1.3. ВВЕДЕНИЕ В ХАОТИЧЕСКОЕ УПРАВЛЕНИЕ (СЕТЕВОЙ ВАРИАНТ)

Непейвода Н.Н. д.ф.-м.н. ИПС РАН

Главная ахиллесова пята нынешних концепций бизнес-процесса — превращение людей в процессоры. Все концепции валятся, если агенты проявляют инициативу. В данной работе описывается метод организации деятельности в среде инициативных и ненадежных исполнителей. При этом уничтожается само понятие процесса и заменяется планированием по целям и приоритетам, причём приоритеты не могут выражаться действительными числами. Обосновывается, что максимальная глубина планов должна быть не больше, чем три шага, и что к планам нужно относиться как к временным виртуальным сущностям. Предлагается, какие средства могут использоваться для реализации данной концепции.

1. Несколько слов об истории работы

Работа появилась в 2006-2008 годах в результате исследований по проекту для администрации г. Ижевска. Первый её вариант был опубликован в качестве пленарного доклада на конференции ТИПД-2008. Выяснилось, что за 12 лет её идеи не перестали быть интересными и до сих пор нигде не реализованы. Поэтому редакция журнала попросила меня написать статью на эту тему.

Так как сейчас нахожусь в глубокой самоизоляции в дальней деревеньке Ярославской области и интернет здесь отвратительный и часто сбоит, обновить список ссылок как следует не удалось, приношу за это свои извинения и гарантирую, что после выхода из заточения приведу их в порядок.

2. Постановка задачи

В теории программирования и управления две священные коровы[1]. Более известная — оптимизация, и более коварная, практически нигде не выделяемая явно — процессы. Поэтому все предложенные методы начинают просто дезориентировать, когда процент непредусмотренных ситуаций становится более 20, а помогают как следует лишь тогда, когда он меньше 10. Здесь рассматривается альтернативный и дополнительный подход: мы стремимся создать инструмент, работающий в условиях полного хаоса, а затем учесть порядок как исключение из типичной ситуации.

3. Предпосылки и напоминания

Сначала пару слов о позиции автора. Автор — консерватор (принцип консерватизма: идеи нужно приспосабливать к реальным людям, а не наоборот), и монархист (поскольку демократия, согласно определению Аристотеля, возможна лишь на базе системы законов и обычаев, и, соответственно, говорить о ней, когда нет системы законов и разрушены обычаи, просто нелепо). Ещё более укрепила автора в этой концепции поездка в Италию, знакомство с весьма хаотичным народом, который стал жить хорошо лишь когда принципы приспособили к реальностям итальянского характера, а не наоборот. А наши западники на самом деле московиты, для которых нет никакой разницы между «итальянским немцем» и «аглицким немцем», и всё равно, у кого брать опыт.

Прежде всего, напомним один из базисных принципов теории творческого мышления (в приложении к техническим системам описанной, в частности, в [9]): с самого начала выявить и обострить основное проблемное противоречие, а в дальнейшем так же поступать с остальными выявляющимися проблемами. Поэтому все появляющиеся проблемы ставятся по возможности жёстко и безжалостно.

Далее, напомним, что модальность «в принципе возможно» нужно понимать «практически невозможно[2]». Если бы об этом чаще помнили, меньше было бы ситуаций типа «Хотели, как лучше, а получилось, как всегда» (Черномырдин).

Весь анализ базируется на теории неформализуемых понятий (её изложение дано, в частности, в учебнике [3]). Основные её принципы и результаты, применительно к нашему анализу, следующие.

1. Каждое действительно сложное и важное понятие активно сопротивляется формализации, порождая новую альтернативу, не учтённую ею.

   В частности, русский человек и русский чиновник всегда нарушает и будет нарушать инструкции, поскольку инструкции претендуют на формализацию реальной жизни, и делают это на базе абстрактных принципов. Как говорили в советские времена, «что-либо разумное можно сделать лишь в порядке исключения».

2. Тем не менее эффективная деятельность возможна лишь на базе формализации понятий.

---

[1] Здесь это термин. См. [3]

[2] А заодно есть смысл посмотреть и другие принципы взаимодействия практика и теоретика из [7]

Если нужно что-то сделать, нужно временно избавиться от сомнений. Плохо исполненная инструкция лучше, чем её полное отсутствие. Это хорошо видно в программировании, где все системы и языки программирования являются формализациями неформализуемой деятельности, которая часто без них была бы просто невозможна, но которая начинает вылезать за их пределы, как только развивается достаточно.

3. Необходимо чётко понимать, что мы в огромном числе случаев формализуем неформализуемое.

Эвфемизмы типа «данная деятельность трудно формализуема» заводят в быстрый концептуальный тупик. Формализация на самом деле сама порождает себе альтернативу. Такова, например, знаменитая теорема Гёделя о неполноте[3]. Но процесс формального порождения альтернативы на самом деле работает лишь в принципе. Нахождение действительно интересной альтернативы — творческий процесс.

4. Знание о формализации предполагает знание ее альтернатив.

Например, знание классической евклидовой геометрии стало полнее, и ее громадные преимущества были чётко осознаны лишь тогда, когда Лобачевский переоткрыл неевклидову геометрию, которую древние греки знали ещё до Евклида, сознательно отвергли и забыли[4].

5. Любая формализация заводит в тупик, и тем быстрее и основательнее, чем более эффективна она на первых порах.

Примером служит, в частности, завоевавшая монопольное положение в мировой науке система грантов. Первоначально она сыграла роль допинга, позволив быстро выкачать результаты, которые уже были на подходе (хотя, насколько видно, в частности, по информатике, где автор внимательно за этим следил, одновременно помешала их углублению и доведению до логического конца, поскольку это требует кропотливой и внешне неэффектной работы: «Если хочешь сделать что-то большое и чистое, пойди в зоопарк и вымой слона»). В дальнейшем этот допинг оказался наркотиком, поскольку наука села на иглу легко объяснимым профанам обещаний и стала крутиться в цикле с периодом приблизительно 12 лет (именно через такое время наблюдается возвращение старых концепций под новым именем).

6. Тупик образуется ещё тогда, когда отказаться от формализации нет сил. Смотри предыдущий пример.

7. Тупик становится намного труднее для выхода и основательнее, если с самого начала не позаботиться, чтобы осознать альтернативы существующей успешной формализации.

Смотри пример из п. 4.

8. Тупика можно избежать, если одновременно поддерживаются несколько несовместимых альтернатив.

Даже в императорском Китае, славном формализованной системой аттестации кадров (государственные экзамены), в которой было детально регламентировано по возможности (для русского человека до невозможности) все, периодически в качестве альтернативы формализованным экзаменам устраивались «экзамены цветов и плодов», на которых выявлялись способные люди, не укладывавшиеся в формализованную систему аттестации.

