УДК 004.415.2.925.8

Биков Д. П., Фісун М. Т.

ПОШАРОВА МОДЕЛЬ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ САПР СУДЕН

У статті розглядаються різні моделі побудови програмного забезпечення САПР корпуса судна: монолітна, пошарова, клієнт-серверна, об'єктно-орієнтована тощо. Відзначено переваги й недоліки кожного з підходів. Показано, що найефективнішим підходом є комбінація наведених вище моделей програмного забезпечення, коли на верхньому рівні обирається пошарова модель, усередині кожного шару використовується об'єктно-орієнтована модель, а для забезпечення взаємозв'язків між процесами використовується клієнт-серверна модель. Запропоновано й розкрито зміст п'яти шарів: бази даних, атрибутів, геометрії, графіки, ядра. Такий підхід було застосовано при створенні САПР ДЕЙМОС, окремі версії якої пройшли перевірку при виконанні робіт з планово-технологічної підготовки суднобудівного виробництва для конкретних проектів суден.

Вступ

Програмне забезпечення (ПЗ) за існуючими нормативними документами [1, 2 та ін.] традиційно поділяється на дві частини - системну і прикладну. Частину системного ПЗ становлять операційні системи (ОС), утиліти для підтримки працездатності й надійності всієї системи, засоби копіювання і відновлення та інше, тобто те ПЗ, що необхідне для всіх комп'ютерів незалежно від функціонального призначення комп'ютера. Частину прикладного програмного забезпечення становить ПЗ, призначене для розв'язання конкретних функціональних задач, пов'язаних з діяльністю людини, що експлуатує певний комп'ютер. Це може бути будь-який пакет програм з бухгалтерського обліку, наприклад «Бухгалтерія 1С» [3], якщо комп'ютер встановлено у бухгалтерії і призначено для виконання бухгалтерських функцій, чи система автоматизованого проектування суден, наприклад САПР ДЕЙМОС (ДЕталювання Й МОделювання Систем) [4], якщо комп'ютер призначено для виконання проектних робіт у конструкторському бюро. Але з погляду розробника програмного забезпечення такий розподіл не зовсім відповідає суті. Розробник програмного забезпечення для своєї роботи використовує як системне, так і прикладне ПЗ, і сам створює *нове* ПЗ. Тому з його погляду ПЗ можна, очевидно, поділити на дві частини - *базову* і *прикладну*. До базового ПЗ входять ОС, інструментальні засоби для створення ПЗ (різні оболонки - Visual Studio, Borland, Oracle тощо), різні бібліотеки програм інших розробників.

У процесі еволюції методів створення прикладного програмного забезпечення (ППЗ) було розроблено чимало методологій і технологій розробки ППЗ, але вибір конкретної методології або їх сполучень, виходячи з таких критеріїв, як вартість, тривалість розробки і мобільність ППЗ, залишається, як завжди, актуальним завданням.

1. Основна мета і завдання дослідження

При розробці САПР ДЕЙМОС, у створенні якої автори брали участь, основною метою було досягнення високого рівня мобільності ППЗ, щоб в разі потреби САПР ДЕЙМОС могла бути перенесена на іншу платформу (як на іншу ОС, так і на інші процесори). Для досягнення зазначеної мети потрібно було:

- проаналізувати сучасні моделі побудови програмного забезпечення САПР, виявити їх переваги й недоліки;
- обрати найефективнішу комплексну модель ППЗ, використовуючи переваги кожної окремої моделі;
- впровадити комплексну модель ППЗ при створенні САПР ДЕЙМОС;

• перевірити ефективність розробленого за обраною моделлю ППЗ у виробничих умовах.

2. Аналіз переваг і недоліків окремо взятих моделей програмного забезпечення

На сьогодні існують кілька моделей організації ПЗ: монолітна, пошарова, клієнт-серверна, об'єктно-орієнтована тощо [5, 6 та ін.]. Розглянемо основні концепції цих моделей.

2.1. Монолітна модель

Існує багато способів структурування коду програмного забезпечення. Один з підходів, який особливо часто застосовується при створенні невеликих прикладних програм, полягає в організації системи як набору процедур, кожну з яких може викликати будь-яка процедура користувача (рис. 1).

