 [Translated from Russian to Ukrainian - www.onlinedoctranslator.com](https://www.onlinedoctranslator.com/en/?utm_source=onlinedoctranslator&utm_medium=docx&utm_campaign=attribution)

## 2. Системотехніка обчислювальних систем

**Цілі**

Мета цієї глави – дати запровадження системотехніку обчислювальних систем і пояснити, чому знання у цій галузі важливі спеціалістів із програмного забезпечення. Прочитавши цей розділ, ви повинні:

* розуміти, чому програмне забезпечення дедалі ширше застосовується у розробках системотехніки;
* знати основні інтеграційні характеристики систем: безвідмовність, продуктивність, надійність та безпека;
* розуміти, чому у процесі розробки систем необхідно враховувати оточення, у якому працюватиме система;
* мати уявлення про процеси створення систем та системного забезпечення.

Системотехніка як технологія створення систем охоплює процеси створення специфікацій, проектування, розробки, тестування, впровадження та супроводу систем як єдиного цілого. Системотехнік, що займається розробкою обчислювальних систем, не зосереджений лише на програмному забезпеченні, він приділяє однакову увагу програмному забезпеченню, апаратним засобам та засобам взаємодії з користувачами та системним оточенням. Він повинен думати про ті функції, які виконуватиме система і, власне, заради яких будується система, а також про взаємодію системи з її оточенням. Фахівець зі створення програмного забезпечення повинен розуміти завдання системотехніки, оскільки проблеми, що виникають, часто є результатом рішень, прийнятих системотехніками.

Існує безліч найрізноманітніших визначень поняття "система", від дуже абстрактних до надзвичайно конкретних. На мою думку, найбільш вдалим визначенням системи буде наступне.

*Система – це сукупність взаємодіючих компонентів, які працюють спільно задля досягнення певних цілей.*

Це визначення охоплює дуже широке коло систем. Наприклад, така проста система, як олівець, складається з двох або кількох "апаратних" компонентів, у той час як система управління польотами може складатися з тисяч апаратних та програмних компонентів плюс оператори, які приймають рішення на основі даних, що надаються інформаційними підсистемами.

Визначальною ознакою системи є те, що властивості та поведінка системних компонентів впливають одна на одну надзвичайно складним та заплутаним чином. Коректне функціонування кожного системного компонента залежить від багатьох інших компонентів. Так, операційне забезпечення може виконувати свої функції, якщо працює процесор і відповідні апаратні засоби. Процесор може виконувати обчислення, якщо коректно інстальовано програмні системи, що задають ці обчислення.

Системи мають ієрархічну структуру, тобто. як компоненти містять інші системи. Наприклад, комп'ютерна система поліції містить географічну інформаційну систему, яка пропонує інформацію про місце, де трапилося або може статися якась подія. Системи, що є компонентами інших систем, називають підсистемами. Визначальна властивість підсистем полягає в тому, що вони можуть функціонувати самостійно, незалежно від систем, до складу яких входять. Тому географічну інформаційну систему можна використовувати також інших системах. Разом про те їх поведінка у складі будь-якої конкретної системи залежить, звісно, ​​від взаємодії коїться з іншими підсистемами.

Складність взаємодії між системними компонентами означає, що система не зводиться до сумі її складових частин. Вона має певні властивості, властиві їй саме як цілісній системі. Такі інтеграційні властивості неможливо знайти властивостями будь-якої окремої частини системи. Більше того, вони виявляються тоді, коли система сприймається як єдине ціле. Деякі з цих властивостей можна вивести з аналогічних властивостей окремих підсистем, але частіше є комплексним результатом взаємодії всіх підсистем і їх неможливо оцінити, виходячи з аналізу окремих системних компонентів.

Наведемо приклади інтеграційних властивостей.

1. *Сумарний обсяг системи.*Це приклад інтеграційного показника системи, який можна обчислити, виходячи лише з властивостей окремих компонентів.

2. *Безвідмовність системи.*Ця властивість залежить від безвідмовності окремих компонентів та взаємозв'язку між ними.

3. *Зручність експлуатації системи.*Це дуже складна багатопараметрична властивість, яка залежить не тільки від програмного забезпечення та апаратних засобів системи, але також від оточення, в якому експлуатується система, та від системних операторів.

У цій книзі основна увага приділяється обчислювальним системам, які складаються з апаратних та програмних компонентів та, як правило, мають підсистеми для взаємодії з людиною. Фахівець із програмного забезпечення повинен знати системотехніку обчислювальних систем, оскільки тут програмний компонент відіграє важливу роль. Наприклад, у 1969 році для здійснення посадки людини на Місяць у програмі "Аполлон" використовувалося ПЗ, що займає всього 10 Мбайт, а ПЗ космічної станції - 100 Мбайт. Таким чином, технології інженерії програмного забезпечення часто є критичним фактором розробки складних обчислювальних систем.

### 2.1. Інтеграційні властивості систем

Як зазначалося вище, інтеграційні властивості систем виявляються лише тоді, коли система сприймається як єдине ціле. У цьому полягає складність прогнозування та оцінки таких властивостей, оскільки іноді можна виміряти показники лише підсистем, у тому числі складається комплексна система.

Існує два типи інтеграційних властивостей.

1. Функціональні властивості, що виявляються лише тоді, коли система працює як єдине ціле. Наприклад, велосипед має функціональні властивості транспортного засобу лише тоді, коли зібраний із своїх компонентів.

2. Нефункціональні властивості: безвідмовність, продуктивність, безпека та захищеність (обмеження несанкціонованого доступу до системи), які залежать від поведінки системи в операційному оточенні. Такі властивості часто критичні для обчислювальних систем, оскільки якщо вони не досягають певного мінімального рівня, система не буде працездатною. Деякі функції та можливості системи можуть бути не затребувані всіма користувачами, тому система може бути працездатною і без них. Разом про те система, не надійна чи ефективна у своїх окремих функціях, однаково вважається бракованою.