Теперь перейдем к другой предпосылке — теории мягких моделей В. И. Арнольда [1]. Основные положения теории Арнольда можно сформулировать для наших целей следующим образом.

1. Наиболее легко понимаемые и привлекательные модели чаще всего дезориентируют на практике.

2. Исключительно опасной является операция оптимизации, поскольку оптимум ищется на базе существующего теоретического приближения и может оказаться просто гибельным в реальных условиях.

3. Поэтому любой теоретический совет требует двойной перепроверки: на, скажем грубо, линейном и нелинейном уровне.

4. Если линейное приближение брать в рядовой ситуации, достаточно далеко от оптимумов и особых точек, то оно, как правило, хорошо работает.

---

[3]В качестве отступления можно заметить, что предложенные методы не претендуют на преодоление теоремы Гёделя или теоремы Чейтина о непознаваемом, хотя многие физики и философы выражали уверенность, что недетерминированность вернёт нам иллюзию всесилия рассудка. В частности. в работе [8] (см. также [3]) показано, что эти теоремы не зависят от предположения о детерминированности и от наших конкретных формулировок понятия алгоритма.

[4]Последнее было их единственной ошибкой!

5. Если линейное приближение хорошо работает вблизи оптимума, то оптимум является потенциальной ямой, из которой будет трудно выбраться в случае изменения условий.

В целом все вышеизложенное можно резюмировать следующим образом: оптимальность, устойчивость и жизнеспособность противоречат друг другу. В реальности, достигая двух из этих целей, мы убиваем третью. Таким образом, нужно сознательно выбирать, чем жертвовать: устойчивостью или оптимальностью.

Следующая предпосылка возникла из теории уровней знаний и умений (кн. Белосельский-Белозерский, конец XVIII века, современное изложение и развитие в соответствии с результатами науки XX века см., например, в работе [4]). Основные положения этой концепции можно сформулировать для наших целей следующим образом.

1. Уровни знаний и умений — тонкие сферы в пространстве невежества и полузнания. Между ними достаточно большие расстояния, и перейти ползком с одной на другую невозможно.

2. По этой причине чуть недоученный человек лучше чуть переученного.

3. Достоинства некоторого уровня обычно становятся недостатками следующего, и наоборот. Поэтому линии поведения на разных уровнях совершенно различны. По этой же причине высшие сферы отнюдь не обладают абсолютными преимуществами по сравнению с низшими: “мамы всякие нужны.”

4. Для высших уровней, начиная с уровня метода, знания и умения начинают обладать следующим общим свойством: они становятся в значительной мере независимы от конкретной предметной области и могут прикладываться во внешне совершенно несвязанных областях. Более того, конкретные их применения могут выглядеть для внешнего наблюдателя совершенно различными по форме, поскольку их связывают прежде всего высокоуровневые идеальные структуры.

5. Поэтому часто высокоуровневое знание, применённое в конкретной области, остаётся вещью в себе и выглядит случайной догадкой, пока оно не конкретизировано для людей, стоящих на более низких сферах.

6. Одной из высших сфер знания, присущей менталитету русских специалистов, является сфера многоуровневого критического мышления, соответствующая теории неформализуемых понятий.

С этой концепцией связана структура данной работы: можно было бы просто предложить метод хаотического управления, но тогда бы он выглядел именно вещью в себе, неизвестно откуда взявшейся, и его применение могло бы оказаться более тяжёлой ошибкой, чем неприменение в подходящем для него случае, поскольку он немедленно оброс бы кучей «присущих ему особенностей», на самом деле к сути его отношения не имеющих.

Теперь применим данную концепцию к конкретным русским людям, с которыми мы живем.

•Сильная сторона русских — негативное мышление.

•Позитивное мышление концентрируется на том, чтобы везде увидеть положительные стороны. Оно оптимистично и на первом уровне ведет к успеху.

•Позитивное мышление заставляет закрывать глаза на неудачи. Оно соответствует подростковому мировоззрению американцев и стратегии прямых действий, наименее эффективной и наиболее расточительной, но весьма успешной в случае большого избытка ресурсов и толкового управления ими в течение действия[5]. Неудачи являются наиболее ценным источником информации о путях управления, если их, конечно, проанализировать жёстко и спокойно.

•Негативное мышление концентрируется на препятствиях и недостатках, но на втором уровне — на том, как использовать негативные факторы в положительную сторону.

•Оно соответствует мировоззрению зрелых людей и стратегии непрямых действий. Заставить противника одержать пиррову победу обычно намного выгоднее, чем разбить его в лобовом столкновении.

•Оно требует намного больших затрат интеллектуальных ресурсов, чем позитивное, и плохо поддаётся алгоритмизации.

•Оно специализируется на том, чтобы выглядящее как неудача стало в конечном итоге ступенькой на пути к цели.

---

[5]Примером здесь является американская стратегия времён второй мировой войны. При отвратительном качестве своих воинов был создан громадный избыток технических средств и, самое главное, разработаны и применены методы эффективного управления ими (исследование операций). А без этого, скажем, Роммель или адмирал Того били превосходящие ‘всего в разы’ силы американцев и англичан.

И, наконец, рассмотрим концепцию Д. Гильберта, которая также играет важную роль в нашем анализе. Её основные для нас положения следующие.

1. Большинство высокоуровневых понятий не могут иметь прямой реализации на практике. Они требуют для применения подстановки в них других теоретических концепций и целых планов действий. И лишь после такой подстановки получается реализуемый объект.

2. Тем не менее действительно сильное решение получается лишь при их использовании. Проигрыш при использовании лишь эмпирических понятий – просто фантастический: башня экспонент.

3. Эффективное решение характеризуется важнейшей ролью призраков: понятий, не присутствующих в конкретной программе (плане), но необходимых для её осознанного применения и модификации. Примером такого призрака является предположение об устойчивости математической формализации системы. Устойчивость может быть проверена лишь косвенно, но без неё решение будет реализуемо лишь в принципе.

4. Оборотной стороной призраков являются подпорки: не нужны для решения задачи, но вставляются с тем, чтобы согласовать концепции с имеющимися инструментами либо чуть-чуть выиграть в эффективности. Это, в частности, в программировании конкретные приемы оптимизации программ.

Во избежание недоразумений заметим, что наша концепция не имеет никакого отношения к идеям Пригожина. Мы рассматриваем не самоорганизацию хаоса, а организацию работы целенаправленного агента в хаосе.

4. Основные элементы концепции

Прежде всего, выделим критические моменты анализа нынешнего положения дел и основные направления, на которые ориентирована концепция.

