Blindsight: технотеология против цифрового кода

Михаил Куртов. Генезис графического пользовательского интерфейса. К теологии кода. СПб.: ТрансЛит, 2014. — 88 с.

Современность боится цифровой культуры и тех эффектов, которые та производит. Этот страх тесно спаян с *неизвестностью* и *инаковостью*: подобное лавкрафтианское соединение характерно для теряющего силу Просвещения; хоррор-характер связи, возникающей между *страшным* и *неизвестным*, маркирует приверженность эпохи старому языку описания, слабо применимому к разговору о цифровом. Йог-Сотот радио и телевидения страшен, но он страшен в явленности своего тела и неявленности своего ума, в случае же цифрового кода мы имеем дело, скорее, с «Роршахом» Питера Уоттса, рафинированной неизвестностью, которая порождает эффект испуга только в ситуации осознания того, что та перемещается в слепых пятнах человеческого зрения, изымая себя из восприятия. Эта сущность манифестирует себя, скрываясь, и технотеологический проект, признанием необходимости существования которого заканчивается данная книга, призван стать постделёзианским орудием документирования и вскрытия внутреннего содержания этого процесса.

Интуиции Жильбера Симондона об автономной эволюции техники, развертывание которой подобно эволюции биологического, диалектика Абсолюта Гегеля, общая постделёзианская установка на сшивание разрозненного, лежащего поодаль и ждущего соединения, а также ряд основоположений, имплицированных акторно-сетевой теорией, становятся эвристическими составляющими технотеологического механизма, сконструированного специально для анализа строго *не-механических* объектов.

Бог умер — и священное рассеялось по миру, гибридно переходя из одного локуса в другой, оказалось (для философии) дело-

кализовано и перераспределено, сохранив при этом общий эволюционный характер развертывания онтотеологии, теперь присутствующей в мире скрыто. Мнимая гибель священного несет в себе характер эвкатастрофы — «хорошего падения вниз», ситуации, в которой Бог исчезает из поля зрения спекулятивного мышления, но, с неизбежными временными и структурными флуктуациями, вновь возникает в совершенно неожиданном месте — в *computer science*, становящейся новой точкой локализации теологического.

Так можно выразить основную мысль вышеозначенной книги, выраженную в гипотезе *операторной рекапитуляции* — ситуации, в которой развитие абстракций в *computer science* кратко и сжато повторяет эволюцию онтотеологии. Такое перебрасывание мостков между технологией и теологией становится возможным благодаря авторскому допущению о гомологичности программирования и теологии, которое маркирует возможность разговора о цифровом в теологических категориях не на уровне метафор, а на стадии допущения возможности существования между ними морфологической эквивалентности.

Импульс, запускающий технотеологический механизм, — введение в поле зрения философии графического пользовательского интерфейса (*GUI*) и того, что стоит за ним, — цифрового кода. Графический интерфейс одновременно превращается в осевую сущность (*entity*) текста, становясь опорой технотеологической аргументации, и является метафизическим стержнем, постоянно переопределяемым в ходе раскрытия своей истории и через эти акты присвоения имени высвечивающим аспекты операторной рекапитуляции. На уровень глубже лежит другая опорная конструкция текста, неявно выраженная в определении *GUI* диада «контекст/онтотеология». Дефиниция *GUI* за все время повествования претерпевает три трансформации: в первую очередь задается ее очевидная завязанность на ассоциацию «клавиатура + мышь + дисплей». Затем в качестве сущностно важного для понимания современного персонального компьютера вводится объектно-ориентированное программирование (ООП), сделавшее возможным простое (по сравнению с использованием предыдущих парадигм) программирование графических интерфейсов. Синергия этих компонент становится ассоциацией, фундирующей возникновение и расцвет компьютеров как привычных в нашей повседневности предметов обихода. Затем в конструкции «(клавиатура + мышь + дисплей) + (ООП)» второй член выражения сменяется ключевой для понимания современного

компьютинга абстракцией (которая, правда, на данный момент находится в затянувшемся концептуальном кризисе) *Model — View — Controller*, упорядочивающей понимание компьютера как совокупности данных, а также средств ввода и вывода. Такие порядки переопределения соответствуют шагам эволюции абстракций, начавшейся в Древней Греции и не завершившейся по сей день. Ее этапы — технические варианты досократической философии первовещества (в данном случае диада «структура-становление», задающаяся Гераклитом и Парменидом, репрезентирована императивной и функциональной парадигмами программирования), изобретение понятий формы и объекта Платоном и Аристотелем (объектно-ориентированное программирование), а также абстрактная схема логики компьютинга *Model — View — Controller*, гомологичная августиновской Троице «Отец — Сын — Святой Дух». На этом этапе технотеология прекращает свою работу, достигнув современности, и делается вывод: ситуация в современной *computer science* не аналогична, но гомологична ситуации, сложившейся вокруг проблемы филиокве в современной схоластике.