Щоб проілюструвати складність визначення інтеграційних властивостей, розглянемо такий показник системи, як безвідмовність. Це комплексний показник, який слід розглядати лише на рівні системи, а чи не її окремих компонентів. Компоненти в системі взаємопов'язані, так що збій в одному компоненті може поширитися по всій системі і викликати реакцію у відповідь в інших компонентах. Проектувальники систем часто не можуть передбачити послідовність поширення збоїв у системі, тому важко оцінити безвідмовність системи лише на підставі даних про безвідмовність її окремих компонентів.

Існує три тісно пов'язані між собою фактори, які впливають на загальну безвідмовність системи.

1. *Безвідмовність апаратних засобів.*Цей показник визначається ймовірністю виходу з ладу окремих апаратних компонентів і часом, необхідним їх заміну.

2. *Безвідмовність програмного забезпечення.*Це показник роботи компонента ПЗ без збоїв та помилок. Програмні помилки зазвичай впливають на апаратні засоби системи. Тому система може продовжувати функціонувати навіть тоді, коли програмне забезпечення видає некоректні результати. Безвідмовність програмного забезпечення докладно розглядається у розділах 16 та 17.

3. *Помилки операторів.*Оператори, що експлуатують систему, також можуть припускатися помилок у своїй діяльності.

Усі ці фактори тісно пов'язані між собою. Збої в апаратних засобах можуть породити помилкові сигнали, які потім надходять на вхід програмних компонентів, що, своєю чергою, може призвести до непередбачуваної поведінки програмного забезпечення. Оператори зазвичай припускаються помилок у позаштатних ситуаціях, коли система поводиться незвичайним чином. Такі ситуації часто породжуються якими-небудь збоями у системі. Неправильні дії оператора, у свою чергу, можуть спровокувати збої та помилки в роботі апаратних засобів, що також може призвести до подальшого поширення збійних та хибних сигналів по системних ланцюгах. Таким чином, невелика помилка, що виникла в одній підсистемі і в принципі легко усувається, може призвести до ситуації, яка потребує повного відключення системи.

Безвідмовність системи також залежить від оточення, де вона експлуатується. Як зазначалося вище, важко передбачати системне оточення, в якому експлуатуватиметься система. Іншими словами, складно описати оточення у вигляді обмежень, які мають враховуватись при розробці системи. Підсистеми, що становлять цілісну систему, можуть по-різному реагувати на зміни в системному оточенні, тим самим впливаючи на загальну безвідмовність системи непередбачуваним чином. Внаслідок цього, навіть якщо система є єдиним цілим, буває важко чи неможливо виміряти рівень її безвідмовності.

Припустимо, система призначена для експлуатації за нормальної кімнатної температури. Для того щоб система могла функціонувати за інших температурних режимів, її електронні компоненти повинні бути розраховані для роботи в певному температурному інтервалі, скажімо, від 0 до 45°. При виході з цього температурного інтервалу компоненти можуть поводитися непередбачуваним чином. Тепер припустимо, що система є внутрішньою складовою повітряного кондиціонера. Якщо кондиціонер несправний і жене гаряче повітря через електронні компоненти, то вони, отже, і вся система можуть вийти з ладу. Якщо кондиціонер працює нормально, то система також має працювати нормально. Але внаслідок фізичної замкнутості кондиціонера можуть виникнути непередбачені впливи різних компонентів пристрою один на одного, що може призвести до різних збоїв.

Подібно до безвідмовності, інші інтеграційні характеристики (такі, як продуктивність і зручність експлуатації) також важкі для визначення, але можуть бути оцінені в процесі експлуатації системи. Оцінка інших властивостей, наприклад, безпеки системи та її захищеності, породжує великі складності. Ці характеристики непросто властиві працюючої системі, вони відбивають ті показники, що вона показує. Наприклад, при розробці заходів захищеності, де одним із показників є неможливість несанкціонованого доступу до даних, порівняно легко прорахувати всі можливі режими доступу до даних та виключити небажані. Тому оцінити рівень захищеності можна лише через характеристики системи, властиві їй за умовчанням. Більше того, система буде вважатися властивою захищеністю доти, доки хтось не зламає її засоби захисту.

### 2.2. Система та її оточення

Будь-яка система залежить від сигналів, даних або іншої інформації, що надходить на її входи; іншими словами, система функціонує у певному оточенні, яке впливає на її функціонування та продуктивність. Іноді оточення можна розглядати як самостійну систему, що складається з багатьох інших систем, які впливають одна на одну.

На рис. 2.1 показано кілька систем, об'єднаних у систему життєзабезпечення офісної будівлі. Система опалення, електроенергетична система, система освітлення, системи водопостачання та каналізації та система безпеки є підсистемами будівлі, яку також можна розглядати як систему. Будівля розташована на вулиці, яка є системою вищого рівня. Вулиця буде підсистемою системи міста тощо. Таким чином, оточення якоїсь системи саме є системою вищого рівня. У випадку оточення будь-якої системи – це композиція її локального оточення і оточення системи вищого рівня.



*Мал. 2.1. Ієрархія систем*

Розглянемо систему безпеки, що входить до системи життєзабезпечення будівлі (див. рис. 2.1). Її локальне оточення складається з інших систем цієї будівлі. До оточення системи також необхідно віднести системи, що не входять до системи будівлі, але відносяться до системи вулиці та системи міста, включаючи природні системи, у тому числі погодну (тобто вплив погодних факторів на систему безпеки).

Наведемо дві основні причини, якими викликана необхідність враховувати розробки систем їх оточення.

1. У багатьох випадках система призначена саме для реагування зміну певних параметрів оточення. Так, система опалення реагує на зміни у навколишньому середовищі, підвищуючи чи знижуючи температуру своїх опалювальних приладів. Тут правильне функціонування системи проявляється як реакція зміни параметрів оточення.

2. Часто якість функціонування системи може залежати від параметрів оточення непередбачуваним чином. Так, система електропостачання безпосередньо залежить від вуличного оточення будівлі. Наприклад, роботи, що проводяться з благоустрою вулиці, з недогляду можуть пошкодити силовий кабель і, отже, вивести з ладу всю систему електропостачання будівлі. Або грозовий розряд може індукувати великі струми в електричній системі, що може порушити нормальне її функціонування.