Концепция зародилась, когда в соответствии с теорией неформализуемости и общими принципами негативного многоуровневого критического мышления возник следующий вопрос. Система стилей программирования, описанная в книгах [5, 6], сама основана на теории неформализуемых понятий и классифицирует столь разные концепции, как сентенциальное, автоматное, структурное, событийное и функциональное программирование. Но тем не менее: какова же её естественная альтернатива? Для ответа необходимо прежде всего выделить священную корову, которая неявно присутствует во всех работах по программированию. Конечно, искать нужно, прежде всего, нечто вообще необсуждаемое и автоматически принимаемое как данное, либо упрятываемое во внешне невинное математическое определение.

Главный идол программирования — понятие *процесса.* Программа — это заранее распланированный процесс. Конечно же, сама возможность распланировать процесс на много шагов вперед стала уникальным расширением наших возможностей управления и исключительно ценна. Но возникает другой (ехидный) вопрос: а когда же это не работает?

Современное программирование для учёта реальной неполноты наших знаний и реального хаоса, который привносят и программисты, и пользователи, и другие непредсказуемо взаимодействующие элементы программного окружения, пользуется понятием исключительной ситуации. Опыт показывает, что программа, в которой достаточно много исключительных ситуаций, начинает разваливаться под их тяжестью. В классических работах по структурному программированию говорилось, что процент исключительных ситуаций заведомо должен быть меньше 5%, некоторые из методик индустриального и экстремального программирования, базирующиеся на ООП, доводят допустимую частоту их до 10% (если брать рекламные оптимистичные заявления).

Итак, вторая критическая характеристика — это *альтернатива между реальным хаосом (хотя бы частичным), и принципиальным порядком.* Теперь рассмотрим само понятие хаоса, которое становится центральным в нашей работе. Оно опять-таки совершенно отлично от пригожинского. Прежде всего, *мы не предполагаем наличия в хаосе статистических закономерностей,* поскольку уже само введение таких закономерностей неявно протаскивает множество исключительно сильных структур и вместе с ними целое стадо невидимых священных коров (достаточно упомянуть лишь следующие две: измеримость действительными числами, причём при весьма регулярной шкале, допускающей все действия, вплоть до дифференцирования; наличие функционала, характеризующего, скажем, нормальное распределение ошибок). Это в соответствии с духом неформализуемости не исключает использования вероятностных моделей для грубого моделирования и массированного тестирования систем. Но любым тонким результатам, основанным на математическом исследовании таких моделей, нельзя с практической стороны верить ни на грош, пока они не перепроверены совершенно другой моделью.

В качестве примера гибели статистических моделей достаточно рассмотреть знаменитые действия Дж. Сороса по обвалу фунта стерлингов и рубля. Целенаправленная интеллектуальная личность полностью побила статистику (на короткий период, но этого хватило для спекуляций).

Итак, хаос у нас на самом деле – совокупность действующих агентов (живых либо программных), со своими целями и своей внутренней логикой, в которых пытаться разобраться зачастую просто вредно.

Это не примитивный негативизм (когда можно успешно применять стандартные методы управления, действуя от противного): хаотический агент может выполнить действие, может сорвать его и может неожиданно для всех проявить инициативу и выполнить его намного лучше, чем предусмотрено.

Срыв обязательств ещё более или менее учитывается традиционной системой управления, а вот слишком хороший результат либо пропадает в такой системе, либо (в программной) «вешает» её хуже любой ошибки, либо (в административной и фирменной) приводит к изгнанию работника как недисциплинированного.

Наша первая предпосылка — нужно научиться работать в условиях полного хаоса, а дальше видно будет.

Что же мы имеем в ситуации хаоса? Процессы разваливаются, поскольку даже при шансах[6] успеха каждого шага 80%, процесс, распланированный на пять шагов вперед, наверняка сорвется. Поэтому остаётся совокупность действий. Чтобы эта реальная совокупность работала, нужно построить для неё идеальное окружение и конкретизировать некоторые его элементы в качестве управляющих структур.

Традиционное описание действия в современной теории — тройка $\{A\}SB$, где A — предусловие действия, S — само действие, B — его постусловие. Если на входе выполнено предусловие, то после действия с гарантией будет выполнено постусловие.

Подобное описание имело два варианта. В программных логиках предусловия и постусловия рассматривались внутри текста программы, что для нас абсолютно неприемлемо. В динамических логиках эти описания рассматривались как изолированные сущности, по этой причине динамические логики дали ряд интересных математических результатов, но в качестве аппарата для исследования реальных задач использованы не были. Поэтому в качестве аппарата могут быть исследованы динамические логики.

Первый же взгляд на результаты динамических логик показывает, что постановка задачи использования данного класса логик в качестве описания процессов отнюдь не лишена смысла. В них с самого начала исследовались возможности, когда результат не полностью гарантирован (недетерминированные действия), и предусловие зачастую неявно использовалось в качестве критерия возможности применить действие.

Для наших целей формализация действия должна иметь ещё один важнейший компонент. Поскольку в любой момент мы должны иметь возможность осознанного выбора реализуемого действия среди всей совокупности реализуемых в данных момент действий, действия должны снабжаться *приоритетами.* Система приоритетов сразу же ставит следующие три вопроса:

1. Являются ли приоритеты директивными, то есть, обязательно исполняется действие с наивысшим приоритетом, либо рекомендательными, когда действие выбирается с учетом приоритетов, но не обязательно самое приоритетное?

2. Являются приоритеты статическими либо динамическими?

3. Приоритеты линейно упорядочены, либо могут быть несравнимые приоритеты?

Ответим на все три этих вопроса наиболее принципиальным образом.

Первый вопрос. Поскольку хаос состоит из субъектов, при полностью упорядоченном поведении управляющего субъекта (или программы) хаотические индивидуумы могут быстро приспособиться к нему и начать его использовать самым непредсказуемым и изощрённым способом. Поэтому *необходимо сделать приоритеты рекомендательными.*

Второй вопрос. Поскольку важнейшим компонентом описания действия в реальной обстановке является время исполнения, конечно же, приоритеты должны динамически пересчитываться. Далее, это облегчает возможность учета неожиданных хороших сюрпризов.

Третий вопрос. Если система приоритетов будет одномерной, то мы будем захлестнуты текучкой хаоса. Всё время будут исполняться лишь «горящие» действия, и всякая перспектива окажется потерянной. Более того, это означает, что мы не сумеем использовать недостатки хаоса как достоинства. Можно выделить следующие измерения в приоритете действия (пока что весьма грубо): важность действия, ценность действия, необратимость действия, желаемое время исполнения, стратегическая ценность.

Последний компонент приоритета исключительно важен. Стратегические действия должны исполняться практически всегда, хотя их доля может быть разной.

Итак, мы приходим к следующему описанию совокупности действий. Действие определяется структурой

$$\{A\}[P][T]S\,\{B\}, \qquad (1)$$

где A — *условие* действия, P — описание *многомерного приоритета* действия, T **—** *текущий приоритет,* S — само действие, B **—** *обещание* действия.

