1. Понятие человеко-машинного взаимодействия

**Человеко-компьютерное взаимодействие** обеспечивает связь между пользователем и компьютером. Он позволяет достигать поставленных целей, успешно находить решение поставленной задачи. Основной задачей является улучшение взаимодействия между человеком и компьютером, делая компьютеры более удобными (usability) и восприимчивыми к потребностям пользователей.

1. История развития человеко-машинного взаимодействия в России

Так же проблематика производимых человеком операций за компьютером была естественным продолжением классических целей инженерной психологии, за исключением того, что новые проблемы имели существенный когнитивный, коммуникационный и интерактивный характер, ранее не рассматриваемый в инженерной психологии, и способствовали продвижению, таким образом, инженерной психологии в этом направлении.

Эргономические исследования также подчёркивали связь условий работы с явлениями, вызывающими напряжение (стресс), такими, как: рутинная работа, сидячее положение, зрительное восприятие визуальных образов на дисплеях и многими другими, до этого не рассматриваемыми как взаимосвязанные.

В СССР институционализация этого научного направления началась с 1958 года с обзоров американских работ в журнале «Вопросы психологии».

В 1930-е гг. в СССР сформировалась новая дисциплина «психотехника», направленная на анализ и улучшение трудовой деятельности. Интенсивные исследования предпринимались в области авиационной эргономики, где изучались кабины пилотов, уделялось внимание рукояткам, приборным панелям, креслу пилота, рабочему месту штурмана. Многие крупные заводы в это время создавали у себя психофизиологические лаборатории, проводившие исследования с целью улучшения работы людей со станками и конвейерами.

1. История развития человеко-машинного взаимодействия за рубежом

Началом эргономической фазы человеко-компьютерного взаимодействия можно считать диссертацию [Айвена Сазерленда](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B7%D0%B5%D1%80%D0%BB%D0%B5%D0%BD%D0%B4,_%D0%90%D0%B9%D0%B2%D0%B5%D0%BD" \o "Сазерленд, Айвен) (Sutherland, 1963), которая определила развитие компьютерной графики как науки. При этом компьютерная графика нуждалась в эргономических проектах с целью эффективного управления сложными моделями [CAD](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)/[CAM](https://ru.wikipedia.org/wiki/CAM) систем. Исследования в этой области были продолжены в работах *Man-machine symbiosis* ([Licklider](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BA%D0%BB%D0%B0%D0%B9%D0%B4%D0%B5%D1%80,_%D0%94%D0%B6%D0%BE%D0%B7%D0%B5%D1%84_%D0%9A%D0%B0%D1%80%D0%BB_%D0%A0%D0%BE%D0%B1%D0%BD%D0%B5%D1%82%D1%82" \o "Ликлайдер, Джозеф Карл Робнетт), 1960), *Augmentation of human intellect* ([Engelbart](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D0%B3%D0%B5%D0%BB%D1%8C%D0%B1%D0%B0%D1%80%D1%82,_%D0%94%D1%83%D0%B3%D0%BB%D0%B0%D1%81" \o "Энгельбарт, Дуглас), 1963) и *[Dynabook](https://ru.wikipedia.org/wiki/Dynabook" \o "Dynabook)* ([Kay](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D0%B9,_%D0%90%D0%BB%D0%B0%D0%BD_%D0%9A%D1%91%D1%80%D1%82%D0%B8%D1%81" \o "Кэй, Алан Кёртис), [Goldberg](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BB%D0%B4%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%B4%D0%B5%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82)" \o "Голдберг, Адель (программист)), 1977). В результате научных исследований получили развитие те инструменты, без которых трудно представить сейчас работу с компьютером: «[мышь](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BC%D1%8B%D1%88%D1%8C)», [поэлементно-адресуемое (bitmap) отображение](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0), «[окно](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)», [метафора рабочего стола](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D1%84%D0%BE%D1%80%D0%B0_%D1%80%D0%B0%D0%B1%D0%BE%D1%87%D0%B5%D0%B3%D0%BE_%D1%81%D1%82%D0%BE%D0%BB%D0%B0), point-and-click-редакторы.

1. Способы оценивания интерфейсов. Стандарты.

Американский Национальный институт стандартов **(ANSI)** имеет по данному направлению специальную консультативную группу — Комитет по стандартам интерфейса «человек-компьютер» **(The Human-Compute Interface Standard Committee).**

в 1987 г., когда корпорация IBM объявила о намерении создать единую среду разработки приложений (**Systems Application Architecture — SAA**).

Данный проект предусматривает не только разработку единых принципов создания приложений, но и «материализацию» этих принципов на основе соответствующей технологической базы.

**Целями проекта являются:**

* Повышение производительности труда программистов и конечных пользователей;
* Облегчение эксплуатации и сопровождения программного обеспечения;
* Повышение эффективности распределённой обработки информации
* Увеличение отдачи инвестиций в разработку информационных систем

**Проект SAA содержит 4 компонента**:

1. Соглашения по интерфейсу пользователя (Common User Access)
2. Соглашения по программному интерфейсу (Common Programming Interface)
3. Соглашение по разработке приложений (Common Application - CA)
4. Соглашение по коммуникациям (Common Communication Support)
5. Способы оценивания интерфейсов. Руководящие принципы.

* Анкетирование пользователей
* Эксперименты на живых пользователях
* Эксперт
* Определение среднего времени необходимого пользователю по методике GOMS, KLM
* Энтропия RGB-профиля

Оценивается визуальная сложность системы. Очевидно, что программа с десятью панельками и двадцатью кнопочками, сложнее, чем стартовая страница гугл. Естественно, способ очень относительный, но простой и быстрый.

* Информационная производительность

Отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Например, если выводится информационное модальное окно с единственной кнопкой «ОК», то вводимая пользователем информация (клик по кнопке) абсолютно бесполезна, что уменьшает показатель информационной производительности.

* Анализ XML-дерева
* Количество классов, на которые можно разбить объекты интерфейса

1. Проектирование интерфейсов. Рекомендации.