Крім фізичного оточення (довкілля), показаного на рис. 2.1, системи можуть перебувати у певних відносинах з організаційним оточенням, яке включає правила та процедури, засновані на політичних, економічних та екологічних пріоритетах суспільства. Якщо система побудована без урахування організаційного оточення, вона може знайти попиту ринку системних продуктів і буде відкинута користувачами і потенційними споживачами.

На розробку систем впливають як людські, і організаційні чинники, які входять у оточення системи.

1. *Експлуатаційний фактор.*Чи потрібна система внесення змін до робочого процесу її експлуатації, залежно від зміни параметрів оточення? Якщо відповідь це питання позитивний, отже, необхідно навчання персоналу, який експлуатує цю систему. Якщо навчання тривале або персонал може втратити в заробітку, існує можливість, що така система буде відкинута користувачами.

2. *Чинник персоналу.*Чи може використання системи призвести до зниження вимог до кваліфікації персоналу або докорінно змінити способи його роботи? Якщо це, тоді персонал може спробувати протистояти впровадженню системи у тому організацію. Менеджери середньої ланки, які керують проектами, часто підозрюють, що їхній статус в організації знизиться після впровадження комп'ютерних систем.

3. *Організаційний фактор.*Іноді використання системи може змінити структуру владних повноважень у створенні. Наприклад, якщо діяльність організації залежить від правильного функціонування складної системи, оператори цієї системи можуть мати значну вагу у владній структурі організації.

Ці людські, соціальні та організаційні фактори часто виявляються критичними при прийнятті рішення про те, чи буде впроваджуватися система. На жаль, передбачити ці фактори дуже складно, особливо якщо розробники системи не мають достатнього соціального та культурного досвіду. Щоб допомогти передбачити різні ефекти від впровадження систем до організації, розроблені спеціальні методології, такі як соціотехніка Мамфорда (Mumford) [243], методологія програмних систем Чекланда (Checkland) [69, 70]. Поглиблене соціологічне дослідження ефектів застосування обчислювальних систем наведено у роботі [3].

В ідеалі всі відомості про системне оточення слід включити до специфікації системи для того, щоб розробники могли їх врахувати при її проектуванні. Але насправді це неможливо. Зазвичай розробники систем роблять припущення про системне оточення або з урахуванням досвіду експлуатації інших подібних систем, або з здорового глузду. Якщо вони помиляться, система може в деяких ситуаціях функціонувати некоректно. Наприклад, якщо розробники не врахують можливі електромагнітні наведення на систему, вона може вийти з ладу, якщо поблизу неї розташовуються інші системи з великим електромагнітним випромінюванням.

### 2.3. Моделювання систем

У процесі формалізації вимог до системи та на етапі проектування система розглядається як сукупність компонентів та взаємозв'язків між ними. І тому використовуються моделі системної архітектури, які у графічному вигляді надають всю організацію системи, тобто. її компоненти та взаємозв'язки між ними.

Архітектура системи зазвичай представляється у вигляді блокової діаграми (блок-схеми), де блоки відповідають основним підсистем, а існуючі зв'язки між підсистемами позначаються лініями зі стрілками, що з'єднують окремі блоки діаграми. Зв'язки можуть відповідати потокам даних, послідовності включення підсистем у роботу або іншим типам залежності.

На рис. 2.2 представлено блок-схему основних компонентів системи сигналізації, що попереджає про несанкціоноване проникнення в житло. У табл. 2.1 наведено короткий опис підсистем, яким відповідають певні блоки на рис. 2.2.



*Мал. 2.2. Проста система сигналізації*

Таблиця 2.1. Функціональні підсистеми системи сигналізації

|  |  |
| --- | --- |
| **Підсистема** | Опис |
| Датчики рухуДверні датчикиКонтролерСиренаСинтезатор голосуТелефонний інформатор | Реагують на рух у кімнатах, які контролює системаВизначають, чи відкриті зовнішні двері будинкуКерує діями всієї системиВидає потужний звуковий сигнал під час незаконного проникнення в житлоСинтезує голосове повідомлення про проникнення до будинкуЗовнішній телефонний дзвінок для повідомлення служби безпеки (наприклад, поліції) про проникнення до будинку |

У цьому рівні деталізації система розбивається деякі підсистеми. Кожна підсистема, своєю чергою, може бути представлена ​​як декомпозиція своїх функціональних компонентів. Це такі компоненти підсистеми, які виходячи з призначення підсистеми виконують якусь одну функцію. На противагу цьому підсистема зазвичай виконує кілька функцій. Звичайно, декомпозицію підсистем (і самої системи) можна проводити за іншими ознаками, наприклад, конструктивним або технологічним.

Історично склалося так, що модель системної архітектури використовується для відокремлення апаратних та програмних компонентів системи, які зазвичай розробляються паралельно. Разом з тим протиставлення "апаратні засоби - програмне забезпечення" в сучасних системах найчастіше недоречне і несуттєве, оскільки практично всі системні компоненти мають певні обчислювальні можливості. Наприклад, машини, що пов'язують безліч комп'ютерів в єдину мережу, складаються з репітерів\*, мережевих шлюзів\*\* та кабелів. Репітери та шлюзи мають процесори та програми, що керують цими пристроями, і, звичайно ж, інші електронні компоненти.

*\* Репітер - пристрій, що передає сигнали, що надходять по одному кабелю, до іншого кабелю без маршрутизації або фільтрації пакетів. -*Прим. ред.

*\*\* Шлюз – пристрій для об'єднання мереж, що використовують різні протоколи передачі пакетів. -*Прим. ред.

На рівні системної архітектури раціональніше класифікувати підсистеми відповідно до виконуваних ними функцій, не акцентуючи спеціально увагу на тому, чи є вони апаратними або програмними компонентами. Питання про те, чи буде ця функція реалізована апаратно чи програмно, часто вирішується на основі нетехнічних факторів, таких як час, необхідний для створення компонента, або виходячи з наявності на ринку промислових виробів відповідних готових пристроїв.