[6]Здесь и в дальнейшем, когда говорится о вероятности либо шансе, ни в коем случае не имеется в виду математическая или статистическая формализация вероятности; это лишь грубая оценка относительного числа каких-то событий.

Условие становится гораздо более жёстким, чем в программных логиках: это то, без чего действие не может быть реализовано, например, готовность соответствующих данных либо исполнителя. Обещание, зато, становится гипотезой: может быть, это случится, пока что мы именно на это и рассчитываем. Текущий приоритет является двумерным. Он состоит из тактического и стратегического приоритетов действия.

Базовый процесс хаотического управления характеризуется следующей моделью.

Базовая модель хаотического управления

Имеется совокупность действий, описанных в форме

$$\{A\}[P]\boldsymbol{S}\{B\} \quad (2)$$

Имеется качественная оценка ситуации: от рядовой до критической («К нам едет ревизор»).

Имеется база данных о готовности исполнителей и информационных объектов.

Без учёта качественной оценки ситуации пересчитываются приоритеты всех действий, для которых выполнены условия. С учетом качественной оценки ситуации определяется процент стратегических действий (от 50% до 0%).

Недетерминировано выбираются действия и распределяются между исполнителями.

После получения некоторого количества результатов процесс планирования повторяется.

Действие, распределённое для исполнения, но не давшее никакого отклика, вновь включается в потенциально исполнимые после пары шагов планирования либо истечения вероятного интервала его исполнения.

Отметим следующие факторы.

Для организации базового хаотического управления нам нужно описание совокупности действий. Ныне реально легче всего получить описание такой совокупности из описаний процессов, поскольку модель процессного управления слишком въелась в головы большинства людей. Разложив описания процессов на действия и, более того, запросив у нескольких экспертов несколько описаний одного и того же процесса, получаем совокупность действий. Стратегический приоритет может устанавливаться как субъективно авторизованным индивидуумом, так и полуавтоматически, рассматривая действия, встречающиеся во многих описаниях процессов и не относящиеся к числу рутинных.

Далее, вопрос об оптимизации процесса хаотического управления даже не поднимается. Мы сконцентрировались на устойчивости.

Далее, улучшение плавучести в хаотическом море невозможно без детальной системы журналирования и регулярного анализа журнала. Приведём маленький пример. Оценка реальности обещания нужна, и разумнее всего получить её из журнала с записью деятельности исполнителя, которому действие поручается. Но даже если он до сих пор не срывал действие, рассчитывать на то, что он его обязательно выполнит, нельзя: тут-то вас и стукнет хаос!

И, наконец, рассмотрим чрезвычайные ситуации. Обычные чрезвычайные ситуации здесь ликвидированы как класс: просто реальность учитывается на очередном цикле распределения работ. Но остаются два вида весьма неприятных чрезвычайных ситуаций: отмена действия и «Москва неожиданно изменила все формы отчетности и нормы».

Отмена действия решается соединением журналирования с принципом перестраховки: при прочих равных (и даже не очень равных) условиях необратимое действие исполняется лишь тогда, когда другого выхода не остается.

Второе — это уже один из многих творческих вопросов, возникающих здесь. А именно, здесь нужна методика программирования, причем новая. Да и теоретических проблем здесь очень много: в частности, совершенно другая система описания действий и новая система приоритетов, не исследованная в современной информатике.

5. Учет порядка

Чтобы наше решение стало получше, мы ни в коем случае не должны пытаться оптимизировать его (что противоречит живучести), но должны учесть реальность в виде неожиданно прорывающегося через хаос порядка.

Пусть ситуация в некотором количестве случаев предсказуема. Тогда можно было бы пытаться как-то организовывать процессы. Но прямой ввод процессов в данную модель ведет к грубейшему концептуальному противоречию и самоубийству системы.

Тем не менее даже в базовой модели, погруженной в реальное окружение, процессы присутствуют как призраки, в терминах которых реальные люди описывают реальную деятельность.

Чтобы повысить эффективность работы в случае некоторого порядка, можно часть из этих призраков использовать для создания подпорок. Такими подпорками могут стать виртуальные процессы.

Виртуальный процесс — сеть действий глубиной не более 5 действий (реально лучше не планировать более чем на 3 шага вперед). Виртуальный процесс образуется из процесса-призрака вырезанием из него фрагмента, опирающегося на реализуемые в данный момент действия и использующего лишь доступных исполнителей. Действия, включенные в виртуальный процесс, временно получают высокий стратегический приоритет. Никаких условных операторов и циклов в виртуальном процессе нет и не нужно: виртуальный процесс может быть в любое время отменён и заменён другим, что реализует

эффекты от условий и циклов. Зато недетерминированность и распределенность должны присутствовать с самого начала.

Наличие виртуальных процессов позволяет также эффективнее использовать приятные неожиданности: они просто приводят к появлению тех виртуальных процессов, которые без них были бы немыслимы.

В совокупности относительное поведение нашего возможного решения и существующего решения от упорядоченности можно представить следующей диаграммой.

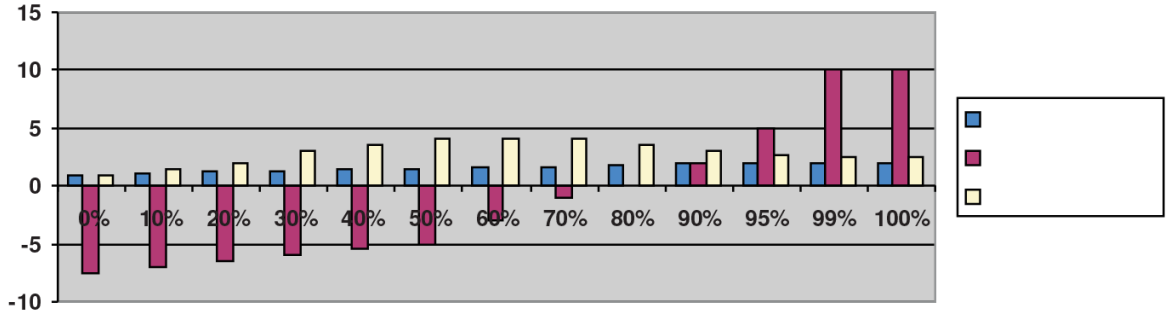

Рис. 1: Качество работы элементарного хаотического решения (синее), виртуальных процессов (желтое) и стандартного метода (фиолетовое)

По горизонтали отложен ожидаемый процент ошибок и исключительных ситуаций, по вертикали — качественная шкала полезности (10 — великолепно работает, 0 — не помогает, но и не мешает, -10 — полностью дезориентирует). Данные о программировании взяты из анализа множества работ по структурному программированию, метрике программ, экстремальному и индустриальному программированию. Практически нигде там нет этих данных в прямом виде, они берутся лишь из косвенных советов и наблюдений. В частности, интересно, что при начале работ по структурному программированию, после осознания концепции исключительной ситуации, работоспособной считалась лишь система, в которой исключительных ситуаций заведомо меньше 5%, а объектно-ориентированный подход позволил поднять эту планку до 10%. Но в данном случае брались заведомо оптимистические оценки. Для нашего же решения брались самые пессимистические оценки.