• Отсутствие модальности

Поскольку модальность приводит к ошибкам взаимодействия с программой, хорошим принципом создания интерфейсов является исключение модальности.

• Сохранность пользовательских данных

Назначением большинства программ в конечном итоге является упрощение работы, выполняемой человеком. Пользовательские данные — это результат работы человека. Если этот результат будет утерян, работу придется выполнять снова, что нельзя назвать упрощением. Поэтому любая программа должна обеспечивать сохранность данных пользователя, будь то простое текстовое сообщение, 3D-модель или научная статья. О важности данных может судить только сам пользователь, но никак не программа, с которой он взаимодействует. Путей для обеспечения сохранности данных может быть много: это и автоматическое сохранение любых изменений, и обратимость операций, и создание резервного архива (backup).

• Формирование команд по принципу «объект -> действие

1. Сначала указать объект, а затем действие, которое необходимо совершить с этим объектом (модель «объект-действие»).

2. Сначала указать действие, а затем объект, к которому следует применить это действие (модель «действие-объект»).

Предпочтительной является первая модель — «объект-действие». Во-первых, она исключает многие модальные ошибки, поскольку переключение действия обычно сопровождается переключением режимов работы. Во-вторых, она проще для восприятия человеком, поскольку, как правило, локус внимания пользователя уже находится на объекте, когда возникает необходимость выполнить какое-то действие с этим объектом.

Использование второй модели — «действие-объект» — тоже допустимо, но только в тех случаях, когда ее применение достаточно аргументировано.

• Монотонность

Привычки могут формироваться у человека лишь в том случае, если каждое действие можно выполнить всегда одним и тем же образом. Интерфейс можно назвать монотонным, когда каждое элементарное действие в нем можно выполнить ровно одним способом. Иначе есть риск создать интерфейс, который не будет устраивать ни ту, ни другую группу пользователей.

• Видимость

1. текущем состоянии системы и смене состояния в результате действий пользователя;

2. способах управления и воздействия на систему.

• Состоятельность

Взаимодействие с подобными по виду элементами и объектами должно происходить всегда единообразно, иначе будут возникать ошибки, связанные с модальностью интерфейса. Примером может являться интерфейс, в котором в одной ситуации элемент, выглядящий как кнопка, необходимо нажать, а в другой ситуации элемент, выглядящий подобным образом, необходимо переместить.

1. Виды интерфейсов (GUI, Web-интерфейс)

WEB-приложений, к которым относятся WEB-сайты, Интернет-магазины, поисковые порталы, корпора­тивные автоматизированные системы и т.д. От способов построения интерфейса также зависит правильность понимания отдаваемых системе команд, что позволяет не навредить системе или используемым ею компонентам и работать эффективно.

1. **Программные средства реализации взаимодействия**

Системное ПО (System Software) – это программы и комплексы программ, являющиеся общими для всех, кто совместно использует технические средства компьютера, и применяемые как для автоматизации разработки (создания) новых программ, так и для организации выполнения программ существующих.

Системное ПО делится на пять групп:

1. Операционные системы (ОС).

2. Системы управления файлами.

3. Интерфейсные оболочки для взаимодействия пользователя с ОС и программные среды.

4. Системы программирования.

5. Утилиты.

Под операционной системой (ОС) понимают комплекс управляющих и обрабатывающих программ, который, с одной стороны, выступает как интерфейс (интерфейс – это комплекс спецификаций, определяющих конкретный способ взаимодействия пользователя с компьютером) между аппаратурой компьютера и пользователем с его задачами, а с другой – предназначен для наиболее эффективного использования ресурсов вычислительной системы и организации надежных вычислений.

Назначение системы управления файлами – организация более удобного доступа к данным, организованным как файлы. Именно благодаря системе управления файлами вместо низкоуровневого доступа к данным с указанием конкретных физических адресов нужной нам записи используется логический доступ с указанием имени файла и записи в нем.

Для удобства взаимодействия с ОС могут использоваться дополнительные интерфейсные оболочки. Их основное назначение - либо расширить возможности по управлению ОС, либо изменить встроенные в систему возможности.

Система программирования состоит из следующих компонент: транслятор с соответствующего языка, библиотеки подпрограмм, редакторы, компоновщики и отладчики.

Из-за огромного разнообразия прикладного ПО существует множество вариантов его классификации. Рассмотрим наиболее общую классификацию прикладных программ. Разделим данное ПО на 2 больших класса:

1. ПС общего назначения. К таким относятся программы, обеспечивающие выполнение наиболее часто используемых, универсальных задач (текстовые редакторы, табличные процессоры, графические редакторы, СУБД и т.д.).

2. ПС профессионального уровня. Программы этого класса ориентируются на достаточно узкую предметную область, но проникают в нее достаточно глубоко (издательские системы, САПР - системы автоматизированного проектирования, программы 3D-графики, программы видеомонтажа, нотные редакторы, АСУ - автоматизированные системы управления и т.д.).

Каждый класс разбивается на многочисленные подклассы.

1. Офисное ПО

Помимо рассмотренных программных средств, возникает много трудно классифицируемого по способам применения ПО. Многие из них роднит среда их применения - в делопроизводстве, в различных аспектах управления мелкими и средними предприятиями. Эти программы можно назвать офисными. Сюда можно отнести следующие виды программ:

2. Средства обработки текстовой информации.

3. Средства табличной обработки информации

4. Системы управления базами данных

Средства разработки презентаций и публикаций

o Автоматизация ввода информации в компьютер

o Автоматизация перевода документов

1. **Основные признаки хорошего интерфейса**

1 Один столбец вместо нескольких

Один столбец точнее отражает то, что вы хотите донести. Пользователи проходят сверху вниз по более предсказуемому пути. В дизайне с несколькими колонками есть риск отвлечения пользователя от основной задачи страницы.

