В начале видео автор говорит о том, что технологии развиваются быстро, в то время как мышление человека развивается медленно. Главная проблема – человек легко принимает новую технологию, при этом человек перестаёт принимать новые способы мышления, думать сам, находить новые способы решения проблем. Человек идёт по протоптанной дорожке. Пример из видео: люди, работающие с бинарным кодом не хотели воспринимать только что созданный ассемблер как язык программирования. Они считали, что бинарный код – настоящее программирование. Людям приходится переобучаться, изучать что-то новое и многие не хотят этого делать

Автор рассказывает про 4 основные идеи:

1. Coding -> direct manipulation of data   
   Суть заключается в том, что пользователи могут взаимодействовать с программой так, что вместо написания огромного количества кода программа сама подстраивалась под заданные пользователем ограничения и выполняла набор действий. Иными словами, поведение программы может динамически меняться в зависимости от заданных пользователем параметров, в особенности если параметры можно представить визуально (Паутина компьютеров, подключённых к какой-то сети)  
   2. Procedure -> goals and constraints   
   В будущем программистам не придётся писать строгую процедуру выполнения задачи. Для этого программе будет дана цель, которая будет достигнута, если действия пользователя будут отвечать определённым условиям или ограничениям (Примером в видео послужил планировщик:  
   когда мы говорим программе, что яблоки красные, то при получении яблока программа говорит: яблоко красное. В то же время, если мы потребуем от программы то-то красное, она может предложить яблоко). вы можете выразить свою программу с точки зрения целей и результатов, которые вы хотите от программы. Вы также можете предоставить процедурные стратегии для достижения определенных типов подцелей. Важно, что мы не устанавливаем инструкции для выполнения каких-то заданий, компьютер сам находит решение под нашу задачу  
   Ещё один пример: две программы, написанные на разных языках, в разное время разными людьми. Их задача – понять как друг с другом взаимодействовать, поэтому они сами должны понять как работать друг с другом.   
   3. Text dump -> spatial representations  
   Смысл заключается в том, что мы можем задавать информацию пространственно, придавать ей удобный для нас вид. Автор пытается сказать, что в будущем программисты не будут писать код в текстовых файлах, потому что есть более удобные способы как можно сделать это пространственно   
   4. Sequential -> concurrent  
   Компьютеры выполняют операции поочерёдно из-за особенности строения аппаратной части компьютера. Например: если компьютер имеет только один процессор, то он может выполнять операции только поочерёдно (Если брать как пример архитектуру Фон-Неймана). Во избежании простаивания и иррационального использования ресурсов, автор, как пример, предложил массив соединённых между собой процессоров. В силу того, что транзисторы на кремниевых платах становятся меньше, за счёт этого в пустое место на платах можно ставить больше процессоров, которые будут выполнять больше полезной работы. В силу того, что сегодня мы имеем один процессор на компьютер, то популярностью пользуется тактика «Потоков и блокировок». Автор считает, что такой путь работы с компьютером – билет в один конец, ведь данный подход не самый эффективный и если через 30 лет мы всё ещё будем работать с потоками и блокировками, то это говорить о некомпетентности ведущих умов программной и компьютерной инженерии. В качестве примера хорошего параллелизма в работе хардвэра автор привёл модель «актёр».   
   В конце видео автор говорит о том, что очень опасно, если современные программисты будут идти по протопанной дорожке и не поднимать новые идеи для улучшения работы компьютеров и программ  
   «Самое опасное для творческого человека – думать о том, что ты знаешь, что делаешь». Потому что, когда ты думаешь, что знаешь, что делаешь, ты перестаёшь искать новые способы сделать что-то. Ты перестаёшь видеть новые способы решения проблем  
   Главный посыл автора: консерватизм – плохо. Чтобы быть open-minded нужно сказать себе, что я не знаю, что я делаю. Именно благодаря этому ты по-настоящему можешь почувствовать, что ты свободен в своих действиях и ты можешь придумать всё, что угодно.

Эссе на тему видео The Future of Programming -- Bret Victor   
В данном видео Брет Виктор рассуждает о том, какое будущее ждёт программную инженерию. Автор привёл четыре ключевых изменения, которые, могут случиться в ближайшие 40 лет:   
1. Люди будут напрямую манипулировать данными - программа динамически меняет своё поведение в зависимости от действий пользователей без прямого взаимодействия с кодом.   
2. Программистам не придётся писать строгие процедуры для работы программы. Компьютеру будет дана цель, к решению которой он должен прийти сам, выполняя различные подзадачи  
3. Программы перестанут выглядеть как огромные текстовые файлы. Они приобретут пространственный вид. Иными словами, у программ могут появиться специальные интерфейсы, благодаря которым можно будет создавать содержимое нашего приложения.   
4. Появиться параллельное программирование. Это значит, что на смену «потокам и блокировкам» придёт одновременное выполнение нескольких задач несколькими процессорами, что существенно повысить производительность компьютеров

По моему мнению, Брет Виктор говорит про актуальные вещи. Мне понравился посыл автора о том, что программисты не должны ограничивать себя уже изобретёнными технологиями, а создавать что-то своё, поскольку идти по протоптанной дорожке – самый простой путь, который не приведёт программную инженерию к новым открытиям. Виктор показал 4 основные концепции будущего, которые должны вывести программирование на новый уровень. Данные идеи подчёркивают проблемы современного программирования (Например: недостатки потоков и блокировок), однако есть и некоторые спорные моменты, такие как пространственный вид программ. Насколько я понял, Брет Виктор говорит, что программы будут иметь некий интерфейс, благодаря которому мы способны легко создавать новые элементы внутри нашего приложения, без прописывания кода. Однако, как мне кажется, элементы, что мы создаём, всё равно должны храниться в программе как кусок кода. Выходит, что проблема, не уходит полностью, просто программисту предоставляется более удобный интерфейс для взаимодействия с программой. С другой стороны, мне очень понравилась идея того, что программисты в будущем перестанут писать строгие процедуры для выполнения программы. Вместо этого компьютерам будет даваться какая-то высшая цель и вычислительная машина должна сама найти решение, выполняя различные подцели. Благодаря этому пропадёт нужда в человеке, который будет вручную, как пример, создавать соединение между двумя компьютерам – они всё сделают сами. Также достаточно привлекательно звучит идея о параллельном программировании, так как стандартная схема потоков и блокировок – максимум, что можно сделать в машинах с фон-Неймановской архитектурой и одним процессором. Часто бывают ситуации, когда часть памяти простаивает, пока другая часть работает с CPU. Если же в компьютерах будут стоять множество различных процессоров, то производительность существенно возрастёт, однако необходимо тщательно продумать то, как такая система будет работать без серьёзных ошибок или утечек данных. Что касается прямой манипуляции данными, то я уверен, что такие технологии уже существуют и это весьма полезная вещь, поскольку программистам не нужно лезть в код, чтобы изменить, к примеру, внешний вид приложения или сайта. Виктор сказал, что если через 40 лет мы всё также будем писать код с процедурами напрямую, текстом, используя потоки и блокировки, то это значит, что программные инженеры потерпели крупную неудачу, так как не смогли создать что-то новое. Лично для меня это достаточно спорный тезис, ведь не во всех сферах применимы те технологии, которые он считает прорывными. К тому же данному видео уже больше 10 лет и я уверен, что за это время инженеры уже продвинулись далеко вперёд, создав технологичные решения. Подведя итог, хотелось бы сказать, что с большинством тезисов автора можно согласиться. Все 4 «концепции будущего» несут в себе пользу для программной инженерии и, вполне возможно, станут неотъемлемой частью в работе программистов