Наше решение лишь удовлетворительно для систем, близких к упорядоченным, но зато оно весьма устойчиво по отношению к хаосу. Ухудшение ситуации его не разваливает, а улучшение несколько улучшает. Учёт порядка позволяет улучшить производительность на самом критическом интервале — 20-80% штатных ситуаций.

6. Индустриальное и экстремальное программирование: краткая характеристика

В современной информатике имеется два принципиально различных подхода к формализации и программной поддержке сложных задач.

6.1. Индустриальное программирование (ИП)

При этом подходе заказчик работы рассматривается как информационный аналог покупателя либо, скорее, заказчика в ремесленной мастерской.

Заказчик формулирует задачу, уточняет совместно со специалистами ее постановку и принимает промежуточные и окончательные результаты работы. Заказчик представляет, что ему хотелось бы, но некомпетентен в вопросах, как этого лучше достичь (но в худшем для исполнителя случае считает себя компетентным). Исполнитель вынужден сам анализировать и содержательную область, и средства реализации. Успешная работа возможна, лишь *если будет достигнута взаимная компенсация слабейших сторон исполнителя и заказчика.*

Ввиду некомпетентности заказчика (потребителя), типичные методики индустриального программирования обращают внимание на следующие вопросы, в порядке их приоритета:

a) Внешнее оформление продукта. Именно в данном вопросе удается добиться наиболее прямого и быстрого контакта с заказчиком и частично использовать получаемые сведения для решения других задач.

b) Потребительские качества продукта. Удобство использования и понятность стоят здесь на первом месте.

c) Содержательное наполнение продукта. Поскольку заказчик предполагается полностью некомпетентным в данном вопросе, эту часть исполнители формируют сами с учетом того, чтобы она не вредила первым двум пунктам.

6.2. Экстремальное программирование (XP)

При этом подходе заказчик рассматривается как коллега, заинтересованный прежде всего в содержательных результатах работы. Он, безусловно, предполагается компетентным в рассматриваемом вопросе в гораздо большей мере, чем специалисты, реализующие программную систему. Но специалисты более компетентны в вопросах реализации и свойств информационной среды, используемой для этой

реализации. Таким образом, качественное решение может получиться лишь *соединением сильнейших сторон заказчика и исполнителей.*

Этот подход подобен работе инженеров с заказчиком нового прибора.

При экстремальном программировании, в порядке приоритета, рассматриваются следующие вопросы.

a) Как выразить то, что нужно заказчику, в точных терминах, понятных как заказчику, так и исполнителям.

Здесь возникает взаимно приемлемая спецификация будущей системы. Как показывает опыт, для успешной реализации системы отнюдь не является препятствием то, что одни и те же предложения спецификации заказчик и исполнители понимают по-разному ввиду разницы точек зрения. Это даже помогает для следующей задачи.

b) Строится совокупность примеров, на которых должна будет проверяться будущая система (тестов).

Набор тестов все время пересматривается и дополняется в ходе реализации. В первую очередь тесты проверяют требования, где выявилась разница точек зрения заказчика и исполнителей, чтобы удостовериться, что разные пути ведут к одной и той же цели.

c) Строится архитектура будущей системы. Это единственная внутренняя задача исполнителей.

d) Прорабатываются вопросы внешнего оформления и удобства использования.

При этом главное внимание обращается на то, чтобы внешняя красота не испортила достигнутых содержательных результатов.

Данная характеризация индустриального и экстремального программирования обсуждена в январе 2006 г. на трех семинарах с участием ведущих специалистов (Москва и конференция в г. Переславль) и признана новой и более адекватной, чем имевшиеся до сих пор в литературе. При обсуждении было обращено внимание на то, что данная характеризация полностью абстрагировалась от случайных конкретных деталей реализации того или иного ведущего в данный момент способа реализации ИП либо XP и выделила именно основные черты двух важнейших подходов.

7. Применение к анализу административной деятельности

7.1. Общий анализ

Как известно, в последнее время проблема автоматизированной поддержки административной деятельности, в частности, электронных административных регламентов (ЭАР) является одной из актуальных, и к ее решению ищут подходы во всем мире. В данной работе представлены результаты системного и логического анализа существующего положения вещей и возможных направлений исследований.

Прежде всего, стоит заметить, что данная область является исключительно сложной с концептуальной точки зрения, а вдобавок такой, где легкомысленное отношение к задаче как к чисто технической немедленно и беспощадно мстит за себя. Она представляет собой классический пример одного из основных видов человеческой деятельности последних десятилетий: формализации неформализуемых понятий.

Чиновник не может действовать лишь по инструкции, а если даже он действует точно по инструкции, то в условиях России ситуация на практике оказывается абсолютно невыносимой. Административная деятельность активно сопротивляется формализации, и вместе с тем ее формализация необходима. Таким образом, в частности, нужно четко осознать вред иллюзии примитивного позитивного мышления и постановки как реальных задач идеальных целей, достижимых только в принципе. Вновь напоминаем, что в принципе возможно необходимо понимать как на практике невозможно, а чаще всего даже вредно. Подгонять человека под принцип — основная ошибка идеологизированного мышления (все равно, либералов или коммунистов). Нужно приспосабливать принципы к реальным людям. Так что эта работа написана с точки зрения консерваторов.

Рассмотрим показательный пример. В мировых работах по формализации административной деятельности по умолчанию предполагается, что эта деятельность является бизнес-процессом. Использование термина «процесс» критическому анализу не подвергается вообще, принимается как данное по умолчанию. Опыт работы со сложными неформализуемыми понятиями и анализ возникающих в реальности проблем показывает, что именно такие легкие и необсуждаемые умолчания таят в себе зародыш многих бед.

Итак, первый вопрос: можно ли применять к административной деятельности понятие процесса? И второй вопрос, неразрывно связанный с этим. На самом деле происходит ещё одна подмена: общее понятие процесса заменяется на привычную программистам его реализацию. Анализ всех применяемых инструментов показывает именно это, более того, этой реализацией является понятие программного бизнес-процесса, которое настолько близко во всех имеющихся программных реализациях, что практически можно считать его фиксированным. Даже если после анализа ответ на первый вопрос окажется благоприятным, то остаётся второй.

Итак, первый вопрос: является ли административная деятельность процессом? Отвечая на него, можно увидеть, что административная деятельность имеет два полюса.

