

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка



Кафедра програмних систем і технологій

**Методичні вказівки до виконання курсової роботи
здобувача вищої освіти другого (магістерського) рівня вищої
освіти за спеціальністю 121 “Інженерія програмного
забезпечення”**

Київ-2023

УДК 004.412
ББК В 192.145
Е 605

Укладачі: *О. С. Бичков, доктор технічних наук, професор*
І.А.Юрчук, кандидат фіз-мат. наук, доцент
К.В.Меркулова, кандидат технічних наук, доцент

Рецензент: Шатирко А.В., доктор фіз.мат наук, доцент

Подано основні вимоги щодо тематики, організації, змісту, оформлення та захисту курсової роботи здобувачами вищої освіти другого (магістерського) рівня вищої освіти за спеціальністю 121 “Інженерія програмного забезпечення”

ЗМІСТ

Вступ	4
1. Тематика курсових робіт	5
2. Завдання на курсову роботу	6
3. Порядок роботи над курсовою роботою	9
4. Загальні вимоги до обсягу курсової роботи	11
5. Хід виконання курсової роботи	12
6. Правила оформлення курсової роботи	13
7. Підготовка курсової роботи до захисту і захист	23
8. Список використаних джерел	27
Додаток А – Приклад оформлення титульного листа	28
Додаток Б – Приклад оформлення змісту до пояснювальної записки	29
Додаток В – Приклад оформлення “Списку використаних джерел”	30
Додаток Г – Алгоритм програми (приклад)	31
Додаток Д – Фрагмент блок-схеми алгоритму програми (приклад)	32
Додаток Е – Приклад документування архітектури розробленого програмного забезпечення у відповідності до стандарту Software Architecture Document	33

ВСТУП

Курсова робота виконується студентами денної форми навчання відповідно до освітньо-професійної програми спеціальності 121 “Інженерія програмного забезпечення” на основі знань і практичних навичок.

Метою курсової роботи є вироблення у студентів самостійних навичок наукової роботи, розробки математичної моделі, складання та програмної реалізації алгоритму. Студент в процесі роботи самостійно, на підставі вивченого теоретичного матеріалу і творчого опрацювання літератури по темі курсової роботи, розробляє програму, використовуючи основні можливості відомих йому мов.

В процесі виконання курсової роботи студент набуває навичок науково-дослідної і самостійної творчої роботи. При виконанні завдання курсової роботи студенти отримують досвід роботи з довідковою літературою, стандартами, а також навички складання і оформлення програмної документації.

Робота повинна містити обґрунтування актуальності теми, зв'язок роботи з науковим напрямком науково-дослідної роботи кафедри, огляд літератури за темою і вибір напрямку досліджень, виклад методики досліджень, проведені теоретичні і (або) експериментальні дослідження, проведені студентом, аналіз і узагальнення результатів досліджень, виявлення наукової новизни одержаних результатів, ступеню новизни (вперше одержано, удосконалено, дістало подальший розвиток), висновки, список використаних джерел.

Студентові слід мати на увазі, що наукове дослідження – це цілеспрямоване вироблення нових знань, які розкривають нові явища в суспільстві та природі, для використання їх у практичній соціальній діяльності. З'ясовуючи завдання дослідження, необхідно зважати на те, що між ним й темою курсової роботи є системні логічні зв'язки.

1. ТЕМАТИКА КУРСОВИХ РОБІТ

Теми курсових робіт розробляються і затверджуються кафедрою програмних систем і технологій. Тематика курсової роботи може ґрунтуватися на матеріалах виробничої практики студентів, на базі наукових праць викладачів і студентських учбово-дослідницьких робіт, а також інших розробок.

Тематика курсових робіт повинна відповідати таким вимогам:

бути актуальною;

мати практичне значення;

відповідати сучасному стану розвитку наукових досліджень;

бути націленою на вирішення проблематики, тісно пов'язаної з використанням сучасних інформаційних технологій і систем;

мотивувати студентів на самостійне вирішення актуальних наукових задач;

передбачати вибір ефективних математичних методів та засобів моделювання складних об'єктів та процесів.

Теми робіт розробляються з урахуванням вимог галузевих стандартів вищої освіти для освітньо-кваліфікаційного рівня "магістр", наявного досвіду наукових керівників щодо проведення наукових досліджень. При виборі теми роботи потрібно ураховувати можливості студента щодо спроможності виконання науково-дослідної роботи, його прихильності до проведення наукового дослідження в рамках дисциплін навчальної програми за спеціальністю 121 "Інженерія програмного забезпечення" рівня "магістр"

2. ЗАВДАННЯ НА КУРСОВУ РОБОТУ

Курсова робота – це вид кваліфікаційної роботи, що призначена для формування знань, умінь та навичок з самостійного проведення наукового дослідження під керівництвом наукового керівника. Курсова робота передбачає: проведення аналізу та узагальнення теоретичних положень дослідження щодо процесів, явищ, об'єктів управління, які досліджуються; розроблення чи модифікація методу, алгоритму для обґрунтування запропонованого підходу, на підґрунті якого проводиться дослідження; проведення експерименту та аналіз отриманих результатів на предмет їх вірогідності та достовірності.

Мета написання курсової роботи – узагальнити та систематизувати знання та практичні навички студентів з проведення наукового дослідження, обґрунтування його актуальності, вибору певного підходу до розв'язання проблеми, проведення дослідження та оформлення отриманих результатів з урахуванням знань та навичок, набутих ними з математичної та природничо-наукової підготовки й професійної та практичної підготовки.

У процесі виконання курсової роботи студенти закріплюють вміння та навички користування науковою, нормативною та довідковою літературою, вчать аналізувати існуючі підходи, методики, методи розв'язання основних задач у вибраній науковій проблемі, оформлювати отримані результати у вигляді закінченої роботи за заданою структурою та змістом, які відповідають вимогам до написання курсової роботи.

Основні завдання написання курсової роботи:

ознайомитися з вимогами щодо організації і проведення наукового дослідження, підготовки та оформлення курсової роботи як виду наукового дослідження;

опрацювати методи роботи з основними джерелами інформації та науковою літературою;

обґрунтовувати актуальність, наукове і практичне значення теми, обраної для дослідження; набути знань та умінь щодо використання методології проведення наукових досліджень та оформлення теоретичних та експериментальних результатів у вигляді курсової роботи, яка є першим етапом при підготовці магістерської роботи;

набути умінь щодо самостійного аналізу методів проведення наукових досліджень з використанням: сучасних інформаційних систем та технологій, спеціалізованих програмних пакетів, середовищ розроблення додатків, програмних засобів розподіленої та паралельної обробки даних, мобільних технологій;

набути знань та умінь щодо обґрунтування та пояснення на потрібному рівні основних результатів, які доводять актуальність та своєчасність проведених досліджень; набути досвіду щодо оприлюднення отриманих результатів у наукових виданнях та їх апробацію на конференціях;

ознайомитися з вимогами до підготовки та процедури захисту курсової роботи.

Завдання видається викладачем після викладення ним теоретичного матеріалу та отримання студентами практичних навичок по дисциплінам курсів навчального плану здобувача вищої освіти другого (магістерського) рівня вищої освіти за спеціальністю 121 «Інженерія програмного забезпечення». Відповідно до теми курсової роботи студент одержує вихідні вимоги до роботи у вигляді завдання, що містить назву теми, вихідні дані, обсяг роботи, термін представлення роботи і т.д.

При необхідності в завданні можуть вноситися зміни і доповнення, що повинні бути погоджені зі студентом - автором роботи і завідувачем кафедрою.

Студентам, що не з'явилися за одержанням завдання у встановлений термін, завдання видається керівником тільки з дозволу завідувача кафедрою або декана. Курсова робота повинна бути виконана і представлена до захисту не пізніше, ніж за два тижні до початку екзаменаційної сесії у термін визначений завідувачем кафедри.

