SHOULD BE REPLACED ON REQUIRED TITLE PAGE

Instruction

1. Open needed docx template (folder "title"/<your department or bach if bachelor student>.docx).

2. Put Thesis topic, supervisor's and your name in appropriate places on both English and Russian languages.

3. Put current year (last row).

4. Convert it to "title.pdf," replace the existing one in the root folder.

Оглавление

Список таблиц

Список иллюстраций

Аннотация

Данная работа посвящена сравнительному анализу объектных моделей современных языков программирования, включая C, C++, Golang, Java, Python, Ruby, JavaScript, Scala, Smalltalk и Zonnon. Актуальность работы обусловлена широким распространением объектно-ориентированной парадигмы и значительным разнообразием в ее реализации, что создает сложности для осознанного выбора технологий и архитектурных решений. Целью исследования является выявление сходств, различий, сильных и слабых сторон различных объектных моделей, а также разработка на этой основе предложений по созданию усовершенствованной модели. В результате исследования систематизированы теоретические основы объектных моделей, проведен детальный разбор их реализации в выбранных языках с выявлением областей применения и типичных ошибок, а также предложена авторская модель, интегрирующая лучшие практики. Практическая значимость работы заключается в том, что ее результаты могут быть использованы разработчиками для выбора языка и парадигм, архитекторами — для проектирования систем.

Глава 1

Introduction

1.1 Spacing & Type

This is a section. This is a citation without brackets. and this is one with brackets **A**. Multiple **A**, **B**, **C** Here's a reference to a subsection: **??**. Citation of an online article **D**. Citation of an online proceeding **F**. The body of the text and abstract must be double-spaced except for footnotes or long quotations. Fonts such as Times Roman, Bookman, New Century Schoolbook, Garamond, Palatine, and Courier are acceptable and commonly found on most computers. The same type must be used throughout the body of the text. The font size must be 10 point or larger and footnotes[1] must be two sizes smaller than the text[2] but no smaller than eight points. Chapter, section, or other headings should be of a consistent font and size throughout the ETD, as should labels for illustrations, charts, and figures.

[1] This is a footnote.

[2] This is another footnote.

1.1.1 Creating a Subsection

Creating a Subsubsection

Creating a Subsubsection

Creating a Subsubsection

This is a heading level below subsubsection And this is a quote:

> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

This is a table:

TABLE 1.1
This Is a Table Example

A	B	C
a1	b1	c1
a2	b2	c2

A	B	C
a3	b3	c3
a4	b4	c4

The package "upgreek" allows us to use non-italicized lower-case greek letters. See for yourself: $\upbeta$, $\boldsymbol{\upbeta}$, $\updelta$, $\boldsymbol{\updelta}$. Next is a numbered equation:

$$\|\boldsymbol{X}\|_{2,1} = \underbrace{\sum_{j=1}^{n} f_j(\boldsymbol{X})}_{\text{convex}} = \sum_{j=1}^{n} \|\boldsymbol{X}_{\cdot,j}\|_2 \tag{1.1}$$

The reference to equation (**??**) is clickable.

1.2 Theorems, Corollaries, Lemmas, Proofs, Remarks, Definitions,and Examples

Theorem 1. *Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.*

Fig. 1.1. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

Доказательство. I'm a (very short) proof. □

Lemma 1. *I'm a lemma.*

Corollary 1. *I include a reference to Thm.* **??**.

Proposition 1. *I'm a proposition.*

Remark. I'm a remark.

Definition 1. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition.

Example. I'm an example.

1.3 Section with linebreaks in the name

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Глава 2

Literature Review

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum

libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2.1 Понятие и компоненты объектной модели

Объектная модель представляет собой фундаментальную основу, определяющую способ представления и поддержки языком программирования объектов, классов, наследования, инкапсуляции, полиморфизма и других связанных абстракций. Согласно общепринятой терминологии, объектная модель (object model) имеет два взаимосвязанных значения: (1) свойства объектов в целом в конкретном языке программирования, технологии, нотации или методологии, которая использует эти объекты; (2) набор интерфейсов или классов, через которые программа может исследовать и манипулировать некоторыми специфическими частями своего мира. Примерами объектных моделей являются объектные модели Java, Component Object Model (COM) или Object-Modeling Technique (OMT). Такие объектные модели обычно определяются с использованием таких концепций, как класс, обобщённая функция, сообщение, наследование, полиморфизм и инкапсуляция.

В контексте объектно-ориентированного программирования Cardelli и Wegner в работе "On Understanding Types, Data Abstraction, and Polymorphism"определил объектно-ориентированные языки через три ключевых требования **cardelli1985understa** Язык считается объектно-ориентированным, если он удовлетворяет условиям:

- Обеспечивает поддержку объектов, которые представляют собой абстракции данных с интерфейсом из именованных операций и скрытым внутренним состоянием
- Объекты в языке имеют связанный с ними объектный тип
- Типы могут наследовать атрибуты от супертипов.

Эти требования они сформулировали как формулу:
$object - oriented = dataabstractions + objecttypes + typeinheritance$

Booch в своей фундаментальной работе “Object-Oriented Analysis and Design with Applications” описывает объектную модель как концептуальное представление организованной сложности программного обеспечения **booch2008object**.

> Booch (p. 40-41) concludes that the conceptual framework for anything object-oriented is the object model—a conceptual representation of the organized complexity of software. It consists of four major elements, i.e. abstraction, encapsulation, modularity, and hierarchy. In addition, the object model contains three minor element, i.e. typing, concurrency, and persistence.

Это разделение на основные (*major*) и дополнительные (*minor*) элементы является принципиальным: без основных элементов модель перестаёт быть объектно-ориентированной; дополнительные элементы являются полезными, но не обязательными для поддержки объектной ориентации.

Формальная модель объектной модели была представлена Abadi и Cardelli в работе “A Theory of Objects”, где объекты рассматриваются как примити-

вы объектных исчислений **abadi2012**. Их подход демонстрирует, как можно формально объяснить как семантику объектов, так и их правила типизации, а также разработать все наиболее важные концепции объектно-ориентированных языков программирования: self, динамическую диспетчеризацию, классы, наследование, защищённые и приватные методы, прототипирование, подтипизацию, ковариантность и контравариантность, а также специализацию методов.

2.1.1 Основные компоненты объектной модели (Major Elements)

Абстракция (Abstraction)

Абстракция является одним из наиболее фундаментальных компонентов объектной модели. Согласно Tinelli в его учебных материалах **tinelli_object_model**, абстракция определяется как:

> "The process of identifying similarities between objects, situations or processes and ignoring their differences. A description, or specification, of something that emphasizes some details or properties while ignoring others. It focuses on the essential characteristics of something relative to a viewer's perspective."

Booch в свою очередь определяет абстракцию следующим образом **zschocke_object**

> "An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer."

Абстракция в контексте объектно-ориентированного программирования фокусируется на **внешнем представлении объекта**, определяя концептуальные границы по отношению к другим объектам. Она помогает отделить **поведение объекта от его реализации**. В объектно-ориентированном программировании поведение объекта может быть охарактеризовано через услуги (*services*), которые он предоставляет другим объектам (клиентам), а также операции, которые он может выполнять.

Объектная абстракция определяет **контракт** (*contract*), от которого зависят другие объекты и который должен соблюдаться данным объектом. Контракт устанавливает все предположения, которые клиент может сделать относительно поведения сервера **tinelli_object_model**.

Инкапсуляция (Encapsulation)

Инкапсуляция является процессом **компартментализации элементов абстракции**, которые составляют её структуру и поведение. Booch определяет инкапсуляцию следующим образом **zschocke_object_orientation**:

> “Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.”

Инкапсуляция служит для **отделения контрактного интерфейса абстракции от её реализации**. Согласно Tinelli **tinelli_object_model**:

> “Encapsulation. The abstraction of an object should precede any decisions about its implementation. Implementation details should

> not be accessible to clients. Encapsulation is the process of hiding such details."

В то время как абстракция фокусируется на том, как объект виден извне, инкапсуляция концентрируется на том, что может быть видно только изнутри объекта. Достигнутые результаты инкапсуляции **tinelli_object_model**:

- Greatly facilitates changes that do not impact the abstraction (i.e., the object's contract)
- Leads to a clear separation of concerns (contract vs way to honor it)
- Localizes design decisions likely to change

Абстракция и инкапсуляция являются **взаимодополняющими концепциями**: абстракция фокусируется на наблюдаемом поведении объекта, в то время как инкапсуляция фокусируется на реализации, которая обеспечивает это поведение.

Модульность (Modularity)

Модульность представляет собой свойство системы, которая была декомпозирована на набор связных и слабо связанных модулей. Согласно Tinelli **tinelli_object_model**:

> "Modularity. Modularization divides a software systems into components, modules. Modules may have connections to other modules but can be compiled separately, encapsulate sets of classes and objects, and have an interface and an implementation."

Критически важной является следующая точка **tinelli_object_model**:

> “Classes and objects define a system’s logical structure. Modules define a system’s physical structure. The two structures are by and large orthogonal.”

Модули характеризуются рядом важных свойств, согласно Zschocke **zschocke_object_orientation**. Объекты, когда используются как инструмент моделирования, должны быть близки к ментальной модели нашего мышления (понятность). Объекты должны поддерживать разложимость (деление на подобъекты) и композируемость (комбинация объектов для формирования больших объектов). Изменения в спецификациях должны приводить только к ограниченным изменениям всей системы (непрерывность). Защита предотвращает влияние на объекты каких-либо аномальных условий, которые происходят во время выполнения системы.

Иерархия (Hierarchy)

Иерархия представляет собой **ранжирование или упорядочивание абстракций**. Согласно Tinelli **tinelli_object_model**:

> “Hierarchy. A (partial) ordering of abstractions. Most important hierarchies: "is a"relation (class structure) and "part of"relation (object structure).”

Две наиболее важные иерархии в сложной системе это:

1. **Структура классов** (*class structure*) — иерархия “является” (*is a* hierarchy) или обобщение/специализация

2. **Структура объектов** (*object structure*) — иерархия “часть” (*part of* hierarchy) или целое/часть

Наследование (*inheritance*) является наиболее важной иерархией “является”. Booch определяет наследование следующим образом **booch1994**:

> “Inheritance is a relationship among classes wherein one class shares the structure and/or behavior defined in one (single inheritance) or more (multiple inheritance) other classes.”

Наследование описывает отношение между классами, при котором один класс разделяет структуру и/или поведение, определённое в одном или нескольких других классах. Наследование особенно эффективно для **управления эволюцией во времени**, позволяя системе адаптироваться к изменениям требований **zschocke_object_orientation**.

2.1.2 Дополнительные компоненты объектной модели (Minor Elements)

Типизация (Typing)

Типизация представляет собой **механизм принуждения класса объектов** таким образом, что объекты разных типов не могут быть взаимозаменяемыми или могут быть взаимозаменяемыми только ограниченным образом. Booch определяет типизацию следующим образом **zschocke_object_orientation**:

> “Typing is defined as ’the enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted arrays.”’

Согласно Tinelli, в современных объектно-ориентированных языках **tinelli_object_

> "Every class defines a type, consisting of all objects that are instances of that class. However, not all types are classes. E.g.: Java's basic types (int, bool, ...), Java's interfaces, Traits in Scala."

Различают **статическую и динамическую типизацию** **tinelli_object_model**:

- **Statically typed languages** enforce typing restrictions at compile time: the type of each expression denoting a value is determined and checked before running the program
- **Dynamically typed languages** enforce typing restrictions at run time: types are determined and checked as expressions are evaluated

Cardelli и Wegner подчёркивают защитную функцию типов **cardelli1985understand**

> "A type may be viewed as a set of clothes (or a suit of armor) that protects an underlying untyped representation from arbitrary or unintended use. It provides a protective covering that hides the underlying representation and constrains the way objects may interact with other objects."