На одном полюсе находится та, где практически не возникает проблем при формализации её нынешними методами, когда административный аппарат является чем-то типа машины для реализации запросов и ведения учётных баз. Примером такой деятельности являются почти все функции отделов ЗАГС. Характеристиками такой деятельности являются:

a) Упорядоченность и хорошая регламентированность. Дыр и противоречий в нормативных документах мало, и они на практике встречаются редко.

b) Наличие в любом административном акте либо всего одного гражданина, либо совокупности граждан, объединенных общей целью. Под гражданином здесь понимается человек, обратившийся в административную систему не как её член. Скажем, в акте регистрации брака имеются несколько человек (пара и свидетели), интересы которых почти всегда общие: как можно быстрее отпраздновать свадьбу. Таким образом, в этих областях деятельности внешние взаимоотношения граждан друг с другом либо взаимоотношения граждан с чиновниками играют пренебрежимо малую роль.

c) Отсутствие в решениях волеизъявления чиновников.

Но, как отмечается в отчете московской группы [11], наиболее критическими являются другие области, скажем, землеустройство, характеризующиеся следующими признаками:

a) Неупорядоченность и пере- либо недорегламентированность. Дыр и противоречий в нормативных документах много, и они на практике встречаются на каждом шагу.

b) Наличие во многих административных актах взаимодействия граждан, имеющих противоречащие друг другу интересы. Скажем, в акте отвода земельного участка под строительство часто имеются несколько лиц, претендующих на данный участок, и их интересы конфликтны. Таким образом, в этих областях деятельности внешние взаимоотношения граждан друг с другом либо взаимоотношения граждан с чиновниками играют исключительно важную роль.

c) Присутствие в решениях волеизъявления чиновников.

Видно, что решения, апробированные в мировой практике только в первом случае, пытаются перенести на в корне от него отличный второй. Этим печальным выводом можно завершить первую стадию анализа.

Теперь перейдем к анализу самого понятия процесса, используемого в формализации административных систем.

Коренные различия административного процесса и бизнес-процесса легко усмотреть, скажем, на примере анализа общепринятых в мировой практике принципов оценки систем поддержки бизнес-процессов с точки зрения административных процессов. Выделяются следующие пять принципов [11].

1. «Системность описания» — сочетание методов структурного, функционального и процессного моделирования бизнеса;

2. Открытость для описания новых знаний о моделируемой бизнес-системе;

3. Скорость моделирования и внесения изменений («хоть три раза в день») — технология не должна сдерживать изменения;

4. Выразительность и наглядность результатов (обеспечение взаимопонимания при командной работе — язык общения управленческого звена компании);

5. Генерация документов в общепринятых (мировых и национальных) стандартах.

Анализ данного списка выявил еще раз правильность поговорки о “благих намерениях, которыми вымощена дорога в ...”. Проанализируем различные его пункты.

Казалось бы, пункт 1 не может вызывать никаких возражений. Однако известно, что система является не столько объективным, сколько субъективным понятием, она выделяется на базе некоторой системы предпочтений, и в данном случае неявно заложено предпочтение функционального и процессного описания, а не описания, скажем, на базе ограничений и условий. Даже не анализируется вопрос о том, какой вид описания систем лучше именно для административных систем. Таким образом, этот пункт сформулирован неудовлетворительно, поскольку создает впечатление безальтернативности в случае наличия принципиально важной альтернативы. Его анализ позволяет выделить важную проблему, которая рассматривается в дальнейшем.

Пункт 2, казалось бы, ещё более очевиден. Но, принимая во внимание, что процесс формализации административной системы является сложной смесью описания уже устоявшегося и административного давления с целью изменения некоторых аспектов устоявшегося, мы приходим к выводу, что в данном случае порою нужна не столько открытость, сколько закрытость. Если некто утверждает, что получил

новое знание об административном процессе, это новое знание нужно прежде всего верифицировать на соответствие всей системе, а лишь затем включать в систему. В этом состоит позитивная сторона нередко раздражающего людей процесса согласования административных решений.

Пункт 3 *полностью противоречит сути административной системы.* Представляете себе кошмар гражданина, взаимодействующего с модифицируемой несколько раз на дню административной системой?! Во многих случаях достаточно изменений раз в полгода-год (например, наш институт столкнулся с этой проблемой при регистрации диссертационного совета: когда очередь до рассмотрения заявки провинциалов доходила, оказывалось каждый раз, что Москва уже изменила формы или порядок и нужно подавать заявку заново). А в случае системы образования изменения раз в несколько лет приводят к ситуации “инновационного регресса”. Оценить итог нововведений можно лишь через 10-20 лет. Тут необходим механизм верификации изменений на предмет согласованности с другими частями системы и тщательной их проверки перед введением в исполнение, и, таким образом, совершенно другая система критериев оценки.

Пункт 4 — единственный, который не вызывает серьезных возражений.

Пункт 5, конечно же, выглядит полностью адекватным для административной системы. Но если нечто выглядит слишком хорошим, надо обязательно посмотреть, а нет ли здесь ловушки Дьявола. Она зияет и в данном месте.

В самом деле, коренной разницей бизнес-процесса фирмы и административного процесса учреждения является разница между менеджером и чиновником. Менеджер (или работник, частично исполняющий функции менеджера в некоторых случаях) по определению уполномочен принимать некоторые решения. Первичным источником легитимности решения является то, что оно исходит от уполномоченного на это лица, а *дополнительным ограничением* является проверка его соответствия нормативным актам. Таким образом, в случае бизнес-учреждения действительно исключительно важно обеспечить корректное **ОФОРМЛЕНИЕ** решений уполномоченных лиц, то есть порождение соответствующих легитимных документов.

Но в случае административного учреждения чиновник с функциями менеджера прежде всего является проводником процессов применения законов и административных регламентов. Слишком часто решение, которое он подписывает, является не результатом его собственного волеизъявления, а результатом того, что по закону он обязан совершить данное действие. Таким образом, *оформленный по всем правилам документ, основанный на ошибочных предположениях, является незаконным.* Документ в данном случае является не результатом работы, а результатом взаимодействия процесса волеизъявления лица, запросившего данный документ, и проверки легитимности и обоснованности его запроса со стороны административных органов.

Таким образом, здесь напрашивается переход от описания административного процесса как процесса производства документов к описанию его как процесса ведения и проверки реестров и запросов к реестрам. Например, кредитная карточка сама по себе документом не является, если её идентификационные данные не содержатся в реестре соответствующего банка и если у Вас на счету нет запрашиваемой суммы. Переход от системы документооборота к системе ведения и взаимодействия реестров, видимо, неизбежен при внедрении электронных документов, и его элементы уже начали появляться. Он заодно является способом решить проблему бесконечных справок в традиционном документообороте.

Первой ласточкой в данном направлении можно было считать отмену обязательного представления налоговых деклараций многими категориями граждан. Наличие в фискальных органах убедительной совокупности зарегистрированных документов об уплате налогов по месту основной работы и дополнительных заработков верифицирует гражданина как исправного налогоплательщика и сокращает документооборот.