Блок-схеми можна використовуватиме представлення систем будь-якого розміру. На рис. 2.3 показано архітектуру значно складнішої системи управління польотами. Ця система містить кілька основних підсистем, які є системами великого розміру. Напрямок інформаційних потоків між підсистемами показано лініями, що їх з'єднують, зі стрілками.



*Мал. 2.3. Архітектура системи управління польотами*

**2.3.1. Функціональні компоненти систем**

Як зазначалося в попередньому розділі, системна архітектура описується в термінах функціональних підсистем, незалежно від того, є ці підсистеми апаратними або програмними. Разом з тим функціональні компоненти в системі можна класифікувати за цілим рядом категорій, деякі з них наведені нижче.

1. *Сенсорні компоненти*збирають інформацію про системне оточення. Прикладами можуть бути радіолокатори в системі управління польотами, датчик положення паперу в лазерному принтері або термопара в камері топки котла.

2. *Виконавчі компоненти*роблять деякі дії в оточенні системи. Прикладами можуть бути регулюючий клапан, що закриває або відриває заслінку в трубопроводі для зменшення або збільшення швидкості потоку рідини в ньому, закрилки крил літака, які керують кутом нахилу літака, механізм подачі паперу в лазерному принтері.

3. *Обчислювальні компоненти –*на їхній вхід надходять певні дані, відповідно до яких вони виконують обчислення, потім на виході отримують нові дані. Приклад обчислювального компонента є математичний співпроцесор, що виконує обчислення з числами в експоненційному форматі.

4. *Комунікаційні компоненти*надають можливість іншим системним компонентам обмінюватись інформацією. Як приклад назвемо мережеві інтерфейсні плати комп'ютерів, об'єднаних у локальну мережу.

5. *Координуючі компоненти*узгодять роботу інших компонентів. Прикладом є планувальник завдань у системах реального часу. Планувальник визначає, який процес у цей час може оброблятися процесором.

6. *Інтерфейсні компоненти*перетворюють систему уявлень, якими оперує один системний компонент, систему уявлень, застосовуваних іншим компонентом. Прикладом " людського " інтерфейсного компонента може бути модель будь-якої системи та її у вигляді, зрозумілому іншій людині. Іншим прикладом є аналогово-цифровий перетворювач, що перетворює аналоговий сигнал на послідовність чисел.

У табл. 2.2 описано тип функціональних компонентів архітектури системи сигналізації, наведеної на рис. 2.2.

**Таблиця 2.2. Типи компонентів системи сигналізації**

|  |  |  |
| --- | --- | --- |
| **Тип компонента** | **Компонент** | **Функції компонента** |
| СенсорнийВиконавчийКомунікаційнийКоординуючийІнтерфейсний | Датчик руху, дверний датчикСиренаТелефонний інформаторКонтролерСинтезатор голосу | Реєструє рух у захищеному приміщенні, визначає, чи відчинені зовнішні дверіВидає звуковий сигнал під час незаконного проникнення в житлоЗдійснює телефонний дзвінок до центру управління при проникненні в будинок. Отримує команду у відповідь з центру управлінняКоординує всі системні компоненти. Діє за командами панелі керування та центру керуванняСинтезує повідомлення про проникнення до будинку |

Звичайно, неважко віднести системні компоненти до одного з перерахованих типів. Разом з тим, якщо в системі використовується програмне забезпечення, то зазвичай програмні елементи вбудовуються в більшість системних компонентів. Програмне забезпечення зазвичай використовується для керування всією системою.

Наведена класифікація компонентів допомагає під час проектування систем. Більшість систем містять компоненти всіх типів, і завдання розробника полягає у точному визначенні типу компонента, виходячи зі специфікації системи. Якщо кілька компонентів містять ознаки різних типів, це може призвести до того, що під час проектування системи можуть виникнути певні проблеми.

### 2.4. Процес створення систем

Етапи процесу створення системи показано на рис. 2.4. Ці етапи впливають на процес розробки програмного забезпечення відповідно до каскадної моделі, яка описується в розділі 3.



*Мал. 2.4. Процес створення системи*

Опишемо основні відмінності між процесом створення систем та процесом розробки програмного забезпечення.

1. *Залучення у процес розробки систем різноманітних інженерних дисциплін.*Процес створення систем зазвичай потребує залучення різноманітних інженерних дисциплін. Це може спричинити значні труднощі у розробці систем, оскільки кожна дисципліна використовує свою термінологію.

2. *Невеликий масштаб повторних робіт розробки систем.*Після прийняття рішень у процесі розробки систем (наприклад, про встановлення певних типів радіолокаторів у системі управління польотами) внесення змін до системи може виявитися досить дорогим. Перепроектування системи часто просто неможливе. Це одна з причин широкого використання ПЗ при створенні найрізноманітніших систем – програмні компоненти роблять системи більш гнучкими і дозволяють внести зміни до системи, що розробляється у відповідь на нові вимоги до неї.

До команди розробників систем неминуче включаються фахівці різних профілів. Команда розробників повинна мати широке коло знань, щоб всебічно розглянути всі системні можливості при прийнятті будь-яких рішень. Розглянемо систему управління польотом (СУП), у якій використовується радіолокаційна чи якась інша сенсорна система визначення місцезнаходження літаків (див. рис. 2.3). На рис. 2.5 схематично показані ті інженерні дисципліни, якими повинні мати члени команди розробників системи.



*Мал. 2.5. Інженерні дисципліни, що залучаються до процесу системотехніки*

Для багатьох систем існує практично безліч способів декомпозиції (розбиття) системи на підсистеми. При цьому фахівці різних профілів можуть пропонувати різні варіанти структурної моделі системи, які будуть містити різні функціональні компоненти. Тим самим можливий широкий діапазон альтернативних моделей. Вибір певної моделі не обов'язково ґрунтується лише на технічних аргументах. Нехай, наприклад, однією з альтернатив у розробці СУП є встановлення нової системи радіолокації замість модернізації існуючої. Якщо в команду розробників входять будівельники, то вони можуть наполягти саме на цьому варіанті створення СУП, оскільки він забезпечить роботою і їх будівельні підрозділи, які вони представляють. При цьому для обґрунтування потрібного варіанту можуть, звичайно, залучатись і технічні аргументи.

