М.Ф. Єфремов, к.т.н., доц.
В.М. Єфремов, здобувач
Ю.М. Єфремов, к.т.н., доц.
Житомирський державний технологічний університет

СУТНОСТІ УНІФІКОВАНОЇЇ МОВИ ВІЗУАЛЬНОГО МОДЕЛЮВАННЯ

Робота присвячується розгляду сутностей мови UML (Unified Modeling Language).

Стереотип - новий вид елемента моделі, який успадковує структуру існуючого елемента, але має ряд додаткових обмежень, має іншу інтерпретацію і позначення. Стереотип містить набір тегів величин.

Тегів величина - це призначений для користувача атрибут, який визначається для елементів моделі, а не для об'єктів в працюючій системі. Вона, наприклад, може нести інформацію про управління проектом, вказівки по генерації коду та інші відомості.

Обмеження - формалізована умова, виражене текстовим рядком, за допомогою деякої мови обмежень. В UML використовується мова OCL (Object Constrained Language).

Профіль - набір стереотипів і обмежень з певним призначенням, який застосовується в призначених для користувача пакетах.

Корисно моделювати системи з трьох пов'язаних між собою точок зору: а) *модель класів* описує об'єкти системи і відношення між ними; б) *модель станів* описує історію існування об'єктів; в) *модель взаємодії* описує взаємодії між ними.

UML представляе собою мову візуального моделювання для вирішення різних завдань, який може застосовуватися при визначенні, візуалізації, конструюванні та документуванні артефактів, моделей або програмного середовища. UML не визначає конкретний процес розробки або концепцію аналізу і проектування. Внутрішні механізми UML, що утворюють мета модель мови, формують: *структуру ядра UML, яка уніфікована з компонентами MOF (Meta-Object Facility), що відноситься до концептуального моделювання; мета модель реструктуризована для усунення надлишкових конструкцій і забезпечення можливості створення нових підмножин UML в інших областях (наприклад, SysML); введення технології профілів, що дозволяє визначати специфічні для конкретних предметних областей і технологій розширення*.

В цілому, всі концепції і моделі UML можна віднести к чотирьом областям*: статична структура, елементи проектування, елементи розгортання, динамічну поведінку*. *Статична структура* дозволяє визначити повне безліч об'єктів, їх внутрішні властивості і відносини між собою. *Елементи проектування* є деякі конструкції моделі, призначені як для логічного аналізу, так і для проектування, забезпечує реалізацію систем. *Елементи розгортання* складаються з вузлів, артефактів та подання розгортання системи. Вузол - обчислювальний ресурс часу виконання, що визначає місцезнаходження виконуваних компонентів і об'єктів. *Артефакт* - фізична одиниця інформації або опис поведінки обчислювальної системи. Артефакти розгортаються в вузлах. Подання розгортання системи дозволяє описати конфігурацію вузлів системи і розташування артефактів в цих вузлах.

Після визначення основних концепцій UML стає можливим оцінити вплив методології на управління процесом розробки ПЗ і моделювання програмних компонент. Приділимо увагу основним етапам процесу розробки ПО.

Концептуалізація системи (system conception) означає виникнення «ідеї» додатка. На самому початку існування концепції хтось формує ідеологію, становить проект реалізації і пропонує її потенційному замовнику і користувачеві. На даному етапі розробник повинен точно представляти потреби потенційних споживачів продукту. Разом з тим має бути присутня точна оцінка технологічних можливостей розробників.

Аналіз (analysis) - за своєю суттю, об'єднує процеси дослідження та моделювання. Завдання аналітиків - дослідження і формалізація вимог. Виконується опис продукту, а не процесу або способу його реалізації. Аналіз ключова та складним завданням. Розробники повинні повністю усвідомити складність завдання, що ставляться перед ними завдання перед тим як зайнятися розв'язанням додаткових питань, які неминуче виникнуть на етапі реалізації. Надійні моделі - обов'язкова умова для створення розширеного, ефективного, надійного і коректного додатка. Ніякі виправлення на етапах реалізації не зможуть виправити внутрішню неузгодженість додатки і компенсувати недолік передбачливості.