Заметим, что понятие документа сохраняется в некоторых исключительно важных случаях. Во-первых, документ может не быть частью никакого реестра (как, например, завещание). Во-вторых, реестр административных документов не заменяет самих этих документов. Нормативный акт является результатом волеизъявления уполномоченного на это лица или органа. Он может быть верифицирован, но не порождён. Для документов в собственном смысле этого слова реестры могут быть частью процесса верификации документа (например, нотариально заверенное завещание должно сопровождаться соответствующей записью в реестре нотариуса об акте его составления и заверения). И вообще, процесс верификации данных, документов и далее процессов является важнейшим в данной области, а в бизнес-процессах он носит вспомогательную роль.

Далее, в приведенном перечне не указаны важнейшие для административной системы критерии полного учета административных ограничений и выявления как тех случаев, когда чиновник должен принять решение сам, так и тех, когда его деятельность регламентирована нормативными документами. И уж полностью игнорируются реальные случаи, когда любое из действий чиновника противоречит некоторым из документов, поскольку они не согласованы, а согласовать их возможно лишь на других уровнях иерархии или даже лишь при участии других ветвей власти (например, Думы и Президента).

Таким образом, первая же принимаемая по умолчанию аксиома подавляющего числа работ по электронным административным регламентам оказывается в высшей степени сомнительной. Но, тем не менее, в соответствии с духом теории и практики работы с неформализуемыми понятиями, это отнюдь не исключает использования систем поддержки бизнес-процессов в данной области, требуется лишь

понимать, что это решение будет временным (впрочем, сам характер данной области как неформализуемой делает таковым любое решение; как только решение становится постоянным, оно начинает постепенно увеличивать приносимый вред, а польза постепенно становится мнимой).

Теперь перейдём ко второй аксиоме, принимаемой в большинстве работ по административным системам: нужно пользоваться хорошо разработанными методами инженерии знаний и промышленного программирования. Чтобы увидеть вред данного положения, достаточно рассмотреть процесс внедрения системы формализации административной деятельности. Это внедрение обречено на неудачу, если нет энтузиастов внутри учреждения, деятельность которого формализуется. Таким образом, в данном случае административный работник должен быть не пассивным потребителем, которого интервьюируют высоколобые специалисты и затем расписывают результат в эффектно выглядящей системе, а полноправным соучастником работ. Возвращаясь к информатике, мы видим, что в данном случае немедленно переходят из области, где эффективно индустриальное программирование, в область, где эффективно экстремальное. Но в чистом виде экстремальное программирование, так, как оно описано в распространённых руководствах, неприменимо к административным процессам. В экстремальном программировании система всё время меняется и тестируется, а в административной деятельности, как уже было обосновано, необходимо соблюдать разумный консерватизм, и, конечно же, абсолютно недопустимо превращать чиновников и граждан в подопытных собачек. Поэтому необходим дополнительный анализ проблемы.

7.2. Анализ возможности описания административных систем с помощью средств индустриального программирования

Главная характеристика индустриального программирования, впервые четко сформулированная в книге [5] и поддерживаемая в неофициальных разговорах ведущими мировыми экспертами в данной области (явно они стыдятся это говорить, и данное положение не звучит в популярных книгах и статьях) следующая.

Индустриальное программирование ориентировано на задачи, где внешнее оформление (упаковка) важнее содержания. Его бурное развитие связано с проникновением на рынок информационных технологий критериев потребительского общества, где зачастую упаковка и реклама составляют более 50% стоимости товара в ущерб его реальным потребительским достоинствам. Ущерб от неадекватности ИП задачам ЭАР можно оценить как реальную опасность полной потери самой идеи ЭАР и подменой ее несколькими красиво упакованными стандартными задачами (база данных, документооборот, финансовый учет).

Автору пришлось проанализировать ряд методик информатизации административных учреждений, в том числе и во время непосредственной работы при помощи дочерней фирме Интерин нашего ИПС РАН. К несчастью, все рассмотренные методики носят на себе явный отпечаток индустриального программирования, что связано с саморекламой индустриального программирования как универсального метода[7].

Второй причиной гегемонии индустриального подхода является взгляд на чиновников как на потребителей и исполнителей (а, скажем, типичный менеджер управляет столь сложно организованными и сложно протекающими процессами, с которыми нынешние программисты совладать не могут).

Более того, наиболее детально проработанная и качественная (на 2007 год) открыто опубликованная инструкция по разработке программ электронного правительства ([15], Великобритания) целиком находится в этом же русле.

И одна из опасностей такого подхода в случае медицинских систем была выявлена в ходе наших работ. Как выяснилось после анализа пари помощи нейронных сетей, 98% решений обычных врачей являются результатом механического применения инструкций и “стандартов”, и, как следствие, интеллектуальные системы помощи им просто не нужны. Нужна хорошо организованная и постоянно поддерживаемая адекватно база нормативных документов.

7.3. Анализ возможности описания с помощью средств экстремального программирования

Основная предпосылка экстремального подхода — на первом месте стоит содержание системы. Тем самым он является естественной альтернативой и естественной противоположностью индустриальному.

Поскольку в задачах формализации и поддержки административной деятельности на первом месте стоит ответственность за результаты и за способы их достижения, видно, что данная задача по основной идее гораздо ближе к экстремальному программированию, и, во всяком случае, решать её методами индустриального программирования — грубейшая концептуальная ошибка.

С другой стороны, прямой перенос методик и технологий экстремального программирования на данную область невозможен: здесь нельзя прямо обеспечить непрерывное тестирование системы в ходе создания и изменения.

Анализ существующих методик экстремального программирования (как описанных в книге [10], так и тех, с которыми пришлось столкнуться в практике научного консультирования ведущих информационных фирм в случаях, когда традиционные методы не работали), можно сделать вывод, что непрерывное

[7]Надо всегда учитывать, что ни один успешный метод не может быть универсальным, и такая реклама должна рассматриваться либо как намеренное введение в заблуждение, либо как самообман рекламирующих.

тестирование является в некотором смысле священной коровой XP: модулем методики экстремального программирования, который может быть заменен без ущерба большинству других идей[8]. Тем самым возникает технологическая задача создания варианта XP, ориентированного, в частности, на исполняемые людьми процессы. Ущерб от неадекватности нынешнего варианта XP можно оценить как некоторое снижение скорости и эффективность решения задачи программной поддержки административной деятельности, не наносящее ущерба основным характеристикам.

Впрочем, необходимость приспособления к конкретной области является одной из основных черт применения XP. В частности, *нельзя использовать* XP *в случае, если заказчик не готов сотрудничать с программистом.* Переоценка заказчика, который на самом деле является потребителем или же считает себя настолько квалифицированным, что указывает все мелочи и требует мелочных отчетов (это другая сторона потребительства) является гибельной ошибкой.

