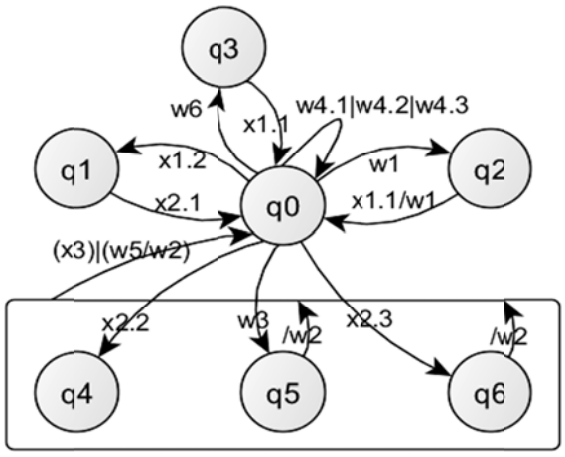

Рис. 3. Діаграма станів автомату в режимі складання схеми

Аналіз дій «Створення» та «Переміщення» компонента показав, що при обох операціях виконуються однакові дії щодо відображення та переміщення рамки. Оскільки при переміщенні об'єкт надсилає повідомлення з координатами рамки ($w4.1 - w4.3$), то оновлення зображення рамки виділено в окрему функцію і для переміщення компонента додаткового стану не введено (при отриманні кожного з повідомлень лише змінюється положення рамки; при створенні такий механізм виявився неможливим, тому що об'єкт ще не існує і Робоче поле само повинно відслідковувати зсув курсору);

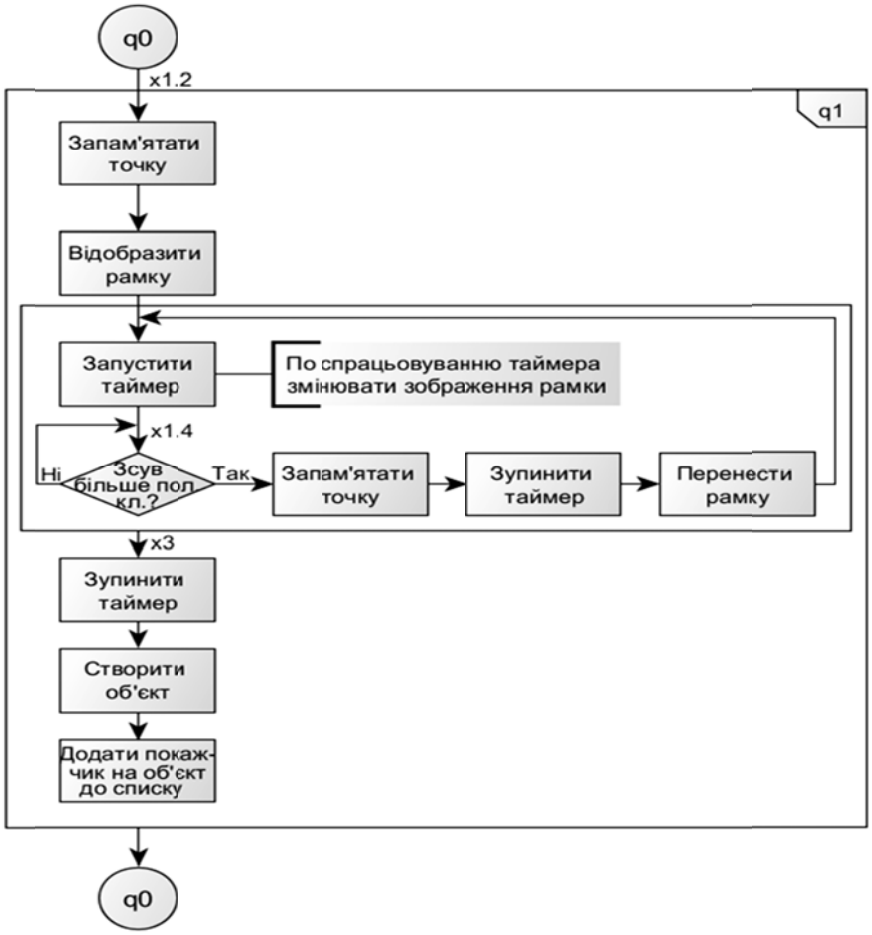

Рис. 4. Діаграма активності, що описує процес створення компонента на схемі СЕЕС

$q2$ – об'єкт не має фокуса. Більшість компонентів при переміщенні над ними курсору не отримують фокус, що є необхідними для здійснення багатьох операцій. Для вирішення цієї задачі при першому переміщенні курсору над об'єктом він встановлює на себе фокус та надсилає Робочому полю повідомлення «Зняття (втрата) фокуса» ($w1$). При отриманні цього повідомлення Робоче поле запам'ятовує ідентифікатор відправника, що дозволяє при наступному переміщенні курсору над Робочим полем зняти фокус з компонента (відсиланням повідомлення «Зняття фокуса»);