Така монолітна структура не забезпечує ізоляції даних; у різних ділянках коду використовується інформація про архітектуру всієї системи. Тиражування такого програмного продукту здійснювати важко, оскільки зміна певної процедури може спричинити помилки в інших частинах системи, які, на перший погляд, не мають до неї відношення. Крім того, налагодження і пошук помилок у таких системах завжди тривають досить довго. Перенесення ж такого програмного продукту на іншу платформу взагалі стає складною задачею.

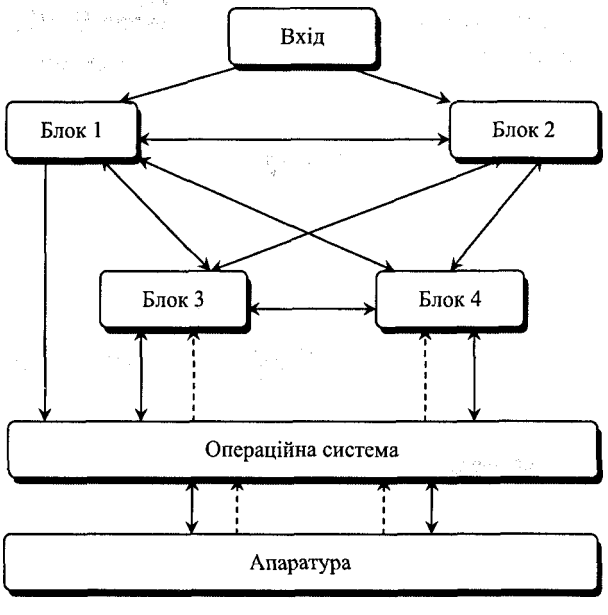

Рис 1. Приклад монолітної структури ПЗ

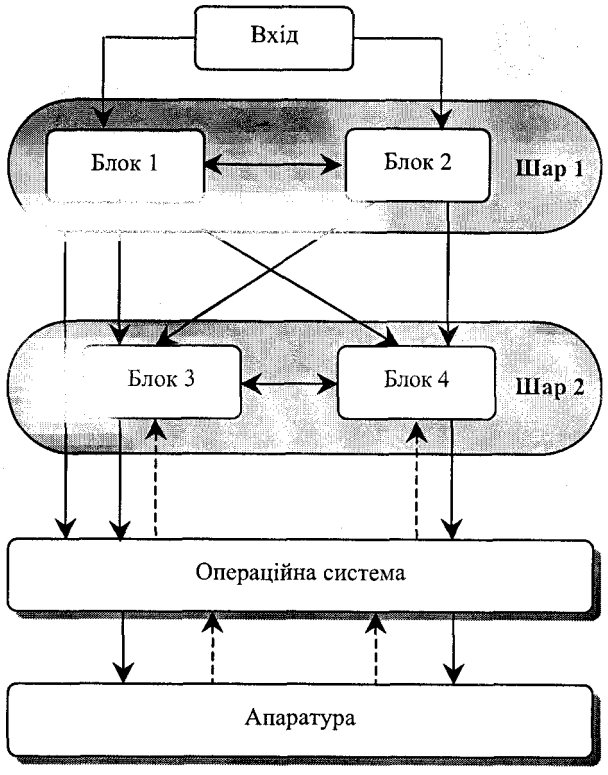

Рис 2. Пошарова структура ПЗ

2.2. Пошарова модель

Інший підхід до структурування системи передбачає поділ її на модулі, що нашаровані один над іншим. Кожний модуль реалізує певний набір функцій, що можуть викликатися іншими модулями. Код, розміщений на тому чи іншому шарі, викликає код тільки з нижчих шарів. У деяких випадках шаруватість примусово обумовлена апаратурою та ОС, наприклад VAX/VMS [7].

Однією з переваг пошарової організації коду є те, що код кожного шару одержує доступ тільки до потрібних йому інтерфейсів (і структур даних) нижчих шарів. Таким чином, зменшується обсяг коду, що має необмежену владу. Крім того, така структура дає змогу при налагодженні ОС починати з найнижчого шару й додавати по одному шару доти, доки вся система не стане працювати правильно. Пошарова структура полегшує і розширення системи: можна цілком замінити будь-який рівень, не торкаючись інших частин (рис. 2).

2.3. Клієнт-серверна модель

Третій підхід до структурування коду — це модель *клієнт-сервер*. Ідея його полягає в поділі програми на кілька процесів, кожний з яких реалізує один набір сервісів: наприклад, розподіл пам'яті, обслуговування бази даних, обслуговування обміну даними між користувачами, візуа-

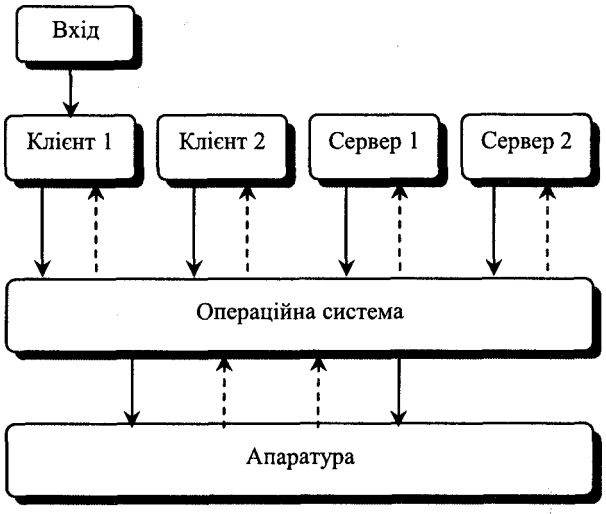

Рис 3. Клієнт-серверна структура ПЗ

лізація об'єктів. Кожний *сервер* (server) виконується в користувальницькому режимі, перевіряючи в циклі, чи не звернувся до нього по обслуговування який-небудь клієнт. *Клієнт* (client), яким може бути або інший компонент системи, або прикладна програма, запитує виконання сервісу, посилаючи серверу повідомлення. Ядро (чи мікроядро) програми доставляє повідомлення серверу, і той виконує запит дії, після чого ядро повертає клієнту результати у вигляді іншого повідомлення, як це показано на рис. 3.

При використанні клієнт-серверного підходу отримуємо систему, що складається з автономних компонентів невеликого розміру. Оскільки всі сервери виконуються як окремі процеси режиму користувача, аварія (можливо, перезапуск) одного з них не порушує роботи інших частин системи. Більше того, різні сервери можуть виконуватися на різних процесорах багатопроцесорного комп'ютера чи навіть на різних комп'ютерах, що робить систему придатною для розподілених обчислювальних середовищ.

Використання моделі клієнт-сервер дає такі переваги:

• *Підвищується надійність.* Кожен сервер виконується як окремий процес, що має власну пам'ять і, отже, захищений від інших процесів. Більше того, завдяки тому, що сервери працюють у режимі користувача, вони не мають безпосереднього доступу до апаратури й не можуть змінити вміст ділянок пам'яті, що належать іншим процесам.

• *Якнайкраще відповідає моделі розподілених обчислень.* Оскільки мережні комп'ютери, що використовують модель «клієнт-сервер», спілкуються один з одним за допомогою передачі повідомлень, локальні сервери можуть легко посилати повідомлення віддаленим машинам при обробці запитів від клієнтських програм. Клієнтам не потрібно знати, обслуговується їхній запит локально чи віддалено.

2.4. Об'єктно-орієнтована модель

Основна мета розробки системи з орієнтацією на дані – це створення мобільного програмного забезпечення, яке можна було б просто й дешево змінювати. Наскільки важливою є здатність до модифікації, стає ясно, якщо взяти до уваги статистику, за якою 70 % ціни програмних продуктів припадає на їх супровід, який містить у собі додавання нових можливостей, модифікацію форматів даних, виправлення помилок і адаптацію до нового обладнання. А це зумовлює зміни в програмах.

Один із способів мінімізації необхідних змін в об'єктно-орієнтованих програмах – це приховування фізичного представлення даних усередині об'єктів [8]. *Об'єкт* – це структура даних, фізичний формат якої схований у визначенні типу. Він має набір властивостей, так званих *атрибутів*, і з ним працює група сервісів. Крім того, що зменшується вплив змін, побудова системи на основі об'єктів має ще ряд безсумнівних переваг:

• *Доступ системи до ресурсів і робота з ними уніфіковані.* Створення, видалення і посилання на об'єкт «конструкція» здійснюється так само, як і на об'єкт «точка» з використанням дескрипторів об'єктів.