Керівник курсової роботи видає завдання, допомагає студентові скласти календарного графіка його виконання і проводить систематичні консультації. Керівник рекомендує студентам основну літературу і довідкові матеріали по темі, призначає в календарному графіку термін виконання окремих розділів курсової роботи і періодичних звітів про хід роботи.

Контроль керівника ні в якій мірі не звільняє студента від відповідальності за правильність виконання роботи і прийнятих рішень. Основною задачею керівника курсової роботи, крім технічного керівництва, є прищеплення навичок самостійної і систематичної роботи студента з дотриманням затверджених програм дисциплін навчального плану. Обов'язок консультанта - застерегти студента від грубих помилок.

У виборі тих або інших рішень ініціатива надається студентові. При цьому керівник може рекомендувати відповідну літературу, журнальні статті і т.п., переслідуючи основну мету - поглиблене самостійне вивчення студентом даного питання.

Студент цілком відповідає за прийняті науково-технічні рішення, правильність виконаних алгоритмів, розрахунків, використаних методів і засобів, якість виконання й оформлення курсової роботи, а також за своєчасне її завершення.

Ціль виконання курсової роботи - придбання навичок самостійної розробки системних і інструментальних програм у сучасних операційних середовищах. На захист виносять методи й алгоритми, використані студентом у процесі виконання курсової роботи до викладеного в пояснювальній записці, а також практичну реалізацію алгоритмів у виді програмного продукту, розробленого студентом самостійно.

3. ПОРЯДОК РОБОТИ НАД КУРСОВОЮ РОБОТОЮ

Курсова робота починається з одержання від керівника індивідуального завдання. У тижневий термін студент повинний розробити і затвердити в керівника календарний графік виконання роботи. У процесі проектування він консультиється з керівником у міру потреби та у зв'язку з виникаючими питаннями.

Рекомендується виконання курсової роботи розбити на наступні етапи:

Підготовчий. Студент повинний зрозуміти поставлену перед ним задачу, ознайомитися з рекомендованою літературою. При цьому варто критично підходити до вивчення джерел: рекомендується відбирати найбільш свіжі, останні дані і використовувати самі авторитетні джерела. Варто ясно представити мету завдання, що вирішується й уважно проаналізувати вимоги, пред'явлені до її рішення.

Проектування. На цьому етапі студент повинний розглянути та описати різні шляхи рішення поставленої задачі, а також самому розробити алгоритмічне і математичне забезпечення для наступної розробки програми.

Реалізація. На початку цього етапу студент повинний вибрати найбільш раціональне рішення і скласти графік подальшої роботи, у ході якої необхідно виконати остаточну розробку і налагодження програмного забезпечення. Варто звернути увагу на повноту, правильність і акуратність ведення документації в ході виконання курсової роботи, на повноту перевірки правильності роботи програми.

Оформлення. Студент зобов'язаний оформити пояснювальну записку і графічний матеріал відповідно до вимог до оформлення технічної документації, відповідно до діючих стандартів. Метою є забезпечення відповідності пояснювальної записки нормам і підготовка студента до захисту курсової роботи.

Заключний. На цьому етапі проводиться захист курсових робіт. Студент зобов'язаний представити керівникові остаточно оформлену пояснювальну

записку до курсової роботи не пізніше, ніж за 7 днів до захисту. Керівник перевіряє роботу і дає вказівки про виправлення або доповнення, що студентові варто розглянути і внести в роботу, після чого підписує пояснювальну записку і креслення. На цьому курсова робота вважається закінченою і може бути представлена до захисту.

4. ЗАГАЛЬНІ ВИМОГИ ДО ОБСЯГУ КУРСОВОЇ РОБОТИ

У пояснювальній записці викладений основний зміст роботи. У ній подається:

- обґрунтування актуальності теми;
- розкривається творчий задум роботи;
- утримуються розрахунки і пояснення, таблиці і графіки.

Загальними вимогами до розрахунково-пояснювальної записки є:

чіткість викладу, логічна послідовність і повна відповідність завданню на курсову роботу;

переконливість аргументації;

конкретність викладу результатів роботи.

Структура пояснювальної записки і приблизний обсяг сторінок окремих розділів:

Титульний лист	1
Завдання на курсову роботу	1
Анотація (реферат)	1
Зміст	1
Вступ	2
Розділи і підрозділи основної частини	30-35
Висновок	1
Список використаних джерел	1
Додатки	20

Загальний обсяг пояснювальної записки повинний складати 40-45 сторінок тексту з урахуванням рисунків (без додатків). Перевищення максимального обсягу неприпустимо і розцінюється, як невміння студента лаконічно викладати результати курсової роботи.

5. ХІД ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Виконання курсової роботи передбачає створення програмного модуля і написання інструкції для користувача.

Програма повинна мати чіткий інтерфейс користувача. Програма може складатися з одного модуля і при необхідності однієї або декількох бібліотек динамічного компонування.

У пояснювальну записку до курсової роботи повинні входити:

- аналіз предметної області у відповідності до постановки задачі;
- вхідна і вихідна інформація;
- математичний опис задачі;
- запропонований алгоритм рішення задачі;
- опис методів створення програмного продукту;
- опис класів і об'єктів програмного середовища, що плануються до використання;
- опис реалізації інтерфейсу користувача;
- опис реалізації програмного модуля;
- інструкція для користувача програми.

У додатку до пояснювальної записки приводиться лістинг програми, блок-схема алгоритму програми. Курсова робота виконується самостійно з рівномірним розподілом роботи протягом двох семестрів.

6. ПРАВИЛА ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ

6.1. Вимоги до оформлення текстових документів

Пояснювальна записка є основним документом, пропонованим студентом при захисті курсової роботи. Вона складається відповідно до вимог діючих стандартів. У ній відбивають етапи роботи і результати, отримані при виконанні курсової роботи.

Пояснювальна записка до курсової роботи друкується на принтері через 1,5 інтервал на одній стороні листа папера формату А4. При цьому необхідно залишати поля: ліворуч -25 мм., праворуч - 10 мм., зверху - 20 мм., знизу - 15 мм.

Текст пояснювальної записки без помарок і виправлень. Помилки, описки і графічні неточності необхідно акуратно підчистити і написати виправлений текст від руки чорнилом, пастою або тушшю чорного кольору. Ушкодження аркушів, помарки не допускаються.

Виклад змісту роботи в пояснювальній записці повинне бути коротким, чітким, що виключає можливість суб'єктивного тлумачення, і вестися від першої особи множини, наприклад : "приймаємо", "вибираємо" і т.п. Такі вирази, як "я беру", "вважаю", "мені задане" і т.п. уживати не слід. Мова викладу повинна бути технічно грамотною, не містити маловживаних слів.

Варто користуватися єдиним і відповідним установленим стандартам термінологією і визначеннями, а при їхній відсутності - загальноприйнятими в науково-технічній літературі.

Скорочення слів у тексті і підписах під ілюстраціями, як правило, не допускається.

Літерним позначенням різних величин «значення символів і числових коефіцієнтів» при першому їхньому використанні варто давати розшифровку безпосередньо під формулою.

У записі обчислень за формулами наводиться тільки вихідна формула, вираження з підставленими цифрами й остаточний результат. Проміжні викладення виключаються.

Весь текст пояснювальної записки поділяють на розділи. Кожен розділ варто починати з нової сторінки. Розділи в межах усієї пояснювальної записки, а також підрозділи і пункти мають порядкові номери, позначені арабськими цифрами з крапкою наприкінці, наприклад: 1. - перший розділ; 2. - другий розділ; 2.1. - перший підрозділ другого розділу; 2.1.1. - перший пункт першого підрозділу другого розділу. Вступ і висновок не нумеруються.

Заголовки розділів пишуть прописними буквами по середині тексту. Заголовки підрозділів пишуть з абзацу, відступаючи ліворуч 15 мм, малими літерами (крім першої прописної). У заголовку не допускаються переноси слів. Пробіли над заголовками і під ними - 2 см. Крапку наприкінці заголовка не ставлять. Якщо заголовок складається з двох позицій, то вони розділяються крапкою. Заголовок підкреслювати не можна.

Нумерація сторінок пояснювальної записки повинна бути наскрізною. Номер сторінки проставляється арабськими цифрами в правому нижньому куті сторінки. На титульному листі число 1 не ставиться.

Текст пояснювальної записки ілюструється схемами, графіками, таблицями, номограмами. Кількість ілюстрацій визначається змістом курсової роботи і повинне бути достатнім для того, щоб додати пояснювальній записці ясність і конкретність, але як правило, не повинно перевищувати 7-10 ілюстрацій.

Всі ілюстрації іменуються рисунками, що нумеруються послідовно в межах розділу арабськими цифрами. Номер рисунку повинен складатися з номера розділу і порядкового номера рисунку, розділених крапкою (наприклад, "рис. 2.5"- п'ятий рисунок другого розділу). Рисунки повинні бути чіткими і виконані тільки чорним кольором. Кожен рисунок має супроводжуватися змістовним підписом. Підпису під рисунком і таблицями повинні бути короткими і пояснювати основний зміст. Підпис подається під рисунком в один

рядок з номером (наприклад. "Рис. 2.5 – Загальний вигляд блоку"). Рекомендується весь матеріал, що пояснює рисунок, приводити в рисунковому тексті пояснювальної записки, а не користуватися вільним місцем на рисунку.

На всі ілюстрації повинні бути посилання в тексті. Посилання на раніше згадані ілюстрації і таблиці дають зі скороченим словом "Дивися" (наприклад: "див. рис. 2.5", "-див. табл. 1").

Таблиці служать для оформлення цифрового матеріалу, приводяться після першого згадування про них у тексті. На всі таблиці повинні бути посилання в тексті, при цьому слово "Таблиця" у тексті пишуть цілком, якщо таблиця не має номера, і скорочено - якщо має номер, наприклад: "...у табл. 1.2". У повторних посиланнях на таблиці й ілюстрації варто вказувати скорочено слово "дивися", наприклад, "див. табл. 1.2". Таблиці повинні нумеруватися в межах розділу арабськими цифрами. Над правим верхнім кутом таблиці поміщають напис "Таблиця" із указівкою порядкового номера таблиці. Номер таблиці складається з номера розділу і порядкового номера таблиці, розділених крапкою, наприклад: "Таблиця 2.4"- четверта таблиця другого розділу. Якщо в пояснювальній записці тільки одна таблиця, то її не нумерують.

Кожна таблиця повинна мати заголовок. Заголовок і слово починають із прописної букви. Заголовок не підкреслюють «Таблиця 2.4 - Результати розрахунку значення змінної». Заголовки стовбців таблиць повинні починатися з прописних букв, підзаголовки - з рядкових, якщо вони складають одну позицію з заголовком, і з прописних, якщо вони самостійні. Робити заголовки таблиці по діагоналі не дозволяється. Висота рядків повинна бути не менш 8 мм.

Таблицю розмішають таким чином, щоб її можна було читати без повороту записки або з поворотом за годинниковою стрілкою. При великій кількості рядків допускається перенос таблиці на інший листок. При цьому заголовок приводиться тільки в її першій частині.

Якщо повторюваний у графі таблиці текст складається з одного слова, його допускається замінити лапками; якщо з двох або більш слів, то при першому

повторенні його заміняють словами "Те ж", а далі - лапки. Ставити лапки замість повторюваних цифр, знаків, марок, математичних символів не можна. Якщо цифрові або інші дані в якому-небудь рядку таблиці не приводять, то в ній ставлять прочерк.

Формули в записці (якщо їх більш однієї) нумерують арабськими цифрами в межах розділу. Номер формули складається з номера розділу і порядкового номера формули в розділі, розділених крапкою. Номер ставиться з правої сторони листа на рівні нижнього рядка формули в круглих дужках, наприклад: (3.1) - перша формула третього розділу. Посилання на формулу вказують порядковим номером формули в круглих дужках, наприклад "... у формулі (2.1)". Опис символів і числових коефіцієнтів варто приводити безпосередньо під формулою зі слова "де" без двокрапки після нього в тій же послідовності, у якій вони дані у формулі з нового рядка.

Рівняння і формули варто виділяти з тексту вільними рядками. Якщо рівняння не уміщається в один рядок, він повинно бути перенесене після знаків " - " або " + ", " - ". " * ". " : ". Кожна формула, крім найбільш простих, повинна мати посилання на джерело, звідки вона взята, якщо вона не виводиться автором.

У список літератури включають усі використані джерела в порядку появи посилань на них у тексті пояснювальної записки. При посиланні в тексті на використовувану літературу вказують порядковий номер, виділений двома квадратними дужками за списком джерел, наприклад [4]. Джерело описується за такою формою: прізвище й ініціали автора, повна назва книги або статті, місце і рік видання, обсяг (для журналу - назва журналу, рік видання, номер сторінки).

Заборонено наводити у списку літератури джерела країни-агресора.

Приклад наведений в Додатку В.

Додатки оформляють як продовження пояснювальної записки на наступних її сторінках або у виді окремої частини, розташовуючи їх у порядку появи посилань у тексті. Кожен додаток варто починати з нового листа з

вказівкою в правому верхньому куті слова "Додаток А", "Додаток Б" і т.д. Номери пишуться буквами, далі поміщається тематичний заголовок.

Текст кожного додатка при необхідності може бути розділений на підрозділи і пункти, нумеровані арабськими цифрами в межах кожного додатка. Перед ними ставитися буква "Д". наприклад Д.1.2.3, що означає третій пункт другого підрозділу першого додатку.

Рисунки, таблиці, формули в межах кожного додатку нумерують арабськими цифрами, наприклад: "Рис. Д. 1.1" (перший рисунок першого додатка); "Табл. Д. 1.1" (перша таблиця першого додатка).

Пояснювальна записка підписується студентом і керівником.

6.2. Вимоги до оформлення графічної документації

При оформленні структурних і функціональних схем програм, блок-схем або структурних схем алгоритмів, схем взаємозв'язку програмних модулів, схем передачі керування, а також інших графічних схем або плакатів, необхідно керуватися діючими стандартами.

Схеми виконуються за допомогою графічних прикладних пакетів.

Плакати і схеми повинні бути супроводжуватися необхідними поясненнями, що дають можливість читати їх без звертання до тексту пояснювальної записки.

6.3. Зміст графічної частини

Графічна частина виконується як додаток до пояснювальної записки. В ній повинні бути представлені схеми алгоритмів, оформлені згідно ГОСТ ЄСПД 19.701-90, і, за необхідності (якщо програма складається з великої кількості файлів і функцій), структурна схема програми або програмної системи.

Блок-схеми виконуються комп'ютерним способом з дотриманням стандартів (див. Додаток Д).

ГОСТ 19.701-90 розповсюджується на умовні графічні позначення (символи) в схемах алгоритмів і програм, що відображають основні операції процесу обробки даних і програмування для систем програмного забезпечення

обчислювальних машин, комплексів і систем незалежно від їх призначення і області застосування.

Стандарт не розповсюджується на записи і позначення, що поміщаються усередині символу або поряд з ним і служать для уточнення виконаних ним функцій.

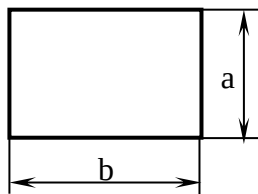
Стандарт встановлює перелік, найменування, форму, розміри символів і функцій, що відображаються символами.

6.3.1. Умовні графічні позначення в блок-схемах алгоритмів і програм

Перелік, найменування, позначення і розміри деяких обов'язкових символів і функцій, що відображаються ними, в алгоритмі і програмі обробки даних, наведено нижче.

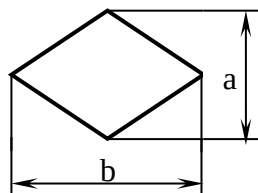
6.3.1.1. Процес

Виконання операції або групи операцій, в результаті яких змінюється значення, форма наведення або розташування даних.