2 Подарите что-либо вместо того, чтобы сразу пытаться продать

3 Объединяйте схожую функциональность, а не фрагментировать UI

Со временем легко создать несколько разных секций, элементов и функций, выполняющих одно и то же. Отслеживайте дублирующуюся функциональность, так как она загружает пользователей. Чем больше таких дублей, тем сложнее пользователю пользоваться вашей услугой.

4 Предоставьте доказательства от других людей, а не только рассказ о себе

Ещё одна отличная тактика убеждения, которая напрямую влияет на конверсию. Одобрение вашего предложения другими людьми побуждает к действиям. Попробуйте показать отзывы, или статистику использования сервиса.

5 Отмена действий вместо подтверждения

Возможность отменить действие выглядит как уважение к интеллекту пользователя и позволяет действиям проходить без запинок. Постоянные подтверждения выглядят, будто вы считаете, что пользователь не знает, чего он хочет. Могу предположить, что большинство действий совершается людьми намеренно, и лишь малая их часть происходит случайно. Особенно неприятно подтверждения выглядят в повторяющихся действиях. Пользователи почувствуют, что они лучше контролируют ситуацию, если они смогут отменить их действия, и если их не будут на каждом шагу спрашивать подтверждения.

6 Вместо общего описания – конкретика о происхождении вашего сервиса

Раскрытие данных о происхождении сервиса одновременно говорит о нём больше и переводит общение на более интимный уровень. Упомяните страну, штат, город – это естественный способ для человека представиться. Если вы делаете то же самое, вы выглядите более дружественным. Часто такое упоминание повышает качество продукта в глазах людей.

7 Сразу показывайте поля ввода, а не прячьте их на отдельной странице

8 Последовательный интерфейс, не требующий постоянного обучения

Постоянный интерфейс означает, что пользователю не нужно постоянно учиться им пользоваться и тратить на это силы. Нажимая на кнопки и передвигая слайдеры, мы учимся, как должны выглядеть и работать. Постоянство помогает облегчать работу, а как только мы лишаемся постоянства, мы вынуждены опять обучаться. Постоянства можно добиваться через цвета, направления, поведение, расположение, размер, форму, метки и язык. Необычные элементы важны тогда, когда нам нужно что-то выделить или привлечь внимание.

1. **Связь человеко-машинного интерфейса с другими науками (эргономика, инженерная психология)**

Научная дисциплина, комплексно изучающая человека (группу людей) в конкретных условиях его (их) деятельности в современном производстве. Эргономика возникла в связи со значительным усложнением технических средств и условий их функционирования в современном производстве, существенным изменением трудовой деятельности человека, синтезированием в нем многих трудовых функций

Ряд эргономических проблем связан с зада­чами производства технически сложных то­варов широкого потребления, а также с проектированием рабочих мест и ус­ловий трудовой деятельности для лиц с пониженной трудоспособностью. Э. не только изучает, но и проектирует це­лесообразные варианты конкретных ви­дов человеческой деятельности, связан­ных с использованием новой техники.

Как самостоятельная научная дисциплина инже­нерная психология начала формироваться в 40-х годах прошлого века.

Однако идеи о необходимости комплекс­ного изучения человека и технических устройств выс­казывались русскими учеными еще в девятнадцатом столетии.

Так, великий русский ученый Д.*И. Менделе­ев* уже в 1880 г. говорил о необходимости при констру­ировании воздухоплавательных аппаратов думать не только о двигателях, но и о человеке и пользоваться данными различных наук. Только тогда будет создан аппарат, «доступный для всех и уютный», —подчерки­вал ученый.

В 1882 г. русским метеорологом *М.А. Рыхачевым* был поставлен вопрос о психологической пригодности к летному делу. Он разработал перечень качеств, необ­ходимых воздухоплавателю для управления летатель­ным аппаратом: быстрота соображения, распоряди­тельность, осмотрительность, внимательность, ловкость, сохранение присутствия духа. Эти положения впоследствии частично использовались при отборе пилотов русской авиации, причем гораздо раньше, чем в других странах. Особенно большое значение профотбор имел для пилотов тяжелых многомоторных само­летов, которые впервые в мире появились в России. Русские авиаторы поставили вопрос о природе летных способностей и наметили возможные пути их опреде­ления и использования в летной практике.

В 20-х годах на многих крупных предприятиях промышленности и транспорта создаются лаборатории психотехники. Основное внимание в них уделялось работам по профессиональному отбору. Был проведен также ряд работ, явившихся прообразом современных инженерно-психологических разработок.

К их числу можно отнести работу по выбору наиболее рациональ­ного расположения букв на клавиатуре, пишущей ма­шинки с учетом времени двигательной реакции, рабо­ты по рационализации шкал авиационных приборов и кабины самолета, работы по организации рабочего места вагоновожатого и т. п. Эти работы проводились под руководством *С.Г. Геллерштейна, И.Н. Шпильрей-на, Н.В. Зимкина* и многих других ученых.

1. Направления стандартизации информационных технологий

**Направления 1-го приоритета**

* *Языки программирования и программный интерфейс.* Участие России в разработке международных стандартов по данной тематике позволяет поддерживать те направления российской математической школы, которые имеют традиционно устойчивую позицию, а также разрабатывать новые языки для перспективного направления программирования.
* *Языки описания документов.* Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую создание, хранение и обращение документов в открытых системах.
* *Программная инженерия.* В сочетании с сертификацией и внедрением систем качества, соответствующих требованиям международных стандартов, участие в разработке и применении этой группы стандартов дает отечественным разработчикам, а также поставщикам и производителям программных средств возможность повысить качество и конкурентоспособность своей продукции как на внутреннем, так и на внешнем рынке.
* *Сервисы управления данными.* Данное направление является перспективным в плане соз0дания и развития отечественных систем распределенных баз данных и формирования национальных информационных ресурсов федерального, регионального и местного уровней в структуре Единого информационного пространства России.
* *Работа в сетях и соответствующие соединения.* Работы в данном направлении позволят стандартизировать функции, необходимые для установления и управ лепил информационным обменом через сети и физические интерфейсы.
* *Безопасность информационных технологий.* Работы в области безопасности информационных технологий позволяют создать комплект стандартов, поддерживающих методы и средства обеспечения безопасности, и первую очередь на уровне личности и различных общественных групп. Данное направление является одним из важнейших с учетом бурного роста информационного обмена между компьютерами всех уровней.