• *Спрощується захист, оскільки для всіх об'єктів він здійснюється однаково.* При спробі доступу до об'єкта підсистема захисту втручається і перевіряє допустимість операції незалежно від того, чи є об'єкт конструкцією, точкою, чи конструктивною лінією.

3. Вибір методології побудови ППЗ при створенні САПР суден

Таким чином, у кожної з цих моделей є свої переваги й недоліки, тому найефективнішим рішенням у виборі моделі програмного забезпечення є їхня комбінація. Саме цей підхід застосовується в таких сучасних САПР суден, як **FORAN, TRIBON, CATIA** та інших. При створенні програмного забезпечення вітчизняної САПР ДЕЙМОС теж було використано комбінацію наведених вище моделей ППЗ, при цьому було обрано певну послідовність і взаємозв'язок моделей.

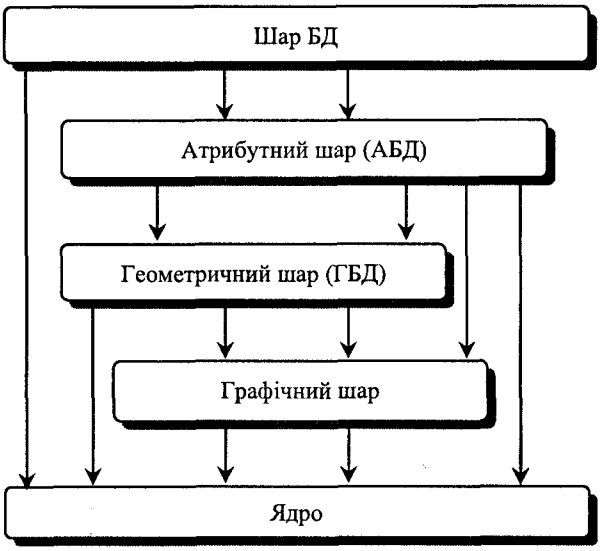

Рис 4. Пошарова модель побудови САПР ДЕЙМОС

Таблиця 1. **Структура об'єкта «ядро»**

Призначення поля	Тип даних	Розмір
Унікальний ключ об'єкта	Int	4 байти
Покажчик блоку даних елемента	void*	4 байти
Розмір блоку в байтах	Int	4 байти
Режим створення (Temp – тимчасовий, Const – постійний, Work – робочий у вказівці)	Enum	4 байти
Властивості об'єкта (видимий / невидимий, що вказується / що не вказується, обраний / недобраний, об'єкт АБД / об'єкт ГБД тощо)	UINT	4 байти

На верхньому рівні розробки ПЗ за використаною методологією обирається пошарова модель, а всередині кожного шару використовується об'єктно-орієнтована модель. Для взаємозв'язку між процесами краще використовувати клієнт-серверну модель. Пошарова модель складається з п'яти шарів (рис. 4):

• *Шар бази даних (БД)*. Відповідає за функції обміну даними між САПР та системою керування базою даних (СКБД). У разі переходу на іншу СКБД слід переписати лише цей шар.

• *Атрибутний шар*. У термінології САПР ДЕЙМОС цей шар ще називають *Атрибутна база даних (АБД)*. Цей шар інкапсулює в собі об'єкти та функції для роботи з фізичними властивостями елементів об'єкта, що проектується.

• *Геометричний шар*. У термінології САПР ДЕЙМОС цей шар має назву *Геометрична база даних (ГБД)*. Відповідає за математичну модель об'єкта, що проектується.

• *Графічний шар*. Основне призначення цього шару – відображення елементів або всього об'єкта, що проектується.

• *Ядро*. Це найнижчий рівень. Інкапсулює в собі функції керування пам'яттю та процесами.

Розглянемо детальніше функції кожного з перелічених шарів, починаючи з нижнього.

3.1. Шар ядра ППЗ САПР ДЕЙМОС

Операційні системи відрізняються одна від одної в першу чергу засобами керування пам'яттю і процесами. Тому ці функції звичайно покладають на частину програмного забезпечення, так зване *ядро*. Цей шар є найнижчим і нічого «не знає» про моделі даних, розміщених вище. Функції вищих шарів звертаються до нього в разі потреби виконання будь-яких сервісів. Таким чином, до функцій ядра належать функції створення, видалення, зміни блоку пам'яті, в якому зберігається об'єкт (при цьому ядро «не знає», який саме об'єкт воно обслуговує), і функції керування процесами (у випадку з платформою Windows NT – потоками). У табл. 1 наведено структуру об'єкта, з яким оперує ядро.