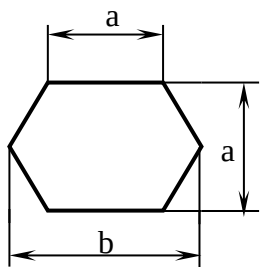
6.3.1.2. Рішення

Вибір напрямку виконання алгоритму або програми залежно від деяких змінних умов.



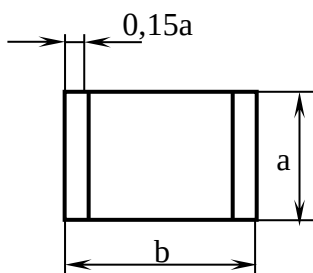
6.3.1.3. Модифікація

Виконання операцій, змінюючої команди або групи команд, що змінюють програму.



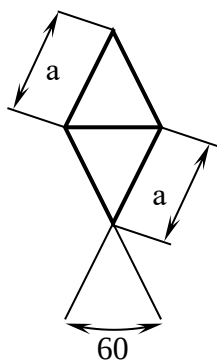
6.3.1.4. Зумовлений процес

Використання раніше створених і окремо описаних алгоритмів або програм.



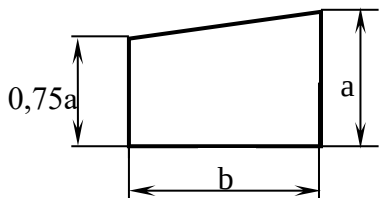
6.3.1.5. Сорткування

Впорядкування множини за заданими ознаками



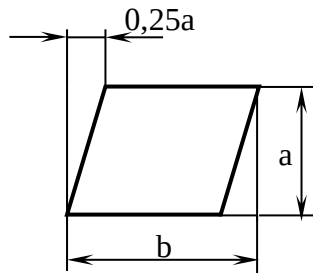
6.3.1.6. Ручне введення

Введення даних вручну за допомогою неавтономних пристроїв з клавіатурою, перемикачів, кнопок.



6.3.1.7. Введення-виведення

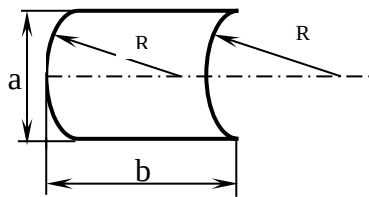
Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (виведення).



6.3.1.8. Неавтономна пам'ять

Уведення-виведення даних у разі використання пристрою, що запам'ятовує, керованого безпосередньо процесором.

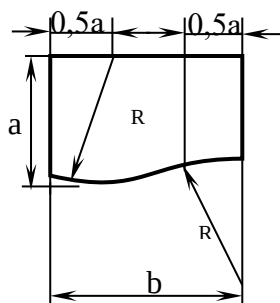
$$R=a$$



6.3.1.9. Документ

Введення-виведення даних, носієм яких служить папір.

$$R=a$$



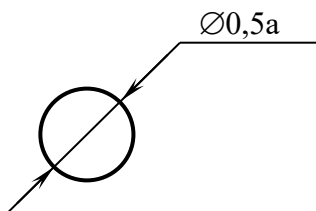
6.3.1.10. Лінія потоку

Вказівка послідовності зв'язків між символами.



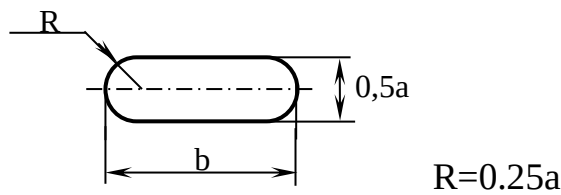
6.3.1.11. З'єднувач

Вказівка зв'язку між перерваними лініями потоку, що зв'язують символи.



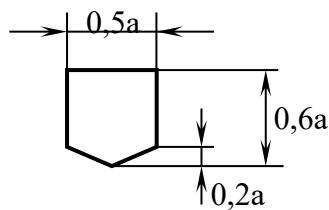
6.3.1.12. Пуск - зупинка

Почало, кінець, переривання процесу обробки даних або виконання програми.



6.3.1.13. Міжсторінковий з'єднувач

Вказівка зв'язку між роз'єднаними частинами схем алгоритмів і програм, розташованих на різних листах.



6.3.1.14. Співвідношення геометричних елементів символів

Розмір a повинен вибиратися з ряду 10, 15, 20 мм. Допускається збільшувати розмір a на число, кратне 5. Розмір b рівний $1.5a$.

Напрям ліній потоку за умовчанням (без вказівки стрілок) - вправо і вниз. У випадку, якщо потік направлений вліво або вгору, слід указувати стрілки на лінії потоку. Не допускається перетин ліній потоку. Якщо це необхідно, слід користуватися внутрішньо сторінковим з'єднувачем.

6.4. Порядок комплектування документації до курсової роботи

Документація (пояснювальна записка) комплектується в такій послідовності:

- титульний лист (приклад приведений в Додатку А);
- завдання на курсову роботу;
- реферат;
- зміст (приклад приведений в Додатку Б);
- основна частина;
- список використаних джерел (приклад приведений в Додатку В);
- додатки.

До додатках подається:

- блок-схема алгоритму роботи програми (фрагмент блок-схеми алгоритму програми приведений в Додатку Д);
- початковий текст коду програми ;
- результати роботи програми (у вигляді рисунків - знімків екрану).

Усі документи приводяться до формату А4 і брошуруються.

До пояснювальної записки надається DVD-RW диск, на якому міститься:

- 1) пояснювальна записка з додатками в форматі MS Word;
- 2) текст програми;
- 3) файл виконавчого модуля;
- 4) результати роботи програми, якщо програма їх генерує, як окремі об'єкти.

7. ПІДГОТОВКА КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ І ЗАХИСТ

Курсова робота закінчується захистом. За 10 днів до початку захисту визначається час і місце її проведення. Перенос дня захисту курсової роботи можливий тільки при наявності поважних причин. Курсові роботи допускаються до захисту тільки з дозволу керівника. Під час захисту не тільки перевіряється рівень виконання студентом роботи і глибина розуміння їм виконаної роботи, але також здобувається досвід виступу перед аудиторією з всебічним обґрунтуванням запропонованих рішень інженерної задачі. Для досягнення цих цілей захист проводиться перед спеціальною комісією з двох-трьох викладачів, виділених кафедрою, при особистій участі керівника курсової роботи, у присутності студентів навчальної групи. Такий публічний захист дозволяє також ознайомити групу з роботою її членів і виробити однакові вимоги до курсових робіт у керівників.

При підготовці до захисту студент повинний скласти план і написати тези доповіді з викладом основних положень роботи. Належну увагу необхідно приділити техніко-економічним питанням. При цьому варто звернути увагу на точність і послідовність викладу доповіді. Матеріал повинен бути викладений протягом 5-7 хв. і супроводжуватися поясненням схем, що відображають суть проробленої роботи. При захисті комплексних (групових) робіт представляють схеми всієї проектної групи.

Захист курсової роботи проводиться перед комісією в такому порядку:

- 1) повідомлення студента про зміст роботи і її реалізації;
- 2) питання членів комісії і присутніх;
- 3) відповіді студента на задані питання;
- 4) зауваження членів комісії;
- 5) відповіді студента на зауваження і при наявності заперечень, коротке обґрунтування своєї точки зору;
- 6) висновок комісії з оцінки роботи.

У доповіді необхідно розповісти про цілі і завдання, що ставляться перед студентом; шляхи їх рішення; про основні отримані результати; відзначити

важливі й оригінальні рішення в роботі; сформулювати висновки і надати рекомендації з практичного використання результатів проектування.