**Направления 2-го приоритета**

* *Сбор данных и системы идентификации.* Работы в данной области позволяют создать комплект стандартов, поддерживающих разработку идентификационных карт и соответствующих устройств для использования в межотраслевых приложениях и в международном обмене.
* *Мультимедиа и представление информации.* Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую кодированное представление, обработку, обмен аудио, изображениями, мультимедиа- и гипермедиа информацией для разнообразных приложений.
* *Пользовательский интерфейс.* Работы в данной области позволяют создать комплект стандартов, поддерживающих пользовательский интерфейс для интерактивной деятельности в локальных и распределенных средах с использованием аудио, изображений, мультимедиа и гипермедиа информации, включая специальные интерфейсы для людей, имеющих физические недостатки или работающих в специфических условиях.
* *Офисное оборудование.* Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую адекватный уровень требований к эксплуатационным характеристикам и методам тестирования офисного оборудования.
* *Кодированные наборы символов.* Стандартизация в данной области позволяет обеспечить необходимую нормативную базу, поддерживающую множества графических символов и их кодированное представления для одно- и многоязыковых функций при работе с информацией (интернационализация).

**Направления 3-го приоритета**

* *Среды для информационного обмена.* Работы в данной области включают стандарты, поддерживающие требования к оптическим и магнитным носителям данных и соответствующим устройствам на их основе, обеспечивающим хранение и обмен данными в системах обработки информации.
* *Геоинформационные технологии.* Предусматривают развитие системы стандартов, направленных на повышение качества электронных карт и соответствие их требованиям международных стандартов, на сокращение трудоемкости и сроков изготовления электронных карт для создания предпосылок совместимости различных геоинформационных систем и в перспективе создания национальной базы геоинформационных данных.
* *Информационные технологии в охране здоровья.*

1. **Современное состояние стандартизации в мире**

К настоящему времени на международном уровне сформировалась мощная кооперация организаций, разрабатывающих стандарты в области ИТ. Среди этих организаций в первую очередь следует назвать Международную организацию по стандартизации — **ИСО** (International Organization for Standardization — ISO), Международную электротехническую комиссию — **МЭК** (International Electrotechnical Commission — IEC) и Международный союз электросвязи — **МСЭ** (International Telecommunication Union — ITU); его сектор по телекоммуникациям **МСЭ-Т** является с 1993 года правопреемником МККТТ.

В 1987 году ИСО и МЭК объединили свою деятельность по стандартизации в области информатизации, создав Совместный технический комитет № 1 — **СТК1 ИСО/МЭК «Информационные технологии» (Joint Technical Committee N1 — ISO/IEC JTC1 Information Technology**), основной задачей которого является разработка базовых стандартов информационных технологий вне зависимости от их конкретных приложений.

1. Usability

По международному стандарту **ISO 9241-11** “Руководство по юзабилити”, **юзабилити**— это эффект от использования технологии определенными пользователями в определенных условиях при достижении ими определенных целей. Эффект этот измеряется в эффективности (качество результата), продуктивности (затраты для достижения результата) и удовлетворенности (субъективное отношение к процессу работы и продукту).

В процессе развития компьютерных технологий выделяются три волны:

* первая, “аппаратная” волна пришлась на начало 50-х годов XX века, когда компьютеры стали использоваться для промышленных целей;
* вторая, “программная” волна совпала с широким распространением ПК в 1980-х годах, когда компьютерные технологии появились на многих рабочих местах и в домах;
* третья волна “человеческих факторов“ возникла в середине 1990-х, когда компьютерные технологии стали широко использоваться в разнообразных устройствах.

При определении этапов в теории трех волн рассматривались как технические изменения, так и их влияние на человека и общество. Поэтому периоды не совпадают с техническими этапами появления новых поколений компьютеров

1. Человеко-машинное взаимодействие – исследования, перспективы, будущее

Обмен информацией между человеком и компьютером можно определить как узел взаимодействия. Узел взаимодействия включает в себя несколько аспектов:

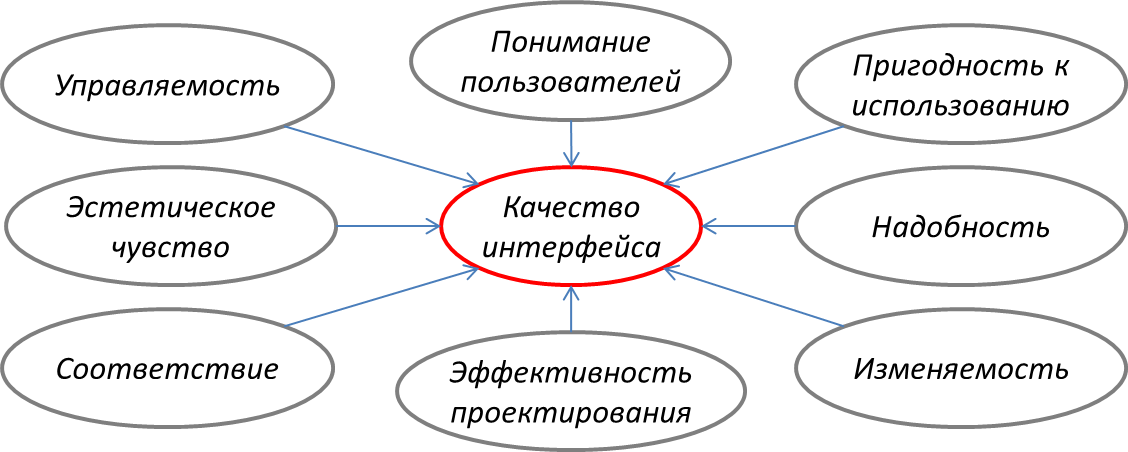
* Область задач: условия и цели, ориентированные на пользователя.
* Область машины: среда, с которой взаимодействует компьютер, то есть ноутбук студента в комнате в общежитии колледжа.
* Области интерфейса: непересекающиеся области, касающиеся процессов человека и компьютера, не относящиеся к сфере взаимодействия.
* Входящий поток: поток информации, который начинается в области задач, когда пользователь имеет несколько задач, которые требуют использования компьютера.
* Выходной поток: поток информации, который возникает в машине.
* Обратная связь: узлы взаимодействия, проходящие через интерфейс, оцениваются, модерируются и подтверждаются, так как они проходят от человека через интерфейс к компьютеру и обратно.