В определении *GUI* на неявном уровне всегда присутствуют два слоя, составляющих уже упомянутую таксономию «контекст/онтотеология», в которой код становится напряжением между этими двумя полюсами. Ассоциация клавиатуры, мыши и дисплея контекстуальна и исторична, завязана на эволюцию технических, но не цифровых объектов и потому чисто инструментальна. Вторая же компонента определения, объектно-ориентированное программирование и, вслед за ним, схема *MVC*, представляет онтотеологический полюс, чистую развертку сущностно философской концепции, совершенно прямым путем вошедший в *computer science* тогда, когда Алан Кэй, один из пионеров современной информатики, артикулировал необходимость применения идей Платона в программировании.

Мы хотели бы отстоять точку зрения, согласно которой сопротивление попыткам редукции имманентно природе кода как объекта, благодаря чему любая попытка его концептуализации приводит к ускользанию от спекулятивного мыслителя сущности этого объекта, моделирует ситуацию, в которой код не остается кодом, а превращается (в случае технотеологии) в постоянно становящееся хранилище абстракций, эволюционное движение, в корне которого лежит желание экспроприации теологического, в конечном счете снимающееся во взрыве, экспроприирующем саму концепцию в теолого-метафизической контратаке.

Такое ускользание нередуцируемого превращает код в этап уже упомянутой эволюции абстракций, движение, чистое процессуальное становление, которое парадоксальным образом (и в этом сохраняется мотив дуального развертывания структуры-операции) становится статичной, четко определенной тюрьмой, уничтожающей любые попытки благотворной объективации кода путем выдавливания его в область отсутствующего.

Иными словами, принимая технотеологический язык объяснения, исследователь производит процедуру элиминации, превращая код в этап гомогенного движения, *собственно кодом* не являющийся как раз в силу примата становления в его определении. В таком понимании код не может быть объективирован — только концептуализирован. Это не мешает основной задаче технотеологии как постделёзианского инструментального механизма предсказания и объяснения, но вредит пониманию кода как независимой сущности, которая имплицирует свои собственные внутренние онтологические категории и способна многое рассказать философии безотносительно технотеологического проекта, который, в терминах Грэма Хармана, будет попыткой «подрыва» такого странного и гетерогенного объекта, как цифровой код. «Надрывом» в данном случае будет необходимость технотеологического рассмотрения не кода, но того, что стоит за ним, скрывается от взгляда кого угодно, только не технотеолога.

В случае теологии кода объяснение оказывается заперто в ловушке между *аналогией* и *гомологией* — достаточную объяснительную и прогностическую силу этот механизм обретает, только будучи фундирован метафизической необходимостью эволюции абстракций, эволюции, для которой развитие техники декларируется центральным и единственным сюжетом, а «человеческое, слишком человеческое» оказывается только рядом сопутствующих факторов. Проблематичность технотеологического возникает тогда, когда эволюция техники заменяется эволюцией абстракций, центральным сюжетом которой являются контекст и историко-культурные предпосылки формирования технических объектов, становящиеся эпифеноменами исторического развертывания кода. Второй стороной такого развития становится скрытое в техническом онтотеологическое, гомологически сшивающее объектно-ориентированное программирование с идеями Платона, а дискуссии, на данный момент разворачивающиеся вокруг модели *Model — View — Controller*, — с проблемой филиокве.