7.4. Выбор методики

Таким образом, при выборе метода реализации программной поддержки административной системы лучше ориентироваться на экстремальное программирование и избегать реализаций и реализаторов, работающих методами индустриального.

Эти соображения дополняются следующими. Поскольку в случае административной системы заказчики, безусловно, гораздо более компетентны, чем эксперты- информатики, в вопросах конкретики административных процессов, единственным приемлемым способом работы для успешной разработки и внедрения хорошей программной системы поддержки деятельности является экстремальное программирование.

Более того, именно экстремальное программирование является единственной возможностью обеспечить не только успешную разработку, но и успешное внедрение, поскольку внедрение сложной информационной системы в существующую сложную организационную структуру возможно лишь при наличии энтузиастов внутри данной структуры, а чиновники, привлечённые как эксперты, будут рассматривать получившуюся систему как родную (что оправданно, поскольку они вложили в нее свои знания и опыт) и всячески помогать её продвижению.

7.5. Хаотический процесс в административной системе

И, наконец, перейдём к самой первой из выявленных проблем: а насколько адекватен сам подход, основанный на описании функций и процессов?

В случае современных систем моделирования бизнес-процессов процесс представляется как аналог компьютерной программы. Это означает, что, с точностью до выбора вариантов, он жёстко фиксирован. Именно в данной связи исключительную важность приобретает требование возможности его быстрого перепрограммирования. Но, поскольку в случае административного процесса быстрое перепрограммирование исключено, становится весьма сомнительной обоснованность самого процессного подхода.

Более того, в случае русских реальных исполнителей (помните, что в принципе хорошие решения нами отбрасываются с самого начала; лучше реально удовлетворительное, чем в принципе отличное) всё описание процесса будет состоять из массы обработки исключительных состояний и сбоев, и станет совершенно неработоспособным для сколько-нибудь вариативной системы. Оно годится лишь для работ, где вся деятельность однозначно расписана существующими нормативными документами (например, упомянутые выше многие функции ЗАГСов или, к несчастью, деятельность рядовых врачей). Тем не менее как временное и ограниченное решение такие системы могут применяться в некоторых областях, три основных характеристики которых описаны выше. Если же представить себе гипотетически полную регламентацию работы, то сразу же возникнут следующие факторы. Во-первых, эту систему будут целенаправленно взрывать и разрушать изнутри и снаружи (недовольные чиновники и взбешённые граждане). Во- вторых, в России всегда всё лучшее делалось не по воле верхов, а по собственной инициативе русских людей. Царь, скажем, приказывал вернуть Ермака и не ставить фантастическую задачу завоевания Сибири, а ограничиться реальной задачей охраны Руси от набегов сибиряков. И таких примеров несть числа. Так что если даже удастся сломать оппозицию, то «русский порядок» обязательно превратится в полный кошмар, а все лучшие качества русского народа будут беспощадно подавляться в угоду абстрактным принципам.

Возникает вопрос, а можно ли упорядоченно работать в столь хаотичной и недисциплинированной среде?

Как уже было обосновано ранее, ответ: «Да!» Можно, если перейти от процессов к запросам и ограничениям. Тогда события исчезают, поскольку просто идет поток пересмотра запросов, а действия планируются в соответствии с множеством запросов. Но это подход, противоположный заложенному в современные системы бизнес-логики.

8. Применение хаотического управления к административной системе

Рассмотрим программную модель административного процесса, ориентированную на работу в реально совершенно хаотической обстановке.

---

[8]И даже с пользой для них, поскольку некоторые весьма квалифицированные специалисты убегали из фирм, пытавшихся «применять XP», не выдержав психологического прессинга ежедневного тотального тестирования.

Обстановка описывается как совокупность состояний элементов. Например, состояние документа может быть «находится на экспертизе в экономическом отделе», состояние чиновника может быть «в полной прострации, способен лишь на такие действия, которые не допускают вариантов» либо «отсутствует по неизвестной причине и недоступен по телефону.» Состояния граждан в описание обстановки не входят, поскольку граждане не являются частью административной системы. Критическое действие программной либо гибридной (человек-компьютер) системы состоит в определении по данному состоянию возможных в данном состоянии действий, затем оценке их приоритетов и выбора реально реализуемых приоритетных действий. Таким образом, в 'реале' полностью отсутствует планирование на несколько шагов. Зато в нём полностью отсутствуют исключительные ситуации. Любое событие принимается к сведению и вызывает небольшую перепланировку.

Видно, что уже в данной модели полностью хаотического состояния возникает некоторая структура. А именно, события естественно делить на рядовые, текущие и важные.

Рядовое событие — при котором не требуется внеочередной запуск планировщика. Например, это обращение гражданина за рутинным документом или нормальное завершение очередного пункта работы одним из исполнителей.

Текущее событие — при котором требуется локальная работа планировщика. В частности, это истечение некоторого интервала времени, после чего нужно выдавать новые задания освободившимся либо появившимся сотрудникам, либо неожиданный уход Марьи Ивановны лечить заболевшую кошку.

Важное событие — при котором изменяются приоритеты и требуется полный внеочередной цикл перепланировки. Например, «К нам едет ревизор.»

Таким образом, в данной системе обработка исключительных ситуаций ликвидирована как класс. Причиной этого является то, что мы не пытаемся заглянуть далеко вперёд, а затем лихорадочно поправлять наши планы, поскольку «гладко было на бумаге, да забыли про овраги.»

Необходимо повторно заметить, что такая система неразрывно связана с журналированием действий. В частности, обязательно должны быть журналированы действия, связанные с волеизъявлением чиновника либо гражданина. Журналирование дает возможность осуществлять отмену действий в очень многих случаях. Отмена является опять-таки достаточно рядовым фактом в подобной системе. И лишь отмену можно рассматривать как некоторый аналог исключительных ситуаций.

В связи с отменой необходимо сразу же разделить обратимые и необратимые действия. Необратимые действия нельзя исполнять до тех пор, пока не исполнены все связанные с ними обратимые, хотя бы формально все условия для необратимого действия уже существовали *(принцип перестраховки).* Этот принцип дает возможность более гладко осуществлять отмену действий.

Далее, принцип перестраховки разделяется на две принципиально разных реализации. Если необратимое действие приводит к исчезновению состояний, связанных с внешним запросом, оно должно быть сделано при первой же возможности *(подписано, и с плеч долой!)* Если же оно приводит к изменению состояния самой системы, его реализацию нужно тянуть до последней возможности *(принцип волокиты),* поскольку слишком часто реорганизация разрушает уже налаженную работу.

Виртуальный процесс рассматривается системой планирования как совет, что же делать, чтобы быстрее избавиться от некоторых состояний, