Содержание

[Введение 2](#_Toc319241689)

[1 Прототипирование программного обеспечения 3](#_Toc319241690)

[1.1 Жизненный цикл ПО 3](#_Toc319241691)

[1.2 Прототипирование как этап ЖЦ ПО 3](#_Toc319241692)

[1.3 Роль прототипирования в современных методологиях разработки ПО 3](#_Toc319241693)

[1.3 Понятие прототипа ПО 3](#_Toc319241694)

[1.4 Требования к прототипам ПО 3](#_Toc319241695)

[1.5 Выводы 3](#_Toc319241696)

# Введение

Быстрорастущий уровень современной компьютеризации общества сопровождается появлением нового класса программных инструментов – систем мониторинга. Основная задача подобных решений - систематический анализ и интерпретация протекающих в гетерогенной среде процессов. Полученные в результате мониторинга данные могут быть использованы как для улучшения процесса принятия решений, так и для выявления узких мест исследуемой системы.

Настоящая работа представляет собой исследование современных решений в области мониторинга, оценку их эффективности и применимости согласно выдвинутой модели требований, а также выводы о необходимости появления нового класса инструментов мониторинга, в виду неготовности существующих решений удовлетворять ранее выдвинутым требованиям.

Кроме того, в работе детально представлена предлагаемая авторами модель распределенной системы мониторинга гетерогенной среды, лишенная недостатков классических клиент-серверных систем и рассмотрена реализация каркаса распределенной системы мониторинга с точки зрения современных технологий программирования. Основополагающая идея предлагаемого подхода заключается в использовании механизма разработки и исполнения дополнительных модулей в процессе решения задач мониторинга, а также свойств распределенных систем в процессе эксплуатации.

# 1 Прототипирование архитектуры ПО

## 1.1 Прототипирование ПО

Конструирование или проектирование программного обеспечения – это непростой процесс, требующий много усилий и внимания. Хорошо спроектированное приложение может в дальнейшем сэкономить много времени и сил его разработчикам. Кроме того проектирование является едва ли не самой крупной частью процесса разработки и в значительной степени влияет на успешность всего процесса разработки. Кроме того процесс проектирования является своего рода центральной частью разработки: до него происходит этап выработки требований к разрабатываемому программному обеспечению и начальной разработки его архитектуры, после него – тестирование и доработка приложения.

Результатом этапа конструирования является готовый исходный код программы. Спецификации требовании и проектная документация могут устареть, но исходный код актуален всегда, и именно поэтому он должен быть максимально качественным. Часто процесс конструирования также называют «кодированием» или «программированием». Конструирование — единственный процесс, который выполняется во всех случаях.

Идеальный программный проект до начала конструирования проходит стадии тщательной выработки требований и проектирования архитектуры. После конструирования в идеале должно быть выполнено исчерпывающее тестирование системы. Однако в реальных проектах разработчики часто пропускают этапы выработки требований и проектирования, начиная прямо с конструирования программы. Тестирование также часто выпадает из расписания из-за огромного числа ошибок и недостатка времени. Отсюда можно сделать вывод, что повышение эффективности конструирования программного обеспечения позволяет оптимизировать любой проект, каким бы несовершенным он ни был. Кроме того качество конструирования ПО - это один из основополагающих факторов оценки качества всего ПО в целом.

В процессе конструирования условно можно выделить следующие этапы:

* детальное проектирование – продумывание архитектуры приложения, отрисовка диаграмм (интерфейсов, классов и т.д.),
* кодирование – написание программного кода в соответствии с результатами, полученными на этапе детального проектирования,
* отладка – этап конструирования, на котором локализуются и устраняются ошибки в коде программы,
* интеграция – процесс объединения отдельных компонентов в единую систему,
* тестирование – процесс выявления ошибок программы с применением различных подходов. Он позволяет получить представление о качестве спроектированного приложения и исправить ошибки, которые не были выявлены и исправлены на этапе отладки.

Если в результате этапа конструирования мы получаем готовый программный продукт, то процесс прототипирования позволяет в результате получить макет готовой системы. Такой макет можно проверить на пригодность предлагаемых для применения концепции, архитектурных и технологических решений, а так же предоставить на ранних этапах разработки заказчику.

Прототипирование можно назвать ускоренной версией проектирования, поскольку в этом случае к основными стадиями можно отнести следующие:

* определение начальных требовании,
* разработка первого варианта прототипа системы на основе требовании,
* изучение прототипа и получение обратной связи о необходимых изменениях и дополнениях,
* переработка и улучшение прототипа (с учётом полученных замечаний и предложений изменяются как спецификации, так и прототип).

Зачастую результат, полученный в процессе прототипирования, может и не стать частью готовой системы. Но, тем не менее, он может служить ещё одним шагом на пути к созданию прототипа финальной версии разрабатываемого продукта.