Етап аналізу ділиться на дві стадії. Спочатку здійснюється *аналіз предметної області*, а потім - аналіз програми. Аналіз предметної області застосовується до об'єктів реального світу, семантику яких охоплює додаток. Аналіз предметної області виділяє поняття і відносини, а функціональність при цьому неявним чином міститься в моделі класів. Конструювання моделі предметної області зводиться, головним чином, до прийняття рішень про те, яку інформацію слід відобразити в моделі і як її потрібно представити. За аналізом предметної області слідує аналіз додатка , який відноситься до комп'ютерних аспектів програми, видимим користувачам. Модель програми не наказує конкретної реалізації системи. Вона описує зовнішній вигляд програми, яке сприймається як чорний ящик. Класи додатки можна визначити на етапі аналізу, але їх часто можна взяти з попередніх програм.

В процесі *проектування системи* (system design) розробник приймає стратегічні рішення з найбільш широким спектром наслідків. Він повинен сформулювати архітектуру системи і вибрати глобальні стратегії і політики, що визначають наступні, більш деталізовані етапи проектування.

Архітектура - це високорівневий план, або стратегія вирішення завдання зі створення програми. Її вибір визначається вимогами і великим життєвим досвідом розробника. По можливості, архітектура повинна включати

виконуваний скелет, який можна було б тестувати. Проектувальник повинен представляти, яким чином нова система буде взаємодіяти з уже існуючими.

На етапі проектування класів (class design) розробник розширює і оптимізує аналітичні моделі. Відбувається зсув точки докладання зусиль з концепцій додатки на комп'ютерні концепції. Розробники вибирають алгоритми реалізації основних функцій системи, проте вони повинні утримуватися від вибору конкретних мов програмування.

Реалізація (implementation) - це етап написання реального коду. Розробники реалізують елементи проекту на мовах програмування, створюючи програмний код і структуру бази даних. Досить часто код може частково генеруватися автоматично з проектної моделі.

Після реалізації система завершена, але до експлуатації вона повинна пройти обов'язковий етап *тестування*. Тестування дозволяє також розкрити випадкові помилки, які з'явилися в системі в процесі її створення. Якщо програма має працювати на різному обладнанні або під різними операційними системами (на різних платформах), воно повинно бути перевірено на всіх цих платформах.

Організація повинна навчати користувачів, щоб вони могли повністю використовувати всі переваги нової програми. Навчання прискорює засвоєння користувачами нових навичок. Окрема команда повинна готувати документації для користувача паралельно з розробкою програмного забезпечення. Відділ контролю якості може порівнювати програмне забезпечення з призначеної для користувача документації. Це дозволяє гарантувати, що програма відповідає вихідним вимогам.

При розгортанні системи (deployment) на підприємстві користувача можна очікувати різних ефектів взаємодії з іншими системами, платформами і конфігураціями. Розробники повинні оптимізувати систему під різні навантаження, оформити сценарії і процедури установки. Деякі клієнти вимагатимуть настройки системи під себе. Персонал повинен також виконати локалізацію продукту на різні мови з урахуванням користувальницької локалі (locale - змінна, яка визначає призначені для користувача дані локалізації, тобто специфічні позначення одиниць виміру, формати дат і часу, і т.д.). В результаті відбувається випуск продукту (release).

Після завершення розробки і розгортання системи команда переходить до етапу підтримки. Підтримка має на увазі кілька видів діяльності. В процесі експлуатації виявляються помилки, що містилися в програмному забезпеченні і не виявлені на етапі тестування. Ці помилки необхідно виправляти. Успішне додаток викликає запити на удосконалення, а довго живучий додаток - на реструктуризацію.

Висновок. У даній роботі проведено аналіз сутностей мови візуального моделювання.