В своем выступлении Бретт Виктор анализирует эволюцию программирования, отмечая, что технологии меняются очень быстро, однако мышление людей развивается с меньшей скоростью. Это создает бОльшую проблему – сопротивление идеям и отдача предпочтения старому, привычному, несмотря на объективное улучшение, как показал пример перехода от машинного кода к ассемблеру на IBM 650. Виктор, опираясь на исследования ХХ века, выделяет четыре ключевых направления, которые могут изменить подход к программированию:

1. Прямое манипулирование данными – визуальный подход, где пользователи взаимодействуют с данными, позволяя системе неявно генерировать код (например, рисование с ограничениями в Sketchpad Сазерленда).
2. Программирование с использованием целей – декларативный подход, вместо императивного описания действий. В частности, в Planner или Prolog сопоставление с образцом и логический вывод приводят к макроуправлению и автономному «общению» программ.
3. Пространственное представление информации - переход от текста к визуализации (иерархии, графы зависимостей, интерактивные диаграммы) для лучшего понимания и взаимодействия с кодом (NLS Энгельбарта, GRAIL, Smalltalk).
4. Параллельное программирование – отказ от последовательной архитектуры фон Неймана (один процессор + большая память), в пользу массового параллелизма, например, через асинхронную модель Акторов с обменом сообщениями.

Виктор подчеркивает - лишь после нашего признания того, что мы не знаем, что такое программирование как таковое, мы сможем по-настоящему творить и создавать будущее, переходя к новым мощностям, новой доступности и новой естественности.

Уже сейчас, через десятилетие, мы можем видеть рост данных направлений:

Low-code/no-code платформы, визуальные конструкторы, интерактивные среды анализа данных позволяют ускорить разработку моделей и облегчить создание софта. К примеру, Figma, Webflow, Tableu, Observable, Elma365 являются площадками, позволяющими человеку погружаться в прямое манипулирование. Однако стоит заметить, что до полного отказа от текстового коддинга еще далеко.

Декларативные языки и системы искусственного интеллекта (SQL, React, Copilot, ChatGPT, Cursor) активно развивают парадигму управления целями, несмотря на существующие проблемы надежности и безопасности.

Инструменты для визуализации данных и архитектуры (Miro, D3.js) и современные IDE с навигацией по структуре и графам стали незаменимыми, хотя текст все еще является «источником истины».

Многоядерные процессоры и распределенные системы сделали направление параллелизма критическим. Модели Акторов (Akka, Erlang/Elixir) доказали свою эффективность, но сложность разработки и отладки остается большой преградой для массового применения.

В этой теме я нахожу две ключевые проблемы. Несмотря на то, что идеи прямого манипулирования и визуализации уже реализованы во многих сферах программирования и бизнеса в целом, доминирующей схемой до сих пор остается текстовый код. В последние 30 лет вложено уже очень много инвестиций – существующие языки, фреймворки, образовательные программы и опыты разработчиков, поэтому переход требует высоких рисков и колоссальных усилий, которые бизнесы не способны принять «в моменте», пока старое находится в работоспособном состоянии.

Во-вторых, стоит сказать, что аппаратура ушла далеко вперед, однако основные языки (Python, C#, Java) и мышление большинства программистов все еще последовательны. Необходима не просто новая архитектура, а фундаментально новый способ изучения программирования, начиная с самой базы. Распределенные и параллельные системы требуют более сложного мышления и создают новые виды различных сложных ошибок, которые трудно уловить.

В целом, Бретт Виктор прекрасно обозначил векторы развития, которые уже воплощаются в наше время, а призыв к открытости и борьбе с привычным – «мастхев» для каждого. Будущее в программировании не в слепом следовании его привычным паттернам, а в готовности к развитию, экспериментам и принятию нового, борясь с инерцией сообщества. Прогресс зависит не только от технологий, но и от нашей способности мыслить иначе.