Питання задаються тільки по тим розділам теорії, на яких ґрунтується робота. У процесі захисту студент повинний показати гарні знання відповідно до тих питань, що порушені в курсовій роботі, виявити уміння логічно мислити, переконливо обґрунтувати і відстояти (при необхідності) свою точку зору. На захисті доречні питання, що дозволяють перевірити знання по суміжних дисциплінах, на яких ґрунтується розробка даної роботи.

У результаті захисту курсова робота, відповідно до діючого положення про курсові роботи, іспити і заліки, оцінюється рейтинговою оцінкою. Загальна трудомісткість курсової роботи, як форми підсумкового контролю – 100 балів. Курсова робота, як форма підсумкового контролю виставляється при наявності рейтингової оцінки понад 59 балів. Позитивна оцінка заноситься в залікову книжку за підписом керівника роботи. Критерії оцінювання наведені в Таблиці 7.1.

Таблиця 7.1 – Оцінювані параметри і кількість балів

№	Оцінюваний параметр	Кількість балів
1.	Якість оформлення роботи у відповідності до вимог (методичні рекомендації)	5
2.	Теоретична частина:	25
	<i>повнота опрацювання теоретичних питань роботи та наукова новизна</i>	10
	<i>наявність публікацій/апробацій</i>	10
	<i>участь у Всеукраїнських конкурсах студентських робіт</i>	5
3.	Документування архітектури розробленого програмного забезпечення у відповідності до стандарту <i>Software Architecture Document</i> (додаток Е)	10
4.	Програмне забезпечення:	30
	<i>рівень інженерної та технологічної реалізації програмного забезпечення;</i>	15
	<i>наявність промислового впровадження/продуктивної експлуатації</i>	15
5.	Захист роботи:	30
	<i>- якість доповіді та презентації;</i>	15
	<i>- відповіді на питання</i>	15
Загалом сумарний бал за роботу		100

* Відсоток використання існуючих модулів (бібліотек) повинен складати $\leq 30\%$

** Без демонстрації програмного забезпечення робота приймається лише за наявності витягу із засідання кафедри про її фундаментальний теоретичний напрям.

При оцінці комісія враховує: якість виконання роботи, зміст повідомлення студента і його відповідей на питання, якість графічних робіт, ступінь самостійності й ініціативи, що виявлена студентом при розробці теми, грамотність викладу матеріалу, володіння науково-технічною термінологією.

Критерії оцінювання захисту роботи:

24-30 балів. Виставляється, якщо студент вміє творчо мислити, узагальнювати вивчений матеріал, використовує для цього теоретичні положення і конкретні параметри:

чітко, вичерпно, глибоко розуміє і знає матеріал і вміє їм користуватися в межах вимог програми курсу;

вміє вести співбесіду з членами комісії, при відповідях на навідні і додаткові питання самостійно виправляє окремі допущені неточності;

якісно й у повному обсязі виконав розрахункові і графічні роботи, що входять у курсову роботу.

16-23 балів. Виставляється тоді, коли за перерахованими показниками маються окремі несуттєві недоліки. Наприклад, чіткість і повнота звітів не цілком достатні, маються дрібні недоліки в розрахунках і конструкторській документації, деяка недбалість оформлення і т.д.

8-15 балів. Виставляється при наявності істотних недоліків за перерахованими показниками, що студент самостійно виправити не міг, навіть якщо окремі відповіді були гарними або відмінними.

0-7 балів. Виставляється тоді, коли студент виявив нерозуміння і незнання основного змісту курсової роботи, допустив грубі помилки в пояснювальній записці, розрахунках, документації, не зміг правильно відповісти на питання комісії.

За рейтинговою оцінкою визначається оцінка за шкалою ECTS та традиційна чотирибальна національна оцінка відповідно до Таблиці 7.2.

Таблиці 7.2

Рейтингова оцінка	Оцінка за шкалою ECTS	Оцінка за національною шкалою
91-100	A - відмінно	5 - відмінно
81-90	B - дуже добре	4 - добре
71-80	C - добре	
61-70	D - задовільно	3 - задовільно
51-60	E - достатньо	
41-50	FX - незадовільно (з можливістю допуску до підсумкового контролю за дисципліну)	2 - незадовільно
1-40	F - незадовільно	

Курсові роботи, що мають теоретичну і практичну цінність, можуть бути представлені на конкурс студентських наукових праць, а також передані в промисловість для впровадження.

Після захисту студент складає пояснювальну записку і підшиває у швидкозшивач.

Для підшивки пробиваються отвори на лівій стороні листа. Зброшурована робота здається керівникові для передачі до архіву.

Студент, що не виконав курсову роботу в термін, допрацьовує її самостійно. Консультації в цьому випадку організовуються з урахуванням причин відставання. Рішення про організації консультацій виносить кафедра на підставі письмової заяви студента. При цьому кафедра визначає кількість годин і форму проведення консультацій, графік проектування і термін його виконання.

Відповідно до заяви студента завідувач кафедрою призначає день засідання комісії для захисту роботи.

У випадку одержання незадовільної оцінки на захисті студент одержує нове завдання на курсову роботу для повторного проходження проектування.

8. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1.ДСТУ 3008-2015: Державний стандарт України «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення». – Київ:ДП «УкрНПНЦ», 2016. – 25 с.
2. Положення про організацію освітнього процесу у Київському національному університеті ім. Тараса Шевченка. – К.: 2008. – 124 с.
- 3.Бюлетень вищої атестаційної комісії України – Київ, - №6, 2007 р.

Додаток А – Приклад оформлення титульного листа

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Кафедра програмних систем і технологій

РОЗРОБКА ПРОГРАМИ ДЛЯ СОРТУВАННЯ ВЕЛИКИХ ФАЙЛІВ

Курсова робота

Пояснювальна записка

Виконавець

студент групи ПЗм-21

Іван ІВАНЕНКО

(підпис, дата)

Керівник

доцент кафедри ПСТ

к.т.н., с.н.с. Степан СТЕПАНЕНКО

(підпис, дата)

Київ-2023

ЗМІСТ

Вступ	5
1. Файли	6
1.1 Властивості файлів	8
1.2 Типи файлів	12
1.3 Операції з файлами	13
1.4 Функції роботи з файлами	15
2. Методи сортування	18
2.1 Поняття сортування	18
2.2 Оцінка методів сортування та вимір методу для програмної реалізації	25
2.3 Функції, що використовуються для програмної реалізації методу сортування	27
3. Опис алгоритму програми	28
4. Опис роботи програми	29
5. Інструкція користувачу програми	30
Висновок	32
Список використаних джерел	33
Додаток А - Алгоритм реалізації програми	34
Додаток Б - Тест програми	35
Додаток В – Інтерфейс роботи з програмою	38
Додаток В - Результати роботи програми	40

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

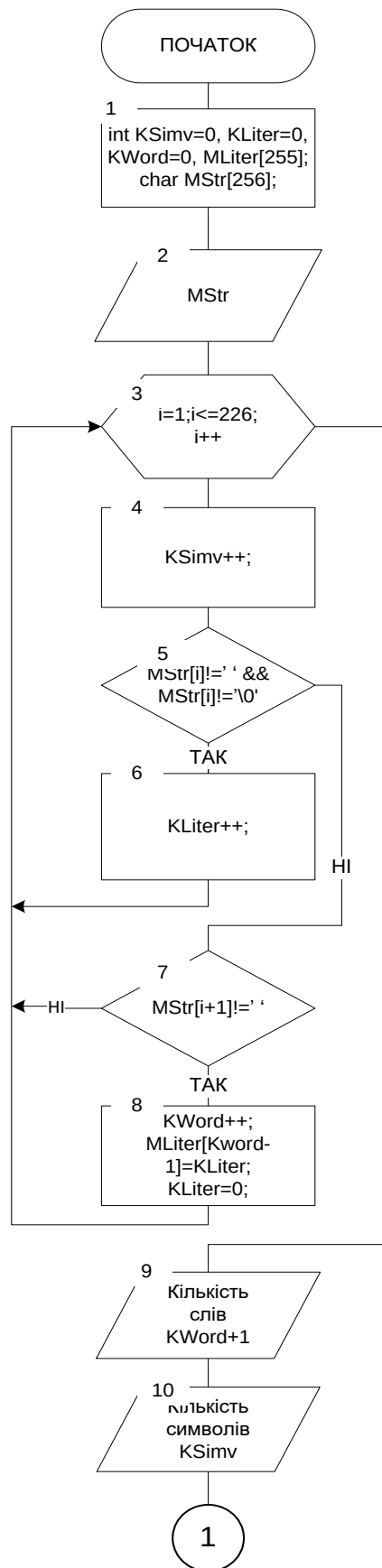
1. Бойко В. І. Схемотехніка електронних систем. Кн. 2. Цифрова схемотехніка : підручник / Бойко В. І. — К. : Вища шк., 2004. — 423 с.
2. Комплекс «еФізика» [електронний ресурс] / Ю. П. Бендес — Режим доступу: <http://efizika.org.ua/complex/>