Введение разделения «контекст/онтотеология» не дает развернуться в качестве самостоятельного актора самому коду, ле-

жащему между двумя этими полюсами, которые самой процедурой задания подобной таксономии выдавливают код как объект из поля философской речи, в то время как именно в нем мы находим *чуждое*, с точки зрения этики прироста знания, возможно, самое ценное, что мы можем найти в цифровых объектах. Отсюда — технотеология способна дать нам только те знания, *которые мы уже имеем в распоряжении*: исторический контекст и онтотеологическую эволюцию философии. Но она отказывается от попытки рассмотреть код в его внутренних онтологических категориях. Аргумент гомологии же представляется нам одновременно внутреннерепрессивной и внешнеполитической деталью технотеологического изобретения, уничтожающей любые попытки рассмотрения вариативности развития медиа, на которые нам указывает медиаархеология (ее можно понимать как археологию потерянных путей и неудавшихся концептуальных революций). Политически же гомология может выступать способом обоснования того, обоснование чего на принципе аналогии было бы проблематично.

Кроме того, эволюция абстракций по принципу гомологии с эволюцией биологических сущностей должна развертываться в механицистской, дискретной логике, подразумевающей возможность деления на этапы и части, которые на поверхностном уровне рассмотрения технотеологического проекта считываются как этапы развития *computer science*. Однако при более глубоком взгляде история абстракций, фундируемая концептом гегелевской развертки Абсолюта (в данном случае это слово не будет простым реферированием Гегеля: одно из последовавших за публикацией книги выступлений Михаила Куртова было озаглавлено «Код как Абсолют»), не признаёт онтологии вещей и самой возможности онтологического различения между этапами становления (точнее, возможности различения этапов становления), достаточно четко артикулируемыми как в истории *computer science*, так и в биологической теории, из которой в случае технотеологии изымается такой объект, как единица эволюции.

У кода забирают возможность не только высказывания, но и имени, и эта возможность должна быть экспроприирована в пользу ее хозяина: понимание кода как становления и эволюции позволяет использовать его в качестве линии маркирования развития, но и только. Ответ на вопрос, *чем является код* или *как он есть*, в случае технотеологии не дается исходя из базовых аксиом, фундирующих развертку этой концептуальной схемы: для технотеолога полем работы выступают не попытки

определения кода, а объяснение, фундирующееся уже заданным определением.

Коду еще не давали голос как независимой, автономной сущности, объекту, топология которого проблематична, но возможна в терминах харманновской или постхарманновской онтографии и может быть проиллюстрирована простым примером феноменально улавливаемого онтологического различия между записанным на листе бумаги кодом программы и этим же кодом, находящимся в процессе выполнения, которое отсылает к логике дискретного становления (о нем говорит, к примеру, Ян Богост в своей статье *Proccess vs Procedure*[1]). Репрессивный механизм гомологии также редуцирует региональные онтологии, неизбежно имплицируемые языками программирования высокого уровня. Находясь в логике иерархии «сверху вниз», они фундируют семантическое в производимой программе, влияют на ее конечный облик, производя на свет большое количество онтологий, которые, в отличие от философских, строятся на принципах *инструментальности*, а не *полноты*. Такие семантические системы при определенном взгляде могут создать ситуацию, в которой язык программирования *Python 2.7* окажется более метафизичным, чем трактаты Мартина Хайдеггера: условием возможности его применения является онтологическое введение перечня основополагающих сущностей и законов их взаимодействия. Заметание подобного онтологического слоя, элиминация самой возможности такой оптики, которую можно назвать *онтологией онтологий*, описанием странных, причудливых взаимодействий, возникающих между часто онтологически противоположными метафизическими схемами, которые представлены блоками кода одной программы, написанными на разных языках программирования, кажется нам скорее политическим, чем философским актом. Вытеснение метафизической реальности, имплицируемой этим взаимодействием, может вредить постулату гомологии как единого однонаправленного движения из-за своей «ветвистости» и вариативности, но не может вредить коду как независимому объекту. Трансматериальная сеть технотеологии схватывает контекстуальное и онтотеологическое, но упускает главное — сам цифровой код. Технотеологический проект постулирует код

1. *Bogost I.* Process vs Procedure. Paper presented at the Fourth International Conference of the Whitehead Research Project // Bogost.com 6.12.2010. URL: http://bogost.com/writing/process_vs_procedure.

как нечто «сущностно новое», но *новой сущности* в этой оптике технотеолог не находит.

Код сложноредуцируем, его границы постоянно находятся под вопросом, однако технотеологический способ объяснения не является единственным языком, на котором можно описывать эту сущность. В конечном счете симондоновская интуиция спасения технических объектов от отчуждения выступает как противовес политическому смыслу технотеологического проекта: код должен быть спасен — и спасен от всего, кроме себя самого.

Степан Козлов