$q3$ – видалення компонента. Покажчики на об'єкти (відповідні компонентам) зберігаються в Робочому полі у вигляді списків (для кожного типа свій список). Видалення компонента передбачає руйнування його вікна, знищення об'єкта, звільнення пам'яті, яку він займає, та видалення покажчика з відповідного списку. При отриманні повідомлення про натискання клавіші Delete об'єкт може виконати лише першу дію; виконання наступних приведе до некоректного завершення його роботи. Тому при руйнуванні вікна об'єкт надсилає повідомлення Робочому полю «Видалити компонент» ($w6$), що дає змогу виконати дії, що лишилися, при першому переміщенні курсору над Робочим полем;

$q4$ – створення шини (при виборі в Бібліотеці компонентів);

$q5$ – створення шини (початком є об'єкт);

$q6$ – продовження шини.

Стани $q4 - q6$ стосуються операцій створення та продовження (з програмної точки зору – створення та приєднання до обраної) шини та є схожими. При вході в кожний з них виконується наступна послідовність дій (умовно блок entry):

– зміна курсору;

– переміщення курсору до вузла сітки;

– запам'ятовування положення курсору як початкової точки шини.

При вході до стану $q5$ додатково запам'ятовується ідентифікатор об'єкта, з якого починається шина (при закінченні створення шини Робоче поле має надіслати йому повідомлення Зміна шини з адресою створеної шини); $q6$ – ідентифікатор обраної шини. Поведінка класу в станах $q4 - q6$ (Блок do) є ідентичною та полягає в кресленні шини відрізками довжиною рівною кроку сітки. Зміна напряму креслення шини відбувається при зсуві курсору більше ніж на половину кроку сітки в напрямі Ox для вертикальної або Oy для горизонтальної поточної частини шини. На рис. 5 наведено логіку роботу класу в станах $q4 - q6$.

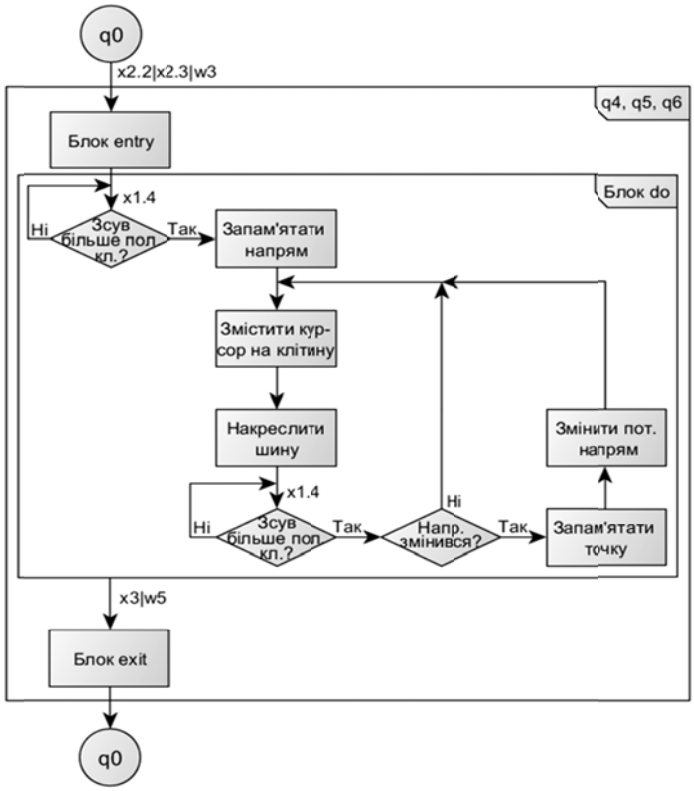

Рис. 5. Діаграма активності, що описує логіку роботу класу

Вихід зі станів $q4 - q6$ (блок exit) може бути ініційований відтисканням лівої кнопки миші на Робочому полі, шині (в цьому випадку створена шина приєднується до обраної) або на компоненті (Робочому полю приходить повідомлення «Кінець створення шини» ($w5$); як відповідь воно повинно надіслати компоненту повідомлення «Зміна шини» ($w2$) з адресою створеної шини). Якщо автомат знаходився в стані $q6$ (продовження шини) і шина була закінчена на іншій, то відбувається об'єднання двох шин (та, з якої був початок нової приєднується до тієї, на якій було відтискання), яке супроводжується відсиланням всім пов'язаним з шиною компонентам повідомлення «Зміна шини» ($w2$). Також при виході зі станів $q4 - q6$ відновлюється зображення курсору.