Оскільки ПЗ за своєю природою є гнучким і порівняно легко налаштованим, часто вирішення багатьох проблем, що несподівано виникли, перекладається на плечі фахівців з програмного забезпечення. Нехай, наприклад, під час створення СУП невдало обрано місце розташування однієї радарної установки – на екрані локатора іноді відбувається роздвоєння зображень. Для видалення цього ефекту необхідно пересунути радарну установку, що практично неможливо. Вирішенням цієї проблеми може бути створення спеціального програмного забезпечення, яке допоможе видалити роздвоєння зображень. Але в цьому випадку може знадобитися потужніша обчислювальна техніка, ніж та, що спочатку запланована, і це, у свою чергу, також може стати певною проблемою.

Перед фахівцями з програмного забезпечення часто ставляться завдання, які потрібно вирішувати без збільшення вартості апаратних засобів. Тому багато так звані програмні помилки не є наслідком будь-яких "вроджених" рис або властивостей програмного забезпечення. Вони можуть бути результатом спроби модернізувати програмне забезпечення відповідно до змін вимог, що висуваються до створюваної системи. Хорошим прикладом такої помилки може бути збій у системі керування багажем в аеропорту Денвера [329].

**2.4.1. Визначення системних вимог**

На етапі визначення системних вимог формуються та формалізуються вимоги до системи, що розглядається як єдине ціле. Як і під час аналізу вимог до програмного забезпечення, тут також необхідні консультації із замовниками системи та її кінцевими користувачами. На етапі визначення вимог зазвичай формуються вимоги трьох типів.

1. *Загальні функціональні вимоги*Основні функції, що виконуються системою, визначаються на найвищому (абстрактному) рівні представлення системи. Деталізація функціональних вимог відбувається на рівні підсистем. Наприклад, при розробці СУП обов'язково буде передбачено вимогу мати базу даних польотів, здійснених у контрольованому системою повітряному просторі. Однак структура цієї бази даних не буде визначена доти, доки не будуть відпрацьовані вимоги до інших підсистем.

2. *Системні характеристики.*Це інтегровані властивості системи, які обговорювалися вище. Вони можуть містити такі властивості, як продуктивність, безвідмовність, захищеність тощо. Ці дисфункції впливають на всі вимоги, що визначаються для підсистем.

3. *Властивості, які мають бути відсутніми у системи.*Деколи набагато важливіше вказати, що система не повинна робити, ніж те, що вона має виконувати. Наприклад, у СУП необхідно вимагати, щоб система не надавала операторам занадто багато інформації, тільки найнеобхіднішу, не відволікає їхню увагу.

Важливою частиною етапу визначення вимог є опис безлічі цілей, виконання яких повинна прагнути система. Вони не обов'язково повинні бути виражені в термінах функціональних властивостей системи, але повинні показати, як вона поводитиметься у своєму оточенні.

Щоб проілюструвати опис багатьох цілей, розглянемо об'єднану систему протипожежної безпеки та захисту від несанкціонованого вторгнення, призначену для встановлення в офісній будівлі. Цілі, що ґрунтуються на функціональних можливостях системи, можна сформулювати наступним чином.

*Система повинна забезпечити попередження про загоряння, що виникли всередині або поблизу будівлі, та несанкціоноване проникнення в цю будівлю.*

Ця мета точно визначає призначення системи, яка має попереджати про якісь небажані події. Таке формулювання підходить для системи безпеки, яка вже існує і яка має бути замінена. На противагу цьому можна сформулювати "ширшу" мету.

*Система має гарантувати відсутність серйозних порушень у нормальному функціонуванні та експлуатації будівлі внаслідок загорянь та незаконних вторгнень.*

Перше формулювання мети обмежує можливості проектування системи. Відповідно до неї від несанкціонованих проникнень можна застосувати складні захисні засоби навіть без внутрішньої системи сигналізації, а для захисту від вогню можна використовувати автоматичну систему пожежогасіння з розбризкувачами води. Але такі кошти можуть вивести з ладу електричну систему та завдати серйозних незручностей працюючим у будівлі.

Іноді основна труднощі у визначенні системних вимог у тому, що система будується у тому, щоб допомогти у вирішенні " злісної " проблеми (wicked problem) [294]. "Злісна" проблема - це проблема такої великої складності і має стільки взаємопов'язаних вхідних впливів, що її неможливо точно описати. Справжня природа такої проблеми може виявитись лише в процесі її вирішення. Як екстремальний приклад "злісної" проблеми можна навести завдання передбачення землетрусів. В даний час не існує точних способів передбачення ні епіцентру землетрусу, ні його часу, ні сили, ні на навколишнє середовище. Тому неможливо заздалегідь повністю спланувати всі дії на випадок великого землетрусу – це можна зробити лише тоді, коли він станеться.

**2.4.2. Проектування систем**

Проектування системи (рис. 2.6) полягає у визначенні системних компонентів на основі функціональних вимог до системи. Процес проектування складається з кількох етапів.



*Мал. 2.6. Етапи проектування системи*

1. *Розбиття вимог до груп.*Вимоги аналізуються та розбиваються на окремі групи. Зазвичай безліч вимог можна розбити на групи багатьма способами, цьому етапі зберігаються все " життєздатні " розбиття.

2. *Визначення підсистем.*Визначаються підсистеми, що індивідуально чи спільно реалізують системні вимоги. Група вимог зазвичай проектується кілька підсистем, тому можна об'єднати кілька вимог до однієї. Разом про те визначення систем впливають як системні вимоги, а й організаційні чи виробничі чинники.

3. *Розподіл вимог щодо підсистем.*У принципі, ця операція має бути виконана на попередньому етапі визначення підсистем. Але на практиці не завжди можна чітко узгодити розбиття вимог та визначення підсистем. Або, наприклад, обмежений асортимент підсистем, які купуються у зовнішніх постачальників (див. розділ 2.4.3), може призвести до перегляду вимог до системи.