Параллелизм (Concurrency)

Параллелизм (*concurrency*) является свойством, которое **отличает активный объект от неактивного**. Согласно Booch и описанию в литературе **zschocke_object_orientation**:

> "Objects can be characterized as asynchronous (sequential), guarded, or synchronous. Concurrency, then, describes whether an object continues processing after it has sent or received a message. This concept helps to distinguish an active object from one that is not active."

Параллелизм позволяет различным объектам действовать одновременно, используя взаимоисключающий доступ к общим ресурсам. Состояние активного объекта может быть изменено внутренними операциями, выполняющимися внутри самого объекта, в то время как пассивные объекты реализуются только как параллельный процесс с точками входа, соответствующими конкретным операциям объекта **zschocke_object_orientation**.

Персистентность (Persistence)

Персистентность представляет собой свойство объекта, благодаря которому **его существование выходит за рамки времени и/или пространства**. Booch определяет это следующим образом **zschocke_object_orientation**:

> "The idea of persistence describes the fact whether an object's existence will transcend over time and/or space. Thus, persistence describes whether an object will exist only during the single execution of a system or between individual executions and/or whether the location of an object has moved away from an address space in memory in which it was initially created."

Персистентность позволяет объектам сохранять своё состояние между различными запусками программы. Существует персистентность во времени

(объект продолжает существовать после завершения его создателя) и персистентность в пространстве (местоположение объекта перемещается из адресного пространства, в котором он был создан) **zschocke_object_orientation**.

2.1.3 Эволюция понимания объектной модели

Важно отметить, что понимание объектной модели эволюционировало с развитием языков программирования. Первоначально, концепции класса, инкапсуляции и наследования были введены в языке SIMULA в 1960-х годах разработчиками Dahl и Nygaard **zschocke_object_orientation**. Впоследствии Smalltalk (разработан Kay в 1970-х) стал первым полностью объектно-ориентированным языком, где всё рассматривается как объект **zschocke_object_orienta**

Классификация, предложенная Booch, синтезировала годы исследований в области объектно-ориентированного программирования и стала стандартной в академической литературе и практическом применении. Как отмечает Booch **booch1994**, принципы абстракции, инкапсуляции, модульности и иерархии были известны и раньше, но объектно-ориентированная парадигма привнесла их **синергию** — взаимное усиление и взаимодополнение этих концепций.

2.1.4 Выбранное определение объектной модели для данной диссертации

На основе проведённого обзора литературы для целей данной диссертации принимается следующее определение объектной модели:

Объектная модель — это концептуальная основа, интегрирующая

совокупность взаимосвязанных компонентов (четыре основных: абстракция, инкапсуляция, модульность, иерархия и три дополнительных: типизация, параллелизм, персистентность), которая определяет способ представления, организации и поддержки языком программирования объектов, классов, их взаимодействия, отношения наследования, механизмы полиморфизма и другие связанные абстракции для эффективного управления сложностью программных систем.

Глава 3

Methodology

Referencing other chapters **??**, **??**, **??**, **??** and **??**

TABLE 3.1
Simulation Parameters

A	B

Parameter	**Value**
Number of vehicles	$\mid \mathcal{V} \mid$
Number of RSUs	$\mid \mathcal{U} \mid$
RSU coverage radius	150 m
V2V communication radius	30 m
Smart vehicle antenna height	1.5 m
RSU antenna height	25 m
Smart vehicle maximum speed	v_{max} m/s
Smart vehicle minimum speed	v_{min} m/s
Common smart vehicle cache capacities	$[50, 100, 150, 200, 250]$ mb
Common RSU cache capacities	$[5000, 1000, 1500, 2000, 2500]$ mb

	A	B
Common backhaul rates		$[75, 100, 150]$ mb/s

...

Глава 4

Implementation

...

Глава 5

Evaluation and Discussion

...

Глава 6

Conclusion

...

Приложение А

Extra Stuff

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Приложение В

Even More Extra Stuff

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.