Выделяют различные подходы к прототипированию. Но, в общем, принято выделять два основных, принципиально различных подхода – это быстрое и эволюционное прототипирование. Эволюционное прототипирование заключается в последовательном создании макетов системы, которые будут все ближе и ближе к реальному продукту. Несомненное преимущество такого подхода в том, что на каждой шаге мы имеем рабочую систему, пусть и не располагающую всеми необходимыми нам функциями, но уже более приближенную к финальной версии, нежели предыдущая версия системы. Такой подход очень удобен в ситуации, когда все требования к системе ещё не определены, и будут определяться в процессе разработки. Однако при использовании быстрого прототипирования заранее предполагается, что создаваемый макет на каком-то этапе будет оставлен и не войдёт в готовую систему. Безусловно, преимуществом такого подхода является скорость – в ответ на требования заказчика сразу проектируется каркас системы. Этот каркас системы отдаётся заказчику, требования вновь уточняются или изменяются и вновь происходит создание каркаса. Стоимость внесения изменений и создания нового каркаса очень низкая, поскольку на этом этапе не нужно писать код системы, а создаётся только её каркас. К преимуществам использования прототипирования как такового можно отнести уменьшение времени разработки и стоимости системы за счёт улучшения спецификации, а также вовлечение пользователей или заказчиков в процесс разработки.

Но у прототипирования так же есть и недостатки:

* недостаточный анализ (акцентирование внимания разработчиков на ограниченном прототипе может отвлечь их от анализа требовании на итоговую систему),
* чрезмерное время на создание прототипа (если разработчики проектируют слишком сложную систему и тратят много времени, то все преимущества от использования прототипирования теряются),
* смешение представлений пользователей или заказчиков о прототипе и готовой системе (есть вероятность, что они могут потерять отличие между прототипом и основой будущей системы и разочароваться в возможностях разработчиков).

На этапе работы над каркасом приложения удобно пользоваться различными вспомогательными средствами, среди которых можно выделить универсальный язык графического описания для объектного моделирования UML. С его помощью можно получить прототип системы в виде различных схем и диаграмм, которые с разных сторон отразят особенности разрабатываемой архитектуры приложения. Полученные диаграммы проверяются только на правильность и соответствие стандартам UML и не гарантируют правильность относительно требований, предъявленных к разрабатываемой системе.

Разрабатываемое приложение должно позволять пользователю составить каркас системы, используя специализированный набор инструментов, затем протестировать разработанный каркас на соответствие предъявленным требованиям, и по результатам тестирования позволить выполнить дополнительные модификации прототипа или же выполнить кодегенерацию на основе созданного прототипа. На данный момент сложно найти программный продукт, позволяющий выполнить этот набор манипуляции.

Можно создавать прототипы:

* архитектуры,
* новой функциональной возможности уже существующей системы,
* структуры илисодержания внешних данных системы,
* инструментальных средств или компонентов,
* рабочих характеристик,
* дизайна пользовательского интерфейса.

## 1.2 Прототипирование как этап ЖЦ ПО

Блаблабла

## 1.3 Роль прототипирования в современных методологиях разработки ПО

Блаблабла

## 1.4 Понятие прототипа архитектуры ПО

Протипы архитектуры создаются, чтобы смоделировать будущую систему в целом. Ни один из отдельных модулей в прототипе не должен быть особенно функциональным. Для создания прототипа не нужно писать программу – он может быть составлен даже на обычном листе бумаги. Главная цель составления такого прототипа – это получить понимание того, как система будет выглядеть в собранном виде, опуская детали.

Готовый прототип архитектуры системы должен дать ответы на многие вопросы:

* Четко ли определены обязанности основных компонентов?
* Являются ли эти обязанности приемлимыми для компонентов?
* Четко ли определена совместная работа основных компонентов?
* Сведено ли к минимуму связывание между компонентами?
* Можно ли выделить потенциальные источники дублиования?
* Можно ли применять определения интерфейсов и ограничения?
* Обладает ли каждый из модулей путем доступа к данным, требуемым ему в ходе выполнения? Может ли он получить его в случае необходимости?

## 1.5 Использование прототипов архитектуры ПО

Бла

## 1.5 Требования к прототипам архитектуры ПО

Бла

## 1.6 Инструменты прототипирования архитектуры ПО

Блабла

## 1.6 Выводы

Согласно приведенным выше рассуждениям, можно сделать вывод о том, что основополагающая проблема эксплуатации современных систем мониторинга заключается в отсутствии на рынке целого класса комбинированных систем, одновременно объединяющих в себе преимущества как распределенных, так и расширяемых систем мониторинга. Кроме того, современные тенденции развития облачных и кластерных решений в области суперкомпьютерных технологий, лишь подтверждают необходимость в появлении подобных инструментов мониторинга.

Таким образом, рассмотренные выше проблемы эксплуатации систем мониторинга, позволяют сделать вывод о неготовности существующих решений комплексно выполнять выдвинутые к ним требования.

Авторами предлагается проект распределенной системы мониторинга и диспетчеризации процессов гетерогенной среды, которая позволяет обеспечить выполнение перечисленных требований. Основополагающая идея предлагаемого в проекте подхода заключается в использовании механизма разработки и исполнения дополнительных модулей в процессе решения задач мониторинга, а также свойств распределенных систем в процессе эксплуатации.