4. *Специфікація функціональних показників підсистем.*Визначаються функціональні властивості кожної підсистеми. Якщо підсистема є програмною, цей етап буде частиною етапу створення специфікації даної підсистеми. На цьому етапі також формалізуються відносини між підсистемами.

5. *Визначення інтерфейсів підсистем.*Для кожної підсистеми визначається власний інтерфейс. Тільки після цього можна розпочати розробку самих підсистем.

На рис. 2.6 лінії, що з'єднують етапи, мають стрілки на обох кінцях. Це означає наявність зворотного зв'язку між етапами та можливість повертатися до попереднього етапу у процесі проектування системи. Дуже часто доводиться повертатися до попереднього етапу для вирішення проблем, що виникли на даному етапі.

Більшість систем можна розробити кілька проектів. Це передбачає широкий діапазон можливих рішень, що складаються з різних комбінацій апаратних та програмних компонентів та людського фактора. Для подальшої розробки вибирається рішення, яке найбільш повно задовольняє системним вимогам. Водночас на вибір проекту часто впливають організаційні та політичні чинники. Наприклад, якщо система розробляється на замовлення уряду, зазвичай вибираються національні постачальники комплектуючих, навіть якщо за певними параметрами вони (комплектуючі) поступаються закордонним; це, природно, впливає вибір проекту системи.

**2.4.3.**Розробка підсистем

На цьому етапі реалізуються підсистеми, які були визначені на етапі проектування системи. Для окремих підсистем етап розробки може вимагати включення різних процесів системотехніки. Так, якщо підсистема є програмною системою, етап розробки включатиме процеси формалізації вимог, проектування, створення тощо.

Іноді розробка всіх підсистем починається "з нуля". Але частіше буває так, що деякі підсистеми можна придбати на ринку промислових виробів і потім інтегрувати в систему, що створюється. Зазвичай, буває дешевше купити готовий виріб, ніж розробляти власну підсистему. На цьому етапі може виникнути необхідність повернутися до етапу проектування для того, щоб на рівні вимог "підігнати" придбаний виріб до системи. Цей виріб може не задовольняти всі вимоги до компоненту, який він замінює, але якщо асортимент комерційних продуктів досить широкий, витрати на повторне проектування невеликі.

Усі підсистеми, зазвичай, розробляються паралельно. Якщо виникають внутрішні проблеми, що переривають процес розробки підсистем, може знадобитися модифікація всієї системи. Коли система значною мірою складається з апаратних компонентів, проведення модифікації системи після початку її компонентів може виявитися досить затратним. Доводиться знаходити якесь "обхідне" рішення для виходу з таких ситуацій. Часто таким рішенням є включення до системи програмних компонентів, оскільки вони досить гнучкі та порівняно легко піддаються модифікаціям. У свою чергу, це веде до зміни вимог, які висуваються до програмного забезпечення, і, як наголошувалося в главі 1, до змін у проекті ПЗ.

**2.4.4.**Складання системи

Складання є інтеграцією незалежно розроблених підсистем в єдину закінчену систему. У процесі збирання може використовуватися метод "великого вибуху", коли всі підсистеми інтегруються одночасно. Але з технічних і організаційних причин набагато краще послідовне складання, коли окремі підсистеми інтегруються в систему по черзі, одна за одною.

Процес послідовного складання вважається найбільш підходящим для системної інтеграції з двох причин.

1. Зазвичай неможливо скласти такий графік робіт, у якому в усіх підсистем етап розробки закінчується одночасно.

2. Послідовне складання зменшує кількість помилок, пов'язаних з неправильною інтеграцією системи. Якщо одночасно інтегрується кілька підсистем, то причиною виявленої в процесі тестування помилки може бути кожна з них. Якщо інтегрується одна підсистема в систему, що вже працює, то причиною виявленої помилки, найімовірніше, буде остання інтегрована підсистема або знову встановлені зв'язки між нею і існуючими підсистемами.

Помилки та дефекти окремих підсистем та системи в цілому часто виявляються саме на етапі складання. Це може викликати полеміку та конфлікти між розробниками різних підсистем через те, чию підсистему визнати "винною" у цьому. Найнеприємніше в цій ситуації те, що на вирішення проблем може знадобитися кілька тижнів, а то й місяців роботи.

**2.4.5. Інсталяція системи**

При інсталяції система "занурюється" в оточення, в якому вона повинна працювати. У процесі інсталяції складних систем можуть виявитися різні проблеми, на вирішення яких часом йде кілька місяців, а то й років. Серед них можуть бути проблеми, наведені нижче.

1. Оточення, у якому встановлюється система, не збігається з тим, котрим вона спроектована. Це загальна проблема, яка часто виникає під час інсталяції ПЗ. Наприклад, програмна система може використовувати функції, які надає лише певна версія операційної системи. Однак версія, що визначаєпоточне оточення інстальованої програмної системи, може не володіти цими функціями. І тут після інсталяції система може працювати зовсім або її функції виявляться не реалізованими.

2. Потенційні користувачі можуть вороже ставитися до впровадження цієї системи в своїй організації. Це може зменшити відповідальність і кількість робіт, необхідних застосування системи у цій організації. Люди можуть свідомо відмовлятися від співпраці з фахівцями, які встановлюють систему. Наприклад, вони можуть відмовлятися від навчання роботі з цією системою або надавати інформацію, необхідну для інсталяції системи.

3. Нова система може співіснувати зі старою доти, поки в організації, де вона встановлюється, не переконаються, що нова система працює так, як потрібно. Це може призвести до певних проблем при інсталяції системи, особливо якщо нова та стара системи не є повністю незалежними, а мають деякі загальні компоненти. Трапляються ситуації, коли нову систему взагалі неможливо впровадити без деінсталяції старої системи. При цьому випробування нової системи можна провести лише тоді, коли стара система не працює.

4. При інсталяції можливі суто фізичні проблеми. Можуть виникнути складності при підгонці системи до будівлі, де вона встановлюється. Ця будівля може не мати достатньої кількості приміщень з каналами для мережних кабелів, можуть бути потрібні додаткові повітряні кондиціонери або інші прилади, що вбудовуються в будівлю тощо. А якщо це пам'ятка історії, яку охороняє закон, то при інсталяції системи зміни в будівлі взагалі неможливі.