Шар ядра оперує блоками пам'яті з інформацією, що необхідна вищим шарам.

Сучасна САПР не може функціонувати в одному процесі. Для ефективнішої роботи програми краще розпаралелювати, тобто деякі функції переносити в рівнобіжний процес (потік). Особливо якщо ці функції становлять розрахункові задачі з великим часом виконання чи, навпаки, маленьку допоміжну задачу, що здійснює лише сервісні функції. Наприклад, функція керування курсором повинна виконуватися в рівнобіжному процесі (потоці) і, крім відображення курсора на екрані, повинна виводити реальні координати в допоміжні вікна.

3.2. Шар бази даних

САПР неможлива також і без використання системи керування базами даних. При розробці САПР ДЕЙМОС було обрано СКБД dbVista [9]. Задачі роботи з базою даних досить об'ємні за часовими характеристиками, з одного боку, а з іншого – ці задачі повинні взаємодіяти з процесами інших користувачів на інших комп'ютерах.

Для ефективної організації роботи СКБД цілком природним буде виділення функцій роботи з базою даних так само в рівнобіжний процес (потік). У табл. 2 наведено структуру об'єкта «процес».

Таблиця 2. **Структура об'єкта «процес»**

Призначення поля	Тип даних	Розмір
Покажчик робочого потоку	CWinThread*	4 байти
Потокова функція	UINT (*proc)(CD2Doc*)	4 байти
Покажчик події	CEvent*	4 байти
Код операції	UINT	4 байти
Покажчик об'єкта поточного немодального діалогового вікна	CDialog*	4 байти
Прапорець припинення виконання поточної операції (завершення потоку)	Bool	2 байти
Текст підказки в StatusBar	char msg[120]	120 байтів
Для повернення фізичних координат	double xy[3]	24 байта
Для повернення параметра на лінії	Double	8 байтів
Для повернення індексу визначеного об'єкта	Int	4 байти
Для повернення введеного рядка	char*	4 байти
Маска для пошуку визначеного об'єкта	UINT	4 байти
Маска натиснутих клавіш при вказівці	UINT	4 байти
Код завершення операції	Int	4 байти
Ознака зациклення операції	Bool	2 байти

3.3. Шар геометричного моделювання

Над ядром розміщується ще один шар – геометричний. У цьому шарі зібрано програми зі створення, зміни, видалення геометричних об'єктів і одержання різної інформації, пов'язаної з ними (наприклад, розрахунків відстаней, кутів тощо). Перелік елементарних геометричних об'єктів і задач залежить від САПР – для одних цілком достатньо прямих ліній і дуг, для інших необхідні криві (сплайни), для одних досить площин і простих поверхонь (циліндр, конус, куля), для інших потрібні складніші типи поверхонь (каркасна, лекальна, лінійчаста тощо).

Крім того, одним з головних завдань систем автоматизації проектних робіт є задача створення геометричної (математичної) моделі об'єкта. Крім використання її для геометричного моделювання і підготовки відповідної графічної інформації, на базі цієї геометричної моделі повинні виконуватися різні інженерні розрахунки. Для моделювання будь-якого просторового тіла цілком очевидна така ієрархія об'єктів:

1. Точка.
2. Лінія.
   2.1. Пряма.
   2.2. Коло.
   2.3. Сплайн.
3. Площина.
4. Поверхня.
5. Текст (по суті, це не геометричний об'єкт, але для візуалізації надалі він необхідний).

Цей перелік становить множину елементарних геометричних об'єктів. Для того щоб за допомогою цих об'єктів змоделювати тіло, потрібен ще один абстрактний тип геометричного об'єкта – комплекс. Цей об'єкт поєднує в собі і встановлює правила об'єднання як елементарних об'єктів, так і інших комплексів. Крім того, кожен об'єкт, крім об'єкта «точка», повинен мати обмеження. Обмеженнями можуть служити як деякі параметри (лінії, площини і параметризовані поверхні), так і посилання на інші об'єкти. Природно, що в цьому випадку найефективнішим засобом для керування як комплексами, так і обмеженнями, є використання списків. Комплекс містить два посилання: перше – на список об'єктів, з яких він складається, друге – на список комплексів, до яких він входить. Список комплексів, як і список обмежень (для комплексів – список вхідних об'єктів) – спеціальний допоміжний об'єкт для організації ієрархії об'єктів.

