V-model

Основной принцип V-образной модели заключается в том, что детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять. Итерации в проекте производятся по горизонтали, между левой и правой сторонами буквы.

Применительно к разработке [информационных систем](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) V-Model — вариация [каскадной модели](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B2%D0%BE%D0%B4%D0%BE%D0%BF%D0%B0%D0%B4%D0%B0), в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования — вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из фаз разработки влияют на развитие системы тестирования на каждой из фаз тестирования. Модель базируется на том, что приёмо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование — на требованиях и архитектуре, комплексное тестирование — на требованиях, архитектуре и интерфейсах, а компонентное тестирование — на требованиях, архитектуре, интерфейсах и алгоритмах

**Цели**[

V-модель обеспечивает поддержку в планировании и реализации проекта. В ходе проекта ставятся следующие задачи:

* **Минимизация рисков:** V-образная модель делает проект более прозрачным и повышает качество контроля проекта путём стандартизации промежуточных целей и описания соответствующих им результатов и ответственных лиц. Это позволяет выявлять отклонения в проекте и риски на ранних стадиях и улучшает качество управления проектов, уменьшая риски.
* **Повышение и гарантии качества:** V-Model — стандартизованная модель разработки, что позволяет добиться от проекта результатов желаемого качества. Промежуточные результаты могут быть проверены на ранних стадиях. Универсальное документирование облегчает читаемость, понятность и проверяемость.
* **Уменьшение общей стоимости проекта:**
* **Повышение качества коммуникации между участниками проекта:**

**Достоинства**

* Пользователи V-Model участвуют в разработке и поддержке V-модели. Комитет по контролю за изменениями поддерживает проект и собирается раз в год для обработки всех полученных запросов на внесение изменений в V-Model[[6]](https://ru.wikipedia.org/wiki/V-Model" \l "cite_note-Further_development_of_the_V-Model-6).
* На старте любого проекта V-образная модель может быть адаптирована под этот проект, так как эта модель не зависит от типов организаций и проектов[[7]](https://ru.wikipedia.org/wiki/V-Model#cite_note-V-Model_tailoring-7).
* V-model позволяет разбить деятельность на отдельные шаги, каждый из которых будет включать в себя необходимые для него действия, инструкции к ним, рекомендации и подробное объяснение деятельности[[8]](https://ru.wikipedia.org/wiki/V-Model#cite_note-activity-8).

**Ограничения**

Следующие моменты не учитываются в V-модели, но могут быть рассмотрены отдельно, либо возможно адаптировать модель под них:

* Не регулируется размещение контрактов на обслуживание.
* Организация и выполнение управления, обслуживания, ремонта и утилизации системы не учитываются в V-модели. Однако, планирование и подготовка к этим операциям моделью рассматриваются.
* V-образная модель больше касается разработки программного обеспечения в проекте, чем всей организации процесса[[9]](https://ru.wikipedia.org/wiki/V-Model#cite_note-limits-9).

Iterative model

* Основная идея – **пошаговая разработка ПО (постепенная, итеративная, инкрементальная)**.

**Когда оптимально использовать итеративную модель?**

* Требования к конечной системе заранее четко определены и понятны.
* Проект большой или очень большой.
* Основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.

[Итеративный](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F) подход ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *iteration* - «повторение») в [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) — это выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл [PDCA](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_%D0%94%D0%B5%D0%BC%D0%B8%D0%BD%D0%B3%D0%B0): *Планирование — Реализация — Проверка — Оценка*([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *plan-do-check-act cycle*).

Преимущества итеративного подхода:

* снижение воздействия серьёзных [рисков](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%81%D0%BA) на ранних стадиях проекта, что ведет к минимизации затрат на их устранение;
* организация эффективной обратной связи проектной команды с потребителем (а также заказчиками, [стейкхолдерами](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B9%D0%BA%D1%85%D0%BE%D0%BB%D0%B4%D0%B5%D1%80" \o "Стейкхолдер)) и создание продукта, реально отвечающего его потребностям;
* акцент усилий на наиболее важные и критичные направления проекта;
* непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом;
* раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта;
* более равномерная загрузка участников проекта;
* эффективное использование накопленного [опыта](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%8B%D1%82%D0%BD%D0%BE%D0%B5_%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B5);
* реальная оценка текущего состояния проекта и, как следствие, большая уверенность заказчиков и непосредственных участников в его успешном завершении.
* затраты распределяются по всему проекту, а не группируются в его конце[[1]](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0#cite_note-1).

Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, мы стремимся к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна.

Спиральная модель

«Спиральная модель» похожа на инкрементную, но с акцентом на анализ рисков. Она хорошо работает для решения критически важных бизнес-задач, когда неудача несовместима с деятельностью компании, в условиях выпуска новых продуктовых линеек, при необходимости научных исследований и практической апробации.  
  
**Спиральная модель предполагает 4 этапа для каждого витка:**

1. планирование;
2. анализ рисков;
3. конструирование;
4. оценка результата и при удовлетворительном качестве переход к новому витку.

Эта модель не подойдет для малых проектов, она резонна для сложных и дорогих, например, таких, как разработка системы документооборота для банка, когда каждый следующий шаг требует большего анализа для оценки последствий, чем программирование. На проекте по [разработке СЭД для ОДУ Сибири СО ЕЭС](http://edsd.biz/odu-sibiri) два совещания об изменении кодификации разделов электронного архива занимают в 10 раз больше времени, чем объединение двух папок программистом. Государственные проекты, в которых мы участвовали, начинались с подготовки экспертным сообществом дорогостоящей концепции, которая отнюдь не всегда бесполезна, поскольку окупается в масштабах страны

Prototype

Основная идея – вместо замораживания требований, создавать, перед началом проектирования и кодирования, **одноразовый прототип**, который может помочь понять требования заказчика.

Прототип - сильно упрощенный вариант разрабатываемого ПО. Он разрабатывается на основе текущих известных требований.

Очевидно, что разработка прототипа тоже проходит этапы проектирования, кодирования и тестирования, но каждый из этих этапов выполняется не формально и не тщательно.

Прототипирование является привлекательной идеей для сложных и больших систем, для которых нет ручного процесса или существующей системы, которые могут помочь выявить требования.

В таких ситуациях, предоставление заказчику возможности «поиграть» с прототипом позволяет выявить неоценимые и неосязаемые результаты, которые могут помочь определить требования к системе.

Ситуации, в которых требуется прототипирование:

для ***демонстрации осуществимости некоторого нового подхода***;

при ***разработке новых систем***, когда не понятно, какие ограничения могут возникнуть, или какие алгоритмы могут быть разработаны для реализации требований.

В обоих ситуациях риск, связанный с данными проектами м.б. уменьшен, за счет выполнения прототипирования.

**Достоинства прототипирования**

1. Опыт полученный при разработке прототипа уменьшает стоимость разработки конечного ПО.
2. Получаются более стабильные требования, благодаря обратной связи с пользователями в требованиях будет меньше изменений.

Вероятно, что качество конечного ПО будет намного лучше, в связи с тем, что опыт полученный разработчики в ходе создания прототипа позволит улучшить проектирование, составление кода и выполнения тестирования

Недостатки

 Тяжелое начало.

 Возможны высокая стоимость и сжатые сроки.

 Поощряет раздувание требований.

 Затрудняет поздние изменения требований.

RUP

Унифицированный процесс - это вариант итеративного процесса, разработанный компанией Rational Software, (сейчас подразделение IBM).

Он является:

* управляемым рисками и прецедентами (требованиями);
* архитектуро-центричным;
* итеративным и инкрементным.

Это сложившийся открытый процесс разработки ПО от авторов UML.

Унифицированный процесс компании Rational (RUP) – это коммерческое расширение UP.

**Схема модели RUP**

Программное обеспечение разрабатывается в течении 4 этапов (фаз):

* 1. фаза анализа и планирования требований (начало) (inception)
  2. фаза проектирования (развития, уточнения) (elaboration)
  3. фаза построения (разработки) (construction)
  4. фаза внедрения (перехода)

**Фазы унифицированного процесса разработки**

У каждой фазы есть

1. цель,
2. основная деятельность с акцентом на одном или более рабочих потоках, и
3. контрольная точка.

**Цели и результаты фаз разработки**

1. **Начало** – проект сдвигается с «мертвой точки». Результат: определяются цели жизненного цикла;
2. **Проектирование** (уточнение) – развитие архитектуры системы. Результат: определяется архитектура программной системы;
3. **Построение** – построение программного обеспечения. Результат: разрабатывается базовая функциональность;
4. **Внедрение** – развертывание программного обеспечения в пользовательской среде. Результат: выполняется выпуск продукта и его развертывание.

В каждой фазе выполняется пять основных рабочих потоков:

* 1. **определение требований** – выяснение того, что должна делать система;
  2. **анализ** – конкретизация и структурирование требований;
  3. **проектирование** – реализация требований в архитектуре системы (как система это делает);
  4. **реализация** – построение программного обеспечения;
  5. **тестирование** – проверяется, работает ли должным образом реализация.

**Ключевое отличие RUP от других моделей** заключается в том, что она отделяет фазы разработки от задач и позволяет множество таких подпроцессов выполняться в рамках фазы.

1. В модели водопада (или итеративной модели, основанной на водопаде), этап процесса был связан с конкретной задачей, выполняемой некоторым процессом, например выявление требований, проектирование и т.п.
2. В RUP такие задачи отделены от фаз (этапов), что позволяет, например, в ходе этапа построения выполнять задачу выявления требований.
3. Т.е. это позволяет некоторые работы по выявлению требований выполнять даже при построении (создании) ПО, что не разрешено в модели водопада.

|  |  |  |
| --- | --- | --- |
| **Достоинства** | **Недостатки** | **Подходящие типы проектов** |
| Все преимущества итеративной модели.  Предоставляет гибкий каркас для выполнения большого разнообразия проектов. | Для каждого проекта должен быть спроектирован его процесс. | Может быть применен к широкому набору проектов, т.к. это позволяет его гибкость. |