1. **Принципы Usage Centered Design**

Ларри Константин, идеолог концепции дизайна, ориентированного на использование (Usage Centered Design, не путать с User-Centered design), в книге «Software For Use», написанной им в 1999 г. совместно с Люси Локвуд, представил следующие принципы разработки интерактивных систем:

* **Структурный принцип:** Проектирование интерфейса должно вестись целенаправленно, с использованием конструктивных решений, основанных на четких и последовательных моделях, узнаваемых для пользователя. Структура интерфейса может формироваться путем группировки связанных объектов и разделения несвязанных, подчеркиванием различий между разнородными элементами и наделение похожими чертами родственных объектов.
* **Принцип простоты:** Дизайн должен быть простым, общие задачи должны быть понятны, общение между программой и человеком должно происходить на родном для него языке.
* **Принцип видимости:** Все необходимые для решения конкретной задачи элементы интерфейса должны быть видимы и не должны отвлекать пользователя посторонней или избыточной информацией.
* **Принцип обратной связи:** Дизайн должен информировать пользователей о выполняемых действиях, изменениях состояния или условий, об ошибках или исключениях. Эта информация должна быть актуальна и интересна пользователю и представлена в четкой, компактной и недвусмысленной форме.
* **Принцип толерантности:** Дизайн должен быть гибким и терпимым к действиям пользователей, позволять отмену и повторное выполнение операций, а также предотвращать ошибки (где это возможно), интерпретируя все входные последовательности в разумные действия.
* **Принцип повторного использования:** Интерфейс должен использовать согласованные внутренние и внешние компоненты, тем самым уменьшая для пользователей необходимость переосмысления или запоминания их (компонентов) назначения и поведения.

1. **Критерии эффективного интерфейса**



1. **Модели пользовательского интерфейса. (Программиста, пользователя, разработчика)**

***Пользовательская модель интерфейса***– совокупность обобщенных представлений конкретного пользователя или группы пользователей о процессах, происходящих во время работы программ­ной системы.

Модель базируется на особенностях опыта конкретных пользователей, который характеризуется уровнем подготовки в предметной области разрабатываемого ПО, *интуитивными моделями* выполнения операций в этой об­ласти, уровнем подготовки в области владения компьютером, а также устоявшимися стереотипами работы с компьютером.

Для ее построения необходимо изучить особенности опыта предполагаемых пользователей, для чего используют опросы, тесты и фиксируют на пленку последовательность вы­полнения операций в реальном процессе.

**Модель программиста** – самая легкая для отображе­ния, так как она может быть формально и недвусмыслен­но описана. На самом деле данная модель это представ­ленная в определенном виде функциональная специфи­кация ПП.

Объекты и данные, составляющие программу, интересны программисту, но не обязательно в плане того, как пользователь взаимодействует с информацией.

Например, с точки зрения программиста интерфейсы предназначены для сохранения и восстановления информации, представ­ляют собой поля данных или записи в базе данных. Точка зрения на них у пользователя может быть иной, чем у раз­работчиков. Одни и те же данные могут быть входом в про­грамму для проверки, личную записную книжку или де­ловую телефонную книгу.

Программист при разработке интерфейса исходит из того, управление какими операциями нужно реализовать в нем, и как это осуществить, не затрачивая существен­ных ресурсов компьютера, своих сил и времени. Его интересуют эффективность, функци­ональность, технологичность, внутренняя стройность и дру­гие не связанные с удобством пользователя характеристики ПО. Именно поэтому большинство интерфейсов существующих программ вызывают серьезные нарекания пользователей.

Работа **проектировщика** подобна работе архитектора. Создание программного продукта во многом похоже на возведение дома.

Проектировщик (архитектор) узнает идеи, пожелания пользователя (владельца дома), соединяет их со своими на­выками и материалами, необходимыми для программиста (строителя), и проектирует программное обеспечение (дом), которое должно удовлетворять нужды пользователя. Модель проектировщика представляет собой нечто сред­нее между моделью пользователя и моделью программиста.

Разработчики обычно не входят в контакт с пользователя­ми, для которых создают программы. Недостающим звеном между пользовательским окружением и программистским миром являются проектировщик пользовательского ин­терфейса и другие члены команды по разработке.

1. Проектирование человеко-машинного взаимодействия. Ментальная модель

***Ментальная модель*** – это внутреннее сознательное представление человека о том, как работают вещи. Это широкая концепция причины действий и их результатов. Люди применяют эти модели к новым ситуациям и таким образом, нет необходимости изучать все с пустого места. Это помогает им разбираться в новом намного эффективнее.

***Ментальная модель***  — это представление человека о процессе взаимодействия с объектом.

***Ментальная модель*** — это интуитивное понимание принципов работы объекта или системы, основанное на прошлом опыте человека, имеющейся информации и здравом смысле.

***Концептуальная модель*** — это действующая модель, которую человек получает, знакомясь с интерфейсом конкретного продукта.

1. Проектирование человеко-машинного взаимодействия. Метафоры

Когда мы переносим знания об окружающем мире в мир компьютеров, начинает действовать концепция **метафор**.

***Определение. Метафора*** — это «понятие, переносящее свойства или признаки одного объекта на другой для выяснения их сходства или аналогии».