Далі, об'єкт може бути перетинанням двох інших об'єктів – і ця інформація може знадобитися для розв'язання певних геометричних задач. Таким чином, цей об'єкт повинен мати математичні моделі в координатах своїх власників (носіїв). Скажімо, якщо точка є перетинанням двох ліній, у ній мають бути наявними математичні описи в координатах обох ліній. Якщо лінія є перетинанням двох поверхонь, у ній мають бути наявними математичні описи в координатах поверхонь і т. д.

Таблиця 3. **Структура елементарного геометричного об'єкта**

Призначення поля	Тип даних	Розмір
Тип об'єкта	Char	1 байт
Кількість носіїв	Char	1 байт
Список границь	Void *	4 байти
Список комплексів	Void	4 байти
Довжина математичної моделі в 1-му носії	Long	4 байти
Математична модель у 1-му носії		
…		
Довжина математичної моделі в *n*-му носії	Long	4 байти
Математична модель у *n*-му носії		

Таким чином, у загальному випадку, елементарний геометричний об'єкт становить блок пам'яті, в якому зберігається певна інформація (див. табл. 3).

3.4. Шар атрибутів

Для того щоб геометрична (математична) модель об'єкта не залишилася звичайним, абстрактним рівнянням і її можна було використовувати для подальших технологічних задач, необхідно, щоб крім шару з описом геометрії був шар, у якому б зберігалася інформація про фізичні властивості. У цьому шарі можливе створення ієрархій не з погляду математики, а з погляду технології [10]. Таким чином, вимоги до фізичної моделі (її ще називають атрибутною, тому що ця модель прив'язується до математичної моделі й у ній зберігаються фізичні атрибути) можна сформулювати так: вона повинна:

– вміти посилатися до геометричного шару;

– підтримувати ієрархічну і, більше того, мережну модель даних;

– вміти зберігати дані.

Таблиця 4. **Структура атрибутного об'єкта**

Призначення поля	Тип даних	Розмір
Тип об'єкта	Char	1 байт
Покажчик списку верхніх об'єктів	CAttrib *	4 байти
Покажчик списку нижніх об'єктів	CAttrib *	4 байти
Довжина блоку з даними	UINT	4 байти
Покажчик блоку з даними	Void *	4 байти

Шар АБД для підтримки вищеозначених функції оперує об'єктами, структуру яких наведено в табл. 4.

Для відображення об'єктів потрібен ще один шар – графічний. Цей шар найбільш немобільний, оскільки графіка цілком залежить і від апаратури, і від операційної системи. Тому він потребує особливої уваги. Для відображення об'єкта необхідні такі характеристики, як тип зображення і колір. А способи зображення залежатимуть від апаратних можливостей і операційної системи.

4. Упровадження запропонованої пошарової моделі ППЗ

Використовуючи такий підхід, було побудовано САПР ДЕЙМОС. За базове програмне забезпечення обрано оболонку Microsoft Visual Studio і мову програмування C++. Це зумовлено вимогами мобільності – в разі потреби САПР ДЕЙМОС може бути перенесена на іншу платформу (як на іншу ОС, так і на інші процесори). Але для забезпечення мобільності і незалежності від ОС було вжито спеціальних заходів щодо організації структури програмного забезпечення ДЕЙМОС, методологічні принципи яких викладено вище.

Усе програмне забезпечення поділяється на кілька виконуючих модулів (exe-файлів) та динамічних бібліотек (dll-файлів), деякі модулі повністю збігаються з шаром у пошаровій моделі (див. табл. 5).