Додаток Г – Алгоритм програми (приклад)

1. Оголошуємо:
 - 1) символьний масив для збереження строчки;
 - 2) цілочисловий масив для збереження кількості букв у кожному слові;
 - 3) змінну-лічильник для кількості символів у строчці;
 - 4) змінну-лічильник для кількості слів;
 - 5) змінну-лічильник для кількості букв у слові.
2. Обнуляємо оголошені змінні.
3. Зчитуємо строчку з консолі і записуємо до символьного масиву.
4. В циклі по одному символу аналізуємо символьний масив і збільшуємо лічильник символів на 1:
 - 1) якщо розглянутий символ не є пробілом, то лічильник букв у слові збільшуємо на 1;
 - 2) якщо розглянутий символ пробіл, а наступний за ним символ не є пробілом і не є 0-символом
 - додаємо одиницю до лічильника слів;
 - записуємо до цілочислового масиву значення кількості букв у слові у відповідності до номеру слова в строчці;
 - обнуляємо лічильник букв у слові;
 - 3) якщо розглянутий символ є 0-символом, що відображає кінець строчки, то перериваємо цикл аналізу символьного масиву.
5. Виводимо кількість слів у строчці на консоль.
6. У циклі виводимо кількість букв у кожному слові.

Блок-схема алгоритму програми приведена в Додатку Д.

Додаток Д – Фрагмент блок-схеми алгоритму програми (приклад)



<Insert OrganizationName>

**<Insert Project Name>
Software Architecture Document (SAD)**

CONTENT OWNER: <Insert Name>

DOCUMENT NUMBER:

-
-
-
-
-
-

RELEASE/REVISION:

-
-
-
-
-
-

RELEASE/REVISION DATE:

-
-
-
-
-
-

BACKGROUND

This template is based on the Software Engineering Institute's "View and Beyond" method for documenting software architectures, as described in Clements, et al., [Documenting Software Architecture: Views and Beyond](#) (Addison Wesley, 2002). The current version is available for [free download](#) from the SEI's architecture web site.

TIPS FOR USING THIS TEMPLATE

To create an instance of this document:

- Insert relevant information on cover sheet and in placeholders throughout.
- Insert relevant information in page header: Move to a page of the body of the report, select *View > Header and Footer* from the main menu, and then replace relevant information in the header box at the top of the page.

To update the contents and page numbers in the Table of Contents, List of Figures, and List of Tables:

- Position the cursor anywhere in the table to be updated.
- Click the *F9* function key.
- Answer "Update entire table".

To insert a figure or table caption:

- From the main menu, choose *Insert > Reference > Caption* and then either *Figure* or *Table* as needed.
- Click the OK button.
- Add a colon and a tab stop after the figure number in the caption itself.
- The caption should use the *Caption* style.
- Add a colon and a tab stop after the table/figure number in the caption itself.

TIPS FOR MAKING YOUR DOCUMENT MORE READABLE

- A gray box containing *CONTENTS OF THIS SECTION* is provided at the beginning of most sections and subsections. After determining what specific information will be included in your document, you can remove this gray box or leave it to serve as a quick-reference section overview for your readers. In the case that text has been provided in the template, inspect it for relevance and revised as necessary.
- Consider hyperlinking key words used in the document with their entries in the [Glossary](#) or other location in which they are defined. Choose *Insert > Hyperlink*.
- Don't leave blank sections in the document. Mark them "To be determined" (ideally with a promise of a date or release number by which the information will be provided) or "Not applicable."
- Consider packaging your SAD as a multi-volume set of documentation. It is often helpful to break your documentation into more than one volume so that the document does not become unwieldy. There are many ways that this can be accomplished. The structuring of the document must support the needs of the intended audience and must be determined in the context of the project. Each document that you produce should include the date of issue and status; draft, baseline, version number, name of issuing organization; change history; and a summary. A few decomposition options are:
 - *A 2-Volume approach*: Separate the documentation into two volumes; one that contains the views of the software architecture and one that contains everything else. A common variant of this approach has one volume per view, and one volume for everything else.
 - *A 3-Volume approach*: Document organizational policies, procedures, and the directory in one volume, system specific overview material in a second, and view documentation in a third.
 - *A 4-Volume approach*: Create one volume for each viewtype [Clements 2002] (module, component-and-connector, allocation) that contains the documentation for the relevant [views](#). Include all of the other information in the fourth volume.
 - Software interfaces are often documented in a separate volume.

In any case, the information should be arranged so that readers begin with the volume containing the Documentation Roadmap (Section 1 in this template).

Table of Contents

1	Documentation Roadmap	4
1.1	Document Management and Configuration Control Information	4
1.2	Purpose and Scope of the SAD	4
1.3	Viewpoint Definitions	6
1.3.1	<Insert name of viewpoint> Viewpoint Definition	7
1.3.1.1	Abstract	7
1.3.1.2	Stakeholders and Their Concerns Addressed	7
1.3.1.3	Elements, Relations, Properties, and Constraints	7
1.3.1.4	Language(s) to Model/Represent Conforming Views	7
2	Architecture Background	8
2.1	Problem Background	8
2.1.1	System Overview	8
2.1.2	Goals and Context	8
2.1.3	Significant Driving Requirements.....	8
2.2	Solution Background	8
2.2.1	Architectural Approaches	8
2.2.2	Analysis Results	9
2.2.3	Requirements Coverage	9
3	Referenced Materials	10
4	Directory	11
4.1	Index.....	11
4.2	Glossary.....	11
4.3	Acronym List	12
5	Sample Figures & Tables	12

List of Figures

Figure 1: Sample Figure 13

List of Tables

Table 1: Stakeholders and Relevant Viewpoints..... 6

Table 2: Sample Table 13

1. Documentation Roadmap

The Documentation Roadmap should be the first place a new reader of the SAD begins. But for new and returning readers, it is intended to describe how the SAD is organized so that a reader with specific interests who does not wish to read the SAD cover-to-cover can find desired information quickly and directly.

Sub-sections of Section 1 include the following.

- Section 1.1 (“Document Management and Configuration Control Information”) explains revision history. This tells you if you’re looking at the correct version of the SAD.
- Section 1.2 (“Purpose and Scope of the SAD”) explains the purpose and scope of the SAD, and indicates what information is and is not included. This tells you if the information you’re seeking is likely to be in this document.
- Section **Ошибка! Источник ссылки не найден.** (“**Ошибка! Источник ссылки не найден.**”) explains the information that is found in each section of the SAD. This tells you what section(s) in this SAD are most likely to contain the information you seek.
- Section **Ошибка! Источник ссылки не найден.** (“**Ошибка! Источник ссылки не найден.**”) explains the stakeholders for which the SAD has been particularly aimed. This tells you how you might use the SAD to do your job.
- Section 1.3 (“Viewpoint Definitions”) explains the *viewpoints* (as defined by IEEE Standard 1471-2000) used in this SAD. For each viewpoint defined in Section 1.3, there is a corresponding view defined in Section **Ошибка! Источник ссылки не найден.** (“**Ошибка! Источник ссылки не найден.**”). This tells you how the architectural information has been partitioned, and what views are most likely to contain the information you seek.
- Section **Ошибка! Источник ссылки не найден.** (“**Ошибка! Источник ссылки не найден.**”) explains the standard organization used to document architectural views in this SAD. This tells you what section within a view you should read in order to find the information you seek.