– *X = {x1.1, x1.2, x2.1, x2.2, x2.3, x3, w1, w3, w4.1, w4.2, w4.3, w5, w6}* – множина допустимих вхідних дій: *x1.1* – переміщення курсору над об'єктом; *x1.2* – переміщення курсору над об'єктом при затиснутій лівій кнопці миші (ЛКМ) та обраному компоненті для створення (окрім шини); *x2.1* – натискання ЛКМ; *x2.2* – натискання ЛКМ на Робочому полі при обраному компоненті для створення «Шина»; *x2.3* – натискання ЛКМ на шині; *x3* – відтискання ЛКМ; *w1* – повідомлення «Зняття фокусу»; *w3* – повідомлення «Початок створен-

ня шини»; *w4.1* – повідомлення «Початок переміщення об'єкта»; *w4.2* – повідомлення «Переміщення об'єкта на одну клітину»; *w4.3* – повідомлення «Кінець переміщення об'єкта»; *w5* –повідомлення «Кінець створення шини»; *w6* – повідомлення «Видалення об'єкта».

– $Y = \{ w1, w2\}$ – множина вихідних дій: *w1* – повідомлення «Зняття фокусу»; *w2* – повідомлення «Початок створення шини».

– f та g – функції станів та вихідних символів (представлені у вигляді графу на рис. 3).

Аналогічно наведеному створено автомат для реалізації поведінки Робочого поля у режимі обміну даними з контролерами автоматизації.

Більшість компонентів мають складну поведінку лише в режимі складання схеми. Тому класи, що її реалізують, та базові класи для компонентів містять лише один автомат. В режимі обміну даними ці компоненти виконують необхідні дії безпосередньо після отримання вхідного повідомлення. Виконання дій на дугах та петлях графів переходів відбувається в закритих методах класу, тобто нащадки базових класів наслідують їх поведінку та додають специфічні для компонента дії за допомогою нового автомату.

ВИСНОВКИ

Задачами програмного забезпечення верхнього рівня системи керування СЕЕС є контроль та управління СЕЕС. Використання об'єктно-орієнтованого програмування з явним виділенням станів для його створення обумовило проходження декількох етапів проектування: формулювання вимог до програмного забезпечення, створення його структури, виділення груп повідомлень між елементами програми та системою та користувачем, створення автоматів, що реалізують поведінку елементів програми. Це дозволило формалізувати опис статичних та динамічних властивостей об'єктів, що полегшило кодування та скоротило час верифікації програмного забезпечення.

ЛІТЕРАТУРА:

1. Gardner C. Networked Intelligent Instrumentation & Control for Switchboards / C. Gardner, D. Johnson, J. Provine // Electric Ship Technologies Symposium. – USA, 2007. – pp. 510-518.
2. Yang T. Present situation and development of power sytem simulation technologies / T. Yang // Automation of Electric Power Systems. – China, 2002. – №17. – pp. 23-47.
3. Velbitskiy I.V. Next generation visual programming technology / I.V. Velbitskiy // The materials of 11th IEEE east-west design & test symposium (IEEE EWDTS). – Rostov on Don, 2013. – pp. 404-410.
4. Khazaei H. Accuracy Evaluation of Delivered Measurements to HMI in a Real SCADA Automation System / H. Khazaei, H. Sheisi, H. Moradmand // Instrumentation Control and Automation. – 2011. – pp. 279–283.
5. Gamma H. Designing Concurrent, Distributed, and Real-Time Applications with UML / H. Gamma. – New York: Addison-Wesley, 2013. – 1060 p.
6. Babakov R.M. Structural representation of the synthesis process control machines operating with automatic transitions / R.M. Babakov, A.A. Barkalov // Control systems and machines. – 2011. – № 3. – pp. 47-53.
7. Kulyamin V.V. Komponentnaya arkhitektura sredy dlya testirovaniya na osnove modeley / V.V. Kulyamin // Programmirovaniye №5. – Moskva, 2010. – pp. 54-75.
8. ISO/IEC 12207-2008. System and software engineering – Software life cycle processes. – SC 7 System and Software Engineering, 2008. – 138 p.
9. Danilina T.G. Software quality evaluation with respect to international standards / Electronic and computer systems // – Kharkov : KHAI, 2012. – №7(59). – pp. 266-269.
10. Li. Juncao. Formalizing hardware/software interface specifications / Juncao Li, Fei Xie. V. Levin // The 26th IEEE Int. Conf. Automated Software Engineering (ASE), 2011. – pp. 235-252.

Рецензент: *д.т.н., проф. Рябенький В.М., завідувач кафедри теоретичної електротехніки та електронних систем НУК ім. адм. Макарова*