Таблиця 5. **Перелік файлів та функцій модулів системи ДЕЙМОС**

Файл	Функція
D2.exe	Головний модуль. Виконує всі дії з ініціалізації та керування системою. У цьому модулі зібрано шар ядра та реалізовані всі діалоги
3Dviewer.exe	Окремий модуль для огляду 3D-моделі з урахуванням невидимості. Багато функцій щодо зміни точок зору, масштабування та підсвічування моделі
Control.exe	Транслятор керуючих програм для верстатів з числовим програмним керуванням. Цей модуль призначений для випуску керуючих програм у різних форматах. Також він може переводити керуючі програми з одного формату в інший
GeomObject.dll	У цій бібліотеці зібрані функції двох шарів: геометричного та атрибутного
Database.dll	Ця бібліотека призначена для шару роботи з базою даних, де зберігається модель судна

Продовження табл. 5

Файл	Функція
DetalDB.dll	У цій бібліотеці знаходяться функції роботи з БД «Деталь»
MaterialDB.dll	Призначений для роботи з довідником матеріалів
Opengl32.dll	Функції, що розташовані в цій бібліотеці, відповідають графічному шару і використовуються для отримання графічних зображень
DeymosImage.dll	Ця бібліотека зберігає різні зображення, що використовуються системою (наприклад, зображення типових деталей, вирізів та інше)

Таким чином, при використанні комбінації різних моделей програмного забезпечення системи ДЕЙМОС домінуючою обрано пошарову модель, що надало цій САПР високу мобільність. Побудована на цій основі методологія і мобільне програмне забезпечення були впроваджені на багатьох підприємствах СНД. Тільки у НДІ «Центр», який є головною організацією Мінпромполітики України у створенні САПР суден, з її допомогою було розроблено понад 30 проектів суден, побудованих на вітчизняних та зарубіжних суднобудівних підприємствах. Завдяки такій архітектурі система постійно модернізується відповідно до вимог різних замовників. У систему легко вбудовуються найновіші пакети ПЗ. Так, нині виконується перехід з графічного ядра *openGL* [11] на графічне ядро *ACIS* [12].

1. ГОСТ 34.201-89. Комплекс стандартов на автоматизированные системы. Техническое задание.– М.: Госстандарт СССР, 1989.– 67 с.
2. ДСТУ ISO 9000-3-98. Настанови щодо застосування ДСТУ ISO 9001 під час розроблення, постачання та супроводження програмного забезпечення.– К.: Держстандарт України, 1998.– 132 с.
3. *Харитонов С. А.* 1С: Компьютерная бухгалтерия 7.7 в системе гибкой автоматизации бухгалтерского учета. Серия «Азбука бухгалтера».– К.: BHV, 2000.– 528 с.
4. Система автоматизированного проектирования судов ДЕЙМОС. Методика построения составной поверхности. Руководство пользователя / *Александров А. П., Быков Д. П., Кирпа В. А.* и др.– Николаев: ОАО «НИИ Центр», 2002.– 84 с.
5. Основы NT и NTFS / Пер. с англ. – М.: Изд. отдел «Русская редакция» ООО «Channel Trading Ltd.», 1996.– 440 с.
6. *Потемкин А.* Трехмерное твердотельное моделирование.– М.: КомпьютерПресс, 2002.– 296 с.
7. *Levy H. M., Eckhouse R. H.* Computer Programming and Architecture: The VAX-11. Bedford : Digital Press MA, 1980.– 153 с.
8. *Айра Пол.* Объектно-ориентированное программирование на C++.– СПб.: Бином; Невский Диалект, 2001.– 464 с.
9. Про організацію структури бази даних у САПР корпусу судна / Биков Д. П., Фісун М. Т. // УСіМ.– 2002, № 6.– С. 37–41.
10. *Шпур Г., Краузе Ф.-Л.* Автоматизированное проектирование в машиностроении.– М.: Машиностроение, 1988.– 648 с.
11. OpenGL Reference Manual. – Addison-Wesley Publishing Company, 1992.– 357 с.
12. ACIS Geometric Modeler. Format Manual. – Spatial Technology, Inc. 1999.– 412 с.

D. Bykov, M. Fisun

LEVEL-BY-LEVEL MODEL FOR CREATING OF SOFTWARE OF CAD-SHIP

Different models of software for CAD-system of hull are considered: monolithic, level-by-level, client-server, object-oriented etc. Advantages and disadvantages of each approach are shown. It is shown, that the most effective approach is the combination of above mentioned models of the software when at the top level the level-by-level model gets out, in the middle of each level the object-oriented model is used, and for providing connections between processes the client – server model is used. The contents of five levels are offered and described: database, attributes, geometry, graphics and nucleus. The described approach was applied at creation CAD-system «ДЕЙМОС» which separate versions have passed check at performance of works on loft production tooling of ship-building jards for concrete projects of vessels.