1.1. Document Management and Configuration Control Information

CONTENTS OF THIS SECTION: This section identifies the version, release date, and other relevant management and configuration control information associated with the current version of the document. Optional items for this section include: change history and an overview of significant changes from version to version.

- Revision Number: << >>
- Revision Release Date: << >>
- Purpose of Revision: << >>
- Scope of Revision: <<*list sections or page numbers that have been revised; provide a summary overview of the differences between this release and the previous one.*>>

1.2. Purpose and Scope of the SAD

CONTENTS OF THIS SECTION: This section explains the SAD's overall purpose and scope, the criteria for deciding which design decisions are architectural (and therefore documented in the SAD), and which design decisions are non-architectural (and therefore documented elsewhere).

This SAD specifies the software architecture for <insert scope of SAD>. All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents.

What is software architecture? The software architecture for a system¹ is the structure or structures of that system, which comprise software elements, the externally-visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refers to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. This definition provides the basic litmus test for what information is included in this SAD, and what information is relegated to downstream documentation.

Elements and relationships. The software architecture first and foremost embodies information about how the elements relate to each other. This means that architecture specifically omits certain information about elements that does not pertain to their interaction. Thus, a software architecture is an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. Elements interact with each other by means of interfaces that partition details about an element into public and private parts. Software architecture is concerned with the public side of this division, and that will be documented in this SAD accordingly. On the other hand, private details of elements—details having to do solely with internal implementation—are not architectural and will not be documented in a SAD.

Multiple structures. The definition of software architecture makes it clear that systems can and do comprise more than one structure and that no one structure holds the irrefutable claim to being the architecture. The neurologist, the orthopedist, the hematologist, and the dermatologist all take a different perspective on the structure of a human body. Ophthalmologists, cardiologists, and podiatrists concentrate on subsystems. And the kinesiologist and psychiatrist are concerned with different aspects of the entire arrangement's behavior. Although these perspectives are pictured differently and have very different properties, all are inherently related; together they describe the architecture of the human body. So it is with software. Modern systems are more than complex enough to make it difficult to grasp them all at once. Instead, we restrict our attention at any one moment to one (or a small number) of the software system's structures. To communicate meaningfully about an architecture, we must make clear which structure or structures we are discussing at the moment—which *view* we are taking of the architecture. Thus, this SAD follows the principle that documenting a software architecture is a matter of documenting the relevant views and then documenting information that applies to more than one view.

For example, all non-trivial software systems are partitioned into implementation units; these units are given specific responsibilities, and are the basis of work assignments for programming teams. This kind of element will comprise programs and data that software in other implementation units can call or access, and programs and data that are private. In large projects, the elements will almost certainly be subdivided for assignment to sub-teams. This is one kind of structure often used to describe a system. It is a very static structure, in that it focuses on the way the system's functionality is divided up and assigned to implementation teams.

Other structures are much more focused on the way the elements interact with each other at runtime to carry out the system's function. Suppose the system is to be built as a set of parallel processes. The set of processes that will exist at runtime, the programs in the various implementation units described previously that are strung

¹ Here, a system may refer to a system of systems.

together sequentially to form each process, and the synchronization relations among the processes form another kind of structure often used to describe a system.

None of these structures alone is *the* architecture, although they all convey architectural information. The architecture consists of these structures as well as many others. This example shows that since architecture can comprise more than one kind of structure, there is more than one kind of element (e.g., implementation unit and processes), more than one kind of interaction among elements (e.g., subdivision and synchronization), and even more than one context (e.g., development time versus runtime). By intention, the definition does not specify what the architectural elements and relationships are. Is a software element an object? A process? A library? A database? A commercial product? It can be any of these things and more.

These structures will be represented in the views of the software architecture that are provided in Section **Ошибка! Источник ссылки не найден..**

Behavior. Although software architecture tends to focus on structural information, *behavior of each element is part of the software architecture* insofar as that behavior can be observed or discerned from the point of view of another element. This behavior is what allows elements to interact with each other, which is clearly part of the software architecture and will be documented in the SAD as such. Behavior is documented in the element catalog of each view.

1.3. Viewpoint Definitions

CONTENTS OF THIS SECTION: This section provides a short textual definition of a viewpoint and how the concept is used in this SAD. The section describes viewpoints that may be used in the SAD. The specific viewpoints will be tailored by the organization.

The SAD employs a stakeholder-focused, multiple view approach to architecture documentation, as required by ANSI/IEEE 1471-2000, the recommended best practice for documenting the architecture of software-intensive systems [IEEE 1471].

As described in Section 1.2, a software architecture comprises more than one software structure, each of which provides an engineering handle on different system qualities. A *view* is the specification of one or more of these structures, and documenting a software architecture, then, is a matter of documenting the relevant views and then documenting information that applies to more than one view [Clements 2002].

ANSI/IEEE 1471-2000 provides guidance for choosing the best set of views to document, by bringing stakeholder interests to bear. It prescribes defining a set of viewpoints to satisfy the stakeholder community. A viewpoint identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. A view, then, is a viewpoint applied to a system. It is a representation of a set of software elements, their properties, and the relationships among them that conform to a defining viewpoint. Together, the chosen set of views show the entire architecture and all of its relevant properties. A SAD contains the viewpoints, relevant views, and information that applies to more than one view to give a holistic description of the system.

The remainder of Section 1.5 defines the viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns.

Table 1: Stakeholders and Relevant Viewpoints

Stakeholder	Viewpoint(s) that apply to that class of stakeholder's concerns

Stakeholder	Viewpoint(s) that apply to that class of stakeholder's concerns

1.3.1. <Insert name of viewpoint> Viewpoint Definition

There will be one of these subsections for each viewpoint defined. The subsections are as follows:

- **Abstract:** A brief overview of the viewpoint
- **Stakeholders and their concerns addressed:** This section describes the stakeholders and their concerns that this viewpoint is intended to address. Listed are questions that can be answered by consulting views that conform to this viewpoint. Optionally, the section includes significant questions that cannot be answered by consulting views conforming to this viewpoint.
- **Elements, relations, properties, and constraints:** This section defines the types of elements, the relations among them, the significant properties they exhibit, and the constraints they obey for views conforming to this viewpoint.
- **Language(s) to model/represent conforming views:** This section lists the language or languages that will be used to model or represent views conforming to this viewpoint, and cite a definition document for each.
- **Applicable evaluation/analysis techniques and consistency/completeness criteria:** This section describes rules for consistency and completeness that apply to views in this viewpoint, as well as any analysis of evaluation techniques that apply to the view that can be used to predict qualities of the system whose architecture is being specified.
- **Viewpoint source:** This section provides a citation for the source of this viewpoint definition, if any.

Following is an example of a viewpoint definition.

1.5.1 Module decomposition viewpoint definition

1.5.1.1 Abstract. Views conforming to the module decomposition viewpoint partition the system into a unique non-overlapping set of hierarchically decomposable implementation units (*modules*).

1.5.1.2 Stakeholders and Their Concerns Addressed. Stakeholders and their concerns addressed by this viewpoint include

- project managers, who must define work assignments, form teams, and formulate project plans and budgets and schedules;
- COTS specialists, who need to have software elements defined as units of functionality, so they can search the marketplace and perform trade studies to find suitable COTS candidates;
- testers and integrators who use the modules as their unit of work;
- configuration management specialists who are in charge of maintaining current and past versions of the elements;
- system build engineers who use the elements to produce a running version of the system;
- maintainers, who are tasked with modifying the software elements;
- implementers, who are required to implement the elements;
- software architects for those software elements sufficiently large or complex enough to warrant their own software architectures;
- the customer, who is concerned that projected changes to the system over its lifetime can be made economically by confining the effects of each change to a small number of elements.

1.5.1.3 Elements, Relations, Properties, and Constraints. Elements of the module decomposition viewpoint are modules, which are units of implementation that provide defined functionality. Modules are hierarchically decomposable; hence, the relation is "is-part-of." Properties of elements include their names, the functionality assigned to them (including a statement of the quality attributes associated with that functionality), and their software-to-software interfaces. The module properties may include requirements allocation, supporting requirements traceability.

1.5.1.4 Language(s) to Model/Represent Conforming Views. Views conforming to the module decomposition viewpoint may be represented by (a) plain text using indentation or outline form [Clements 2002]; (b) UML, using subsystems or classes to represent elements and "is part of" or nesting to represent the decomposition relation.

1.3.1.1. Abstract

1.3.1.2. Stakeholders and Their Concerns Addressed

1.3.1.3. Elements, Relations, Properties, and Constraints

1.3.1.4. Language(s) to Model/Represent Conforming Views

2. Architecture Background

2.1. Problem Background

CONTENTS OF THIS SECTION: The sub-parts of Section 2.1 explain the constraints that provided the significant influence over the architecture.

2.1.1. System Overview

CONTENTS OF THIS SECTION: This section describes the general function and purpose for the system or subsystem whose architecture is described in this SAD.

2.1.2. Goals and Context

CONTENTS OF THIS SECTION: This section describes the goals and major contextual factors for the software architecture. The section includes a description of the role software architecture plays in the life cycle, the relationship to system engineering results and artifacts, and any other relevant factors.

2.1.3. Significant Driving Requirements

CONTENTS OF THIS SECTION: This section describes behavioral and quality attribute requirements (original or derived) that shaped the software architecture. Included are any scenarios that express driving behavioral and quality attribute goals, such as those crafted during a Quality Attribute Workshop (QAW) [Barbacci 2003] or software architecture evaluation using the Architecture Tradeoff Analysis MethodSM (ATAMSM) [Bass 2003].

2.2. Solution Background

CONTENTS OF THIS SECTION: The sub-parts of Section 2.2 provide a description of why the architecture is the way that it is, and a convincing argument that the architecture is the right one to satisfy the behavioral and quality attribute goals levied upon it.

2.2.1. Architectural Approaches

CONTENTS OF THIS SECTION: This section provides a rationale for the major design decisions embodied by the software architecture. It describes any design approaches applied to the software architecture, including the use of architectural styles or design patterns, when the scope of those approaches transcends any single architectural view. The section also provides a rationale for the selection of those approaches. It also describes any significant alternatives that were seriously considered and why they were ultimately rejected. The section describes any relevant COTS issues, including any associated trade studies.

SM Quality Attribute Workshop and QAW and Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

2.2.2. Analysis Results

CONTENTS OF THIS SECTION: This section describes the results of any quantitative or qualitative analyses that have been performed that provide evidence that the software architecture is fit for purpose. If an Architecture Tradeoff Analysis Method evaluation has been performed, it is included in the analysis sections of its final report. This section refers to the results of any other relevant trade studies, quantitative modeling, or other analysis results.

2.2.3. Requirements Coverage

CONTENTS OF THIS SECTION: This section describes the requirements (original or derived) addressed by the software architecture, with a short statement about where in the architecture each requirement is addressed.

3. Referenced Materials

CONTENTS OF THIS SECTION: This section provides citations for each reference document. Provide enough information so that a reader of the SAD can be reasonably expected to locate the document.

Barbacci 2003	Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. <i>Quality Attribute Workshops (QAWs)</i> , Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. < http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html >.
Bass 2003	Bass, Clements, Kazman, <i>Software Architecture in Practice</i> , second edition, Addison Wesley Longman, 2003.
Clements 2001	Clements, Kazman, Klein, <i>Evaluating Software Architectures: Methods and Case Studies</i> , Addison Wesley Longman, 2001.
Clements 2002	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, <i>Documenting Software Architectures: Views and Beyond</i> , Addison Wesley Longman, 2002.
IEEE 1471	ANSI/IEEE-1471-2000, <i>IEEE Recommended Practice for Architectural Description of Software-Intensive Systems</i> , 21 September 2000.

4. Directory

4.1. Index

CONTENTS OF THIS SECTION: This section provides an index of all element names, relation names, and property names. For each entry, the following are identified:

- the location in the SAD where it was defined
- each place it was used

Ideally, each entry will be a hyperlink so a reader can instantly navigate to the indicated location.

4.2. Glossary

CONTENTS OF THIS SECTION: This section provides a list of definitions of special terms and acronyms used in the SAD. If terms are used in the SAD that are also used in a parent SAD and the definition is different, this section explains why.

Term	Definition
software architecture	The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.
view	A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.
view packet	The smallest package of architectural documentation that could usefully be given to a stakeholder. The documentation of a view is composed of one or more view packets.

viewpoint	A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view.
-----------	---

4.3. Acronym List

API	Application Programming Interface; Application Program Interface; Application Programmer Interface
ATAM	Architecture Tradeoff Analysis Method
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CORBA	Common object request broker architecture
COTS	Commercial-Off-The-Shelf
EPIC	Evolutionary Process for Integrating COTS-Based Systems
IEEE	Institute of Electrical and Electronics Engineers
KPA	Key Process Area
OO	Object Oriented
ORB	Object Request Broker
OS	Operating System
QAW	Quality Attribute Workshop
RUP	Rational Unified Process
SAD	Software Architecture Document
SDE	Software Development Environment
SEE	Software Engineering Environment
SEI	Software Engineering Institute Systems Engineering & Integration Software End Item
SEPG	Software Engineering Process Group
SLOC	Source Lines of Code
SW-CMM	Capability Maturity Model for Software
CMMI-SW	Capability Maturity Model Integrated - includes Software Engineering
UML	Unified Modeling Language

5. Sample Figures & Tables

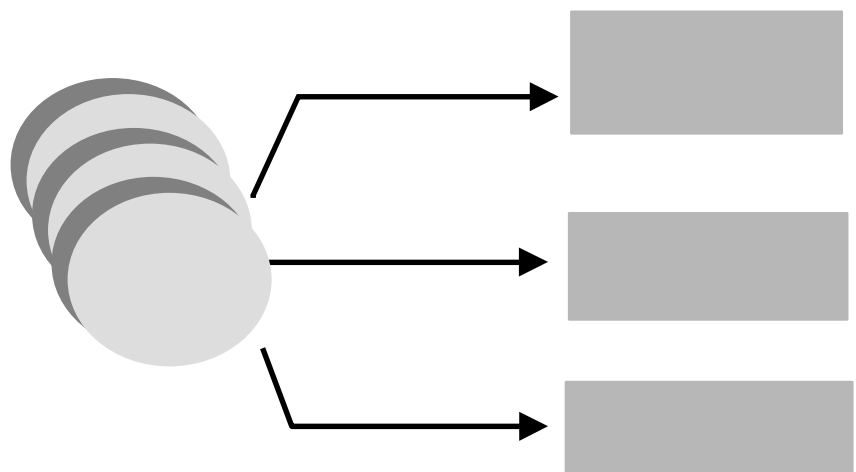


Figure 1: Sample Figure

Table 2: Sample Table

Table Heading	Table Heading	Table Heading	Table Heading
Table Body	Table Body	Table Body	Table Body
Table Body	Table Body	Table Body	Table Body
Table Body	Table Body	Table Body	Table Body
Table Body	Table Body	Table Body	Table Body

Appendix A Appendices

CONTENTS OF THIS SECTION: Appendices may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data, API specification). As applicable, each appendix is referenced in the main body of the document where the data would normally have been provided. Appendices may be bound as separate documents for ease in handling. If your SAD has no appendices, delete this page.

A.1 Heading 2 - Appendix

A.2 Heading 2 - Appendix