**2.4.6.**Введення системи в експлуатацію

Після того, як система інстальована, її необхідно ввести в експлуатацію. Це передбачає проведення навчання системних операторів та зміну їхнього звичайного робочого процесу для того, щоб більш ефективно використовувати нову систему. На цьому етапі можуть виникнути непередбачені проблеми, якщо системна специфікація містить помилки або упущення. Поки система функціонує відповідно до специфікації, ці дефекти можуть не виявитися, і тому розробники можуть не передбачити відповідних режимів експлуатації системи.

Наприклад, проблемою, яка може виявитися тільки після введення системи в експлуатацію, є сумісність нової та існуючої системи. Це може бути суто фізична проблема сумісності. Можливі проблеми під час передачі даних від однієї системи до іншої. Більш "хитрою" проблемою може стати відмінність інтерфейсів різних систем. Тоді введення в експлуатацію нової системи може призвести до зростання помилок системних операторів через неправильне використання інтерфейсних команд нової системи.

**2.4.7.**Еволюція систем

Великі та складні системи мають дуже тривалий термін життя. Протягом свого життя вони вдосконалюються шляхом виправлення помилок у вихідних системних вимогах, а також для врахування нових вимог до них. Обчислювальні компоненти систем замінюються на нові, більш продуктивні компоненти. Організації, що експлуатують систему, можуть бути реорганізовані і, отже, використовувати систему в інший спосіб, ніж передбачалося спочатку. Може змінитися зовнішнє оточення системи, що також потребує внесення змін до неї.

Необхідність еволюції систем, як і програмного забезпечення (ця тема обговорюється у частині VII), викликана низкою причин.

1. Передбачувані зміни у технічній та діловій галузях, отримані на основі ретельного аналізу перспектив розвитку цих областей. Тут необхідно враховувати думку широкого кола фахівців, перш ніж приймати будь-які рішення.

2. Оскільки системи ніколи не є повністю незалежними одна від одної, зміни в одній підсистемі обов'язково вплинуть на продуктивність або поведінку інших підсистем. Отже, при внесенні змін до однієї підсистеми необхідні зміни практично у всіх підсистемах.

3. Причини, що призводять до прийняття певних рішень на початковому етапі проектування вихідної системи, рідко протоколюються або фіксуються взагалі. Це може спричинити перегляд деяких рішень, прийнятих на початковому етапі проектування, а отже, зміни у самій системі.

4. У міру збільшення "віку" системи її структура внаслідок зроблених раніше змін порушується, що, своєю чергою, призводить до зростання витрат на її модифікацію.

Внаслідок дедалі більшої залежності нашого суспільства від систем найрізноманітніших типів значно більше зусиль докладається для вдосконалення існуючих систем, ніж розробки нових. Такі раніше створені системи, які потрібно зберегти (шляхом їх модернізації), іноді називають успадкованими системами (legasy systems). Ці системи обговорюються у розділі 26.

**2.4.8. Виведення систем з експлуатації**

Виведення з експлуатації означає вилучення системи з оточення після закінчення терміну служби. Часто це не передбачає будь-яких складнощів. Але деякі системи можуть містити матеріали потенційно небезпечні для навколишнього середовища. Системотехніки повинні передбачити процедуру виведення такої системи з експлуатації на етапі її проектування. Наприклад, токсичні хімічні сполуки, що використовуються в системі, повинні бути укладені в герметичні контейнери, кожен з яких можна видалити з системи як єдиний елемент для подальшої утилізації.

При деінсталяції програмного забезпечення зазвичай не виникає фізичних проблем. Разом з тим, деякі програмні компоненти можуть бути інкорпорованими в системи, необхідні для деінсталяції ПЗ; наприклад, якщо програмне забезпечення використовувалося для моніторингу стану інших системних компонентів.

Після виведення системи з експлуатації деякі її компоненти можуть бути використані в інших системах. Якщо дані з системи, що деінсталюється, повинні повернутися в організацію, для цього можуть бути застосовані будь-які інші системи. Ці дані часто мають значну вартість. Тема повторного використання даних розглядається у розділі 28.

### 2.5. Придбання систем

Замовниками складних обчислювальних систем зазвичай є великі організації, наприклад військове відомство, уряд чи аварійні служби. Такі системи можна купити як єдине ціле, можна купити окремі частини, які потім інтегруються в створювану систему, можна спроектувати систему та розробити на окреме замовлення "з нуля". Для великих систем процес вибору однієї з цих варіантів може розтягнутися кілька місяців і навіть років. Процес придбання системи – визначення найбільш оптимального в організацію шляху її придбання і вибір найкращого постачальника системи.

Процес придбання системи повністю підпорядковується процесу системотехніки. До початку процесу придбання необхідно розробити системну специфікацію і архітектуру системи, що з двома основними причинами.

1. Для придбання або укладання контракту на розробку та побудову системи потрібна повністю закінчена системна специфікація.

2. Майже завжди дешевше придбати систему, ніж розробити її (як окремий проект). Архітектура системи потрібна для того, щоб визначити, які її підсистеми можна купити, а які необхідно розробляти.

Великі складні системи зазвичай складаються з придбаних компонентів та компонентів, спеціально створених для даної системи. Це одна з передумов, що вимагає включення програмних компонентів до складу систем – програмне забезпечення має "склеїти" в єдине ціле (причому ефективно працююче) окремі апаратні компоненти, що існують. У необхідності розробки програмного "клею" криється причина того, що економія від застосування придбаних компонентів не така велика, як очікується. Докладно тема придбаних систем обговорюється у розділі 14.

На рис. 2.7 показані етапи процесу придбання як готових систем, і розроблюваних на замовлення. Перелічимо деякі важливі моменти процесу придбання.