***Определение. Метафоры*** помогают пользователям освоить новые для себя области деятельности (например, работу с текстовым процессором), осмысляя их в терминах области, которая им уже знакома и понятна (например/ пишущая машинка). **Метафоры** помогают проектировщикам, так как использование метафор позволяет им структурировать элементы интерфейса по аналогии с известной пользователям областью.

1. Проектирование человеко-машинного взаимодействия. Микровзаимодействия.

Микро-UX — стратегия использования простых, небольших элементов в дизайне продукта, ориентированных на решение одной задачи. Преимущественно такие эффекты и интерактивные элементы предназначены для формирования интересного и уникального пользовательского опыта.

1. Речевые интерфейсы.

С ростом числа решаемых задач и мощностью современных компьютеров все более очевидными становятся недостатки средств взаимодействия человека с компьютером на основе командного и графического интерфейсов.

На смену им претендовал речевой, а теперь уже многомодальный интерфейс, который параллельно обрабатывает два или более естественных для человека потока информации, таких как речь, рукописный текст, жесты, движение головы и тела.

1. Многомодальные интерфейсы.

Для решения глобальной проблемы человеко-машинного взаимодействия необходимо использовать дополнительные виды каналов передачи информации (речь, артикуляция губ, жесты, направление взгляда и т.д.).

Такой способ взаимодействия получил название **«многомодальное взаимодействие»**, которое реализуется путем многомодальных интерфейсов

Особенности **«многомодальных интерфейсов»:**

* + Такие интерфейсы свойственны межчеловеческому общению.
  + Здесь мы сами выбираем, какой канал, для передачи какого типа информации нам наиболее удобно использовать в данный момент.
  + Позволяют обеспечить наиболее эффективное и естественное для человека взаимодействие с различными автоматизированными средствами управления и коммуникации.
  + В многомодальных системах информация от аудио, видео, тактильных и других коммуникативных каналов непрерывно обрабатывается, создавая реальное или виртуальное окружение, позволяющее удовлетворить желания пользователя, и оперативно адаптироваться к контексту.

Многомодальное человеко-машинное взаимодействие опирается на ряд принципов:

1. пользователь управляет компьютером, используя несколько физических устройств (клавиатура, мышка, микрофон, видеокамера и т.д.);
2. для коммуникации с компьютером пользователь активизирует движение ряда своих мышц (голосового тракта, рук, глаз и т.д.);
3. информация, передаваемая компьютерными устройствами ввода, может быть обработана на различных уровнях абстракции, обеспечивая различные уровни понимания намерения пользователя;
4. компьютер взаимодействует с пользователем, используя несколько устройств вывода (дисплей, динамики и т.д.);
5. по этим устройствам вывода компьютер может передавать заранее подготовленные данные (файлы с изображениями, аудио файлы и т.д.) или же динамически генерируемые данные (например, генерация текста, графики, синтез речи и т.д.).

Таким образом, компьютерная система может использовать несколько информационных каналов (чувств пользователя) для ввода и вывода.

1. **Подходы к проектированию ПИ**

Для получения эффективного результата разработки ПИ интерфейса используют различные подходы к проектированию:

1. **Подход, ориентированный на пользователя** (User Centered Design, UCD) — основным содержанием этого подхода является ориентация на пользователя, т. е. в первую очередь необходимо узнать, что хочет пользователь получить от проектируемого интерфейса. Далее в процессе проектирования полученные требования реализуются в продукте. При сборе информации используются методы наблюдения за работой пользователя, проводятся интервью.
2. **Системный подход** (System). Пользователь рассматривается как маленькая интеллектуальная часть системы «человек - программный продукт».
3. **Итеративный подход** (Agile) — метод последовательных приближений. Суть итеративного подхода заключается в создании изначально самого простейшего прототипа с целью показать заказчику н затем постепенно дорабатывать прототип, основываясь на реакции заказчика после каждого шага доработки.
4. **Деятельностный подход** (Activity Centered). Изучается деятельность пользователя в целом, и постепенно оптимизируются ее отдельные моменты.
5. 5. *Экспертный подход (Genius).* Заключается в следующем: эксперт собирает важную, по его мнению, информацию, ведёт переговоры с заказчиком, задаёт нужные вопросы. На основе полученной информации создаётся интерфейс.
6. *6. Целeориентированный подход* проектирования *(Goal Centered Design).* Разработка интерфейса ориентируется на цель, которая будет достигаться данным программным продуктом.
7. *7. Средоориентированный подход.* Разрабатывается среда интерфейса как место деятельности оператора.
8. При разработке интерфейса целесообразно гибко пользоваться указанными подходами, учитывая при выборе методов: назначение разрабатываемого продукта, целевую аудиторию, время и бюджет разработки.

При проектировании деятельности пользователей инструментальных интерфейсов важно обеспечить как устойчивость внимания и сосредоточенность, так и возможность переключения между видами работы. Задача проектировщика состоит в том, чтобы минимизировать сложность деятельности в рамках интерфейса, обеспечить системность интерфейса за счет того, что решение сходных задач должно реализовываться подобными действиями за счет одинаковых (или подобных) операций.

Деятельностный подход к проектированию человеко-компьютерного взаимодействия для конкретной проблемы предполагает глубокое изучение работы будущих пользователей в “докомпьютерном” варианте, анализ всех возникающих задач и описание деятельности по их решению.

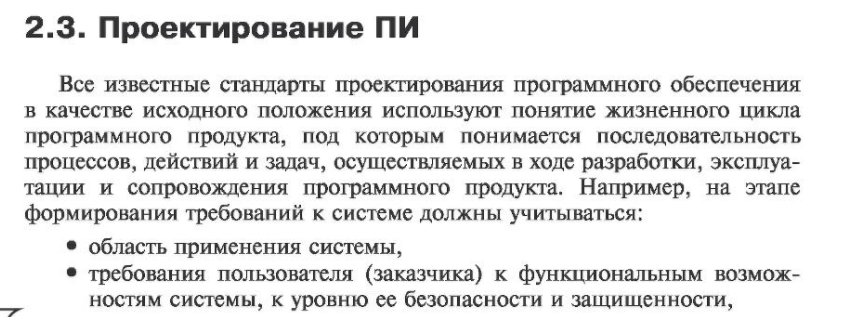
1. Сценарии в проектировании пользовательского интерфейса.

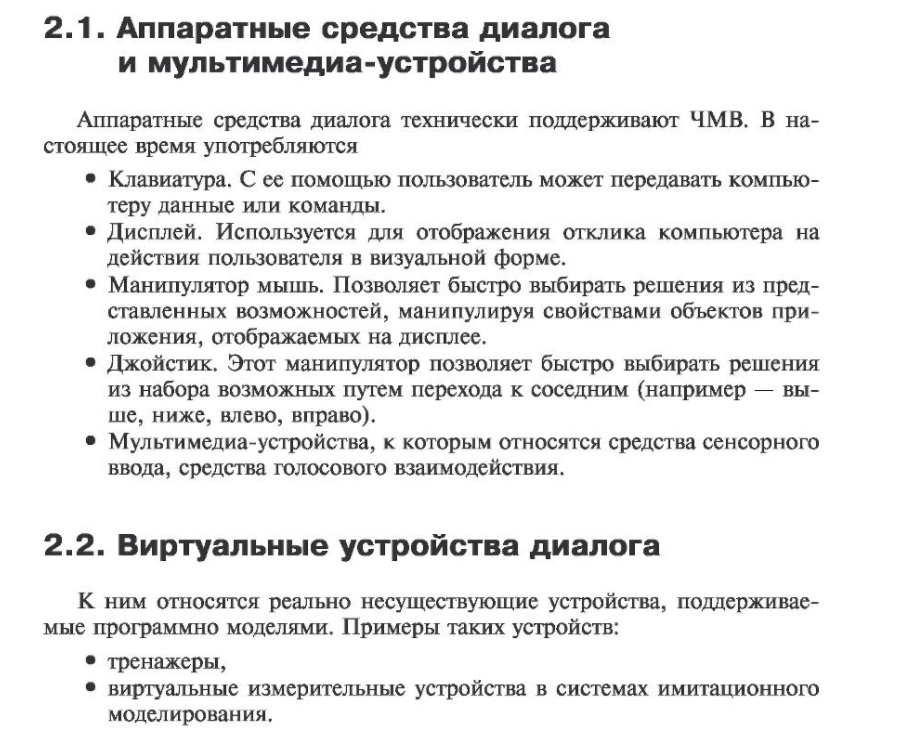
Первым шагом при создании функциональной спецификации является написание **сценариев пользователей**.

***Сценарий пользователя*** *—* краткое, простое описание того, как пользователь пытается удовлетворить потребности с помощью разрабатываемого продукта. Представив процедуру, через которую могли бы пройти потенциальные пользователи, можно более точно выработать требования к программному продукту.

Эффективность сценария определяется в большей степени его охватом, чем глубиной. Иначе говоря, важнее, чтобы сценарий описывал процесс от начата до конца, чем чтобы он описывал каждый шаг в исчерпывающих подробностях.

1. Средства поддержки проектирования ПИ



* http://www.studentlibrary.ru/pic/pix_blank.gif 

http://www.studentlibrary.ru/pic/pix_blank.gif

1. Методологии проектирования программных систем

Большинство методологий состоят из одних и тех же этапов:

* **Планирование** общей стратегии, подхода и структуры команды.
* **Формулирование** требований к проекту.
* **Проектирование** взаимодействия и визуальной концепции и их развитие до уровня подробных спецификаций.
* **Разработка**, тестирование и совершенствование решений.
* **Развертывание** разработанного продукта с использованием различных средств коммуникаций, проведение обучения и запуск в оговоренные сроки.
* **Сопровождение** проекта на основе рекомендаций по улучшению.

**Каскадная методология**

* В *каскадной* (*waterfall*) *методологии* этапы проекта рассматриваются как отдельные *фазы*, причем следующая фаза начинается только после утверждения предыдущей.
* Например, фаза проектирования реально начинается только после того как требования будут утверждены представителями бизнеса, которые подписывают документы с требованиями в конце фазы формулирования требований. Недостаток «чистой» каскадной методологии состоит в допущении, что завершение каждой следующей фазы требует минимальных изменений в результатах предшествующей фазы. Таким образом, если в фазе проектирования вдруг появятся новые требования (а это вполне обычное явление), вам придется вносить изменения в документы, утвержденные в конце фазы формулирования требований. Это может привести к нарушениям планов и графика проекта.

**Гибкие методологии**

* Поскольку постоянны только изменения, проектные команды находятся в постоянном поиске методологий, превосходящих по гибкости каскадную модель. Во многих методологиях используется более гибкий подход, когда некоторые этапы выполняются параллельно, например версии веб-сайта публикуются по ускоренному итеративному графику с использованием ***гибких* (*agile*)** или ***быстрых* (*rapid***) методологий. Гибкие методологии обычно сильнее ориентированы на оперативную совместную работу и в меньшей степени требуют подробного документирования и формальных утверждений.

Знание методологии, выбранной для проекта, помогает понять ряд вещей:

* **Какие вопросы и когда следует задавать.** Например, при работе по «чистой» каскадной методологии вам придется приложить дополнительные усилия к тому, чтобы в требованиях, созданных в фазе формулирования требований, содержалась вся информация, необходимая для фазы проектирования.
* **Как следует организовать совместную работу членов проектной команды и насколько тесным должно быть их взаимодействие**. Например, гибкая методология требует очень тесного сотрудничества, а в каскадных методологиях участники в основном работают сами по себе и вступают в контакт друг с другом один или несколько раз в неделю.
* **Насколько подробной и формальной должна быть документация.** Документы, предоставляемые на утверждение, должны быть формальными – почти как юридические договоры. Каскадные методологии, в которых переход к следующей фазе возможен только после утверждения результатов предыдущей, обычно требуют более формальной документации. Однако формальные документы могут потребоваться и в гибких методологиях, например для фиксирования информации в главных точках принятия решений .
* **Важнейшие контрольные точки, включающие в себя утверждение результатов представителями бизнеса и развертывание для разных групп пользователей.** Методология предопределяет, что именно участники должны совершать в определенных точках проекта: владельцы проекта утверждают результаты в контрольных точках, потенциальные пользователи предоставляют обратную связь при бета- тестировании и т. п.

1. Варианты использования (Use-Case) в проектировании человеко-машинного взаимодействия

Зачем нужны варианты использования?

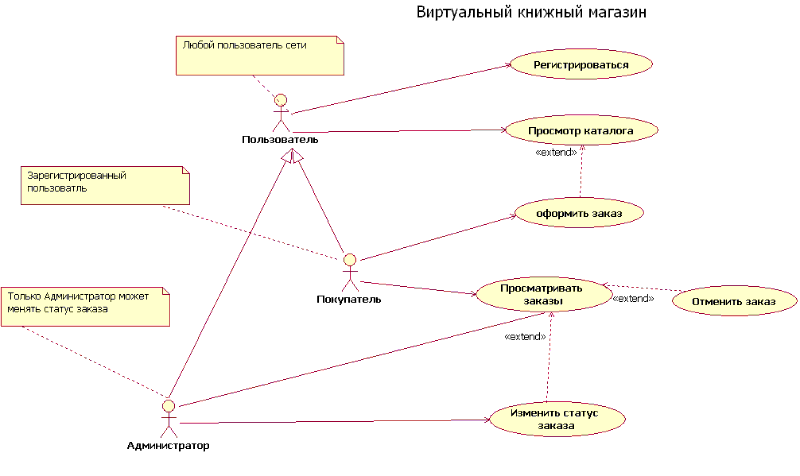
Варианты использования предназначены в первую очередь для определения функциональных требований к системе и управляют всем процессом разработки. Все основные виды деятельности такие как анализ, проектирование, тестирование выполняются на основе вариантов использования. Во время анализа и проектирования варианты использования позволяют понять как результаты, которые хочет получить пользователь влияют на архитектуру системы и как должны себя вести компоненты системы, для того чтобы реализовать нужную для пользователя функциональность.

В процессе тестирования, описанные ранее варианты использования позволяют проще оценить точность реализации требований пользователей и позволяют провести пошаговую проверку этих требований.

Стратегия использования прецедентов при определении требований определяет необходимость дополнительно к вопросу "что пользователи ждут от системы?" задавать вопрос "что система должна сделать для конкретного пользователя?". Такой подход позволяет искать функции, которые нужны многим пользователям и исключать те возможности, которые не могут помочь пользователям выполнять свои повседневные задачи.

Состав диаграммы Use Case

Диаграмма вариантов использования состоит из актеров, для которых система производит действие и собственно действия Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case - овалом. Дополнительно в диаграммы могут быть добавлены комментарии.



Пример диаграммы Use Case

Ответы на следующие вопросы позволят определить актеров, взаимодействующих с системой:

кто взаимодействует с системой или использует систему;

кто передает или принимает информацию в/из системы;

кто является внешним по отношению к системе.

Каждый вариант использования показывает, как конкретный актер использует систему и в дальнейшем расширяется диаграммами состояний и последовательности действий.

Виды взаимодействий

Между актерами и вариантами использования могут быть различные виды взаимодействия. Основные виды взаимодействия следующие:

Простая ассоциация - отражается линией между актером и вариантом использования (без стрелки). Отражает связь актера и варианта использования. На рисунке между актером администратор и вариантом использования просматривать заказ.

Направленная ассоциация - то же что и простая ассоциация, но показывает, что вариант использования инициализируется актером. Обозначается стрелкой.

Наследование - показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для актеров, так для вариантов использования.

Расширение (extend) - показывает, что вариант использования расширяет базовую последовательность действий и вставляет собственную последовательность. При этом в отличие от типа отношений "включение" расширенная последовательность может осуществляться в зависимости от определенных условий.

Включение - показывает, что вариант использования включается в базовую последовательность и выполняется всегда (на рисунке не показан).

Существуют и другие виды взаимодействия, но они, на мой взгляд менее важны и реже применяются.

1. Прототипирование. Виды прототипов.

Прототипи́рование (англ. prototyping от др.-греч. πρῶτος — первый и τύπος — отпечаток, оттиск; первообраз) — быстрая «черновая» реализация базовой функциональности для анализа работы системы в целом. На этапе прототипирования малыми усилиями создается работающая система (возможно неэффективно, с ошибками, и не в полной мере). Во время прототипирования видна более детальная картина устройства системы. Используется в машино- и приборостроении, программировании и во многих других областях техники. Прототипирование, по мнению некоторых разработчиков, является самым важным этапом разработки. После этапа прототипирования обязательно следуют этапы пересмотра архитектуры системы, разработки, реализации и тестирования конечного продукта.

Прототипирование не обязательно выполняется в рамках тех же технологий,что и разрабатываемая система. Как правило, прототип становится приложением к техническому заданию.

Качества которыми должен обладать эффективный прототип:

Этап создания прототипа не должен быть затяжным.

Эффективные прототипы являются одноразовыми. Они предназначены для того чтобы донести идею до заинтересованного лица. После того как идея была донесена, прототип может быть отвергнут.

Эффективные прототипы являются сфокусированными, это означает что следует обращать внимание на сложные части при создании прототипов. Необходимо найти паттерные взаимодействия, которые давно известны в теории usesexperience.

Необходимо обращать внимание на элементы взаимодействия, которые принесут пользу вашему продукту.

Существует 4 основных вида прототипов. Между собой они различаются на основе сферы применения модели, для которой делается прототип.

Промышленные прототипы, например электроники. Обычно они называются мастер-моделью.

Архитектурные презентационные макеты города, дома или отдельной комнаты.

Транспортные – прототипы любого транспортного средства (автомобиль, корабль, самолет и т.д.).

Товарный прототип – модель, которую используют для выставок и презентаций.