1. Придбані компоненти, як правило, не задовольняють точно всім системним вимогам, внаслідок чого необхідне припасування вимог відповідно до цих компонентів. Більше того, зазвичай стоїть нелегка дилема вибору між системними вимогами та властивостями набутої системи. Найчастіше "на жертву" приносяться системні вимоги. Це, своєю чергою, впливає інші підсистеми.

2. Якщо система розробляється на замовлення, специфікація вимог є основою контракту на систему. Таким чином, специфікація має таку ж правову силу, як інша технічна документація.

3. Після вибору розробника системи у контракті з ним необхідно обумовити можливості внесення змін до вимог, хоча це може призвести до зміни вартості системи.



*Мал. 2.7. Процес придбання системи*

Більшість апаратних підсистем та багато програмних підсистем (такі, як системи управління базами даних) не розробляються спеціально для включення до складу великих систем. Часто у них вбудовуються вже готові системи.

Дуже небагато організацій мають змогу проектування, виробництва та тестування всіх компонентів складних великих систем. Організація – розробник системи, яку зазвичай називають провідним чи генеральним підрядником, може укладати контракти розробку окремих підсистем коїться з іншими субпідрядниками (рис. 2.8). Для створення великих систем, таких як системи управління польотами, група розробників може створити консорціум для виконання контракту. До консорціуму повинні входити розробники та постачальники всіх компонентів системи, наприклад розробники обчислювальних пристроїв та програмного забезпечення, постачальники периферійного обладнання та спеціального обладнання (скажімо, радарів).

Модель "підрядник-субпідрядник" мінімізує кількість організацій, що беруть участь у реалізації контракту. Субпідрядники розробляють та виробляють частини системи відповідно до специфікації, що надається провідним підрядником. Після завершення робіт субпідрядниками система збирається з окремих частин провідним підрядником. Готова система поставляється замовнику (покупцеві). Залежно від умов контракту, замовник може надати провідному підряднику вільний вибір субпідрядників або зажадати, щоб субпідрядники вибиралися із заздалегідь обумовленого списку.



*Мал. 2.8. Модель "підрядник-субпідрядник"*

**КЛЮЧОВІ ПОНЯТТЯ**

• Системотехніка – це комплексна технологія створення систем, що потребує залучення багатьох інженерних дисциплін.

• Інтегровані властивості системи – це властивості, які притаманні системі як єдиному цілому, а чи не її окремим компонентам. До інтегрованих системних властивостей належать безвідмовність, продуктивність, зручність експлуатації, безпека та захищеність системи.

• Архітектура системи, зазвичай представлена ​​як блок-схеми, показує основні підсистеми та його взаємозв'язок.

• Функціональні компоненти систем поділяються на такі типи: сенсорні, виконавчі, обчислювальні, координуючі, комунікаційні та інтерфейсні.

• Процес створення систем включає такі етапи: складання специфікації, проектування, розробку, інтеграцію (складання) та тестування. Найбільш відповідальним етапом є складання системи, коли різні підсистеми, іноді від різних виробників, інтегруються у єдину систему.

• Процес придбання системи включає етапи специфікації системи, формування заявки на придбання, вибір постачальника, а потім укладання контракту на купівлю або розробку системи. Часто деякі частини обчислювальних систем набуваються у сторонніх виробників.

**Вправи**

**2.1.** Поясніть, чому системне оточення може мати непередбачуваний вплив на функціонування системи.

**2.2.** Змініть схему на рис. 2.6 таким чином, щоб включити до неї етап придбання підсистем після етапу їх ідентифікації. Покажіть на новій схемі зворотний зв'язок від включеного етапу придбання інших етапів процесу проектування системи.

**2.3.** Поясніть, чому процес специфікації систем, що використовуються аварійними службами для управління надзвичайними ситуаціями, є "злісною" проблемою.

**2.4.** Поясніть важливість отримання повного опису системної архітектури на ранній стадії процесу розробки системної специфікації.

**2.5.** На рис. 2.1 показано ієрархію систем окремої будівлі. Система безпеки, що включає систему захисту від несанкціонованого проникнення в будівлю та протипожежну систему є розширенням системи, представленої на рис. 2.2. Вона містить датчики диму, датчики руху та дверні датчики, відеокамери, керовані комп'ютером, розташовані в різних місцях будівлі, пульт управління, де збирається вся інформація від системи безпеки, засоби зовнішніх комунікацій для зв'язку з відповіднимислужбами, такими як поліція та пожежна охорона. Намалюйте блок-схему архітектури такої системи.

**2.6.** Розробляється система запобігання повеням для міста, якому загрожують часті повені. Система включає безліч датчиків рівня води в річці, зв'язок з метеослужбою, що надає прогноз погоди, зв'язок із службами безпеки (поліцією, береговою охороною тощо), відеомонітори, встановлені в різних місцях, та кімнату керування, обладнану пультом керування та моніторами.

**2.7.** Чергові оператори мають доступ до бази даних та можуть перемикати відеомонітори. База даних містить інформацію з датчиків, розташованих в інших містах, також схильних до ризику повені, про ситуацію в цих містах (рівень води, сила і напрям вітру і т.п.), таблицю висот прибережних міст, розташування обладнання, що контролює рівень води, контактні телефони служб безпеки, частоти місцевих радіостанцій та ін.

**2.8.** Намалюйте блок-схему можливої ​​архітектури такої системи. Також визначте основні підсистеми та взаємозв'язки між ними.

**2.9.** Назвіть три проблеми, які можуть виникнути під час інсталяції системи у сторонній організації.

**2.10.** Розгляньте системотехніку як професію та порівняйте її з професією інженера-електронника та спеціаліста з програмного забезпечення.

**2.11.** Допустимо, ви інженер, включений до групи розробників фінансової системи. У процесі інсталяції системи ви виявляєте, що її використання в організації може призвести до звільнення великої кількості людей. Персонал організації не надає вам інформації, необхідної для завершення інсталяції системи. Що ви робитимете в цій ситуації як інженер-системотехнік? Чи будете ви з почуттям професійної відповідальності прагнути до завершення її впровадження системи, що вимагається від вас контрактом? Чи просто припиніть роботу доти, доки організація не розбереться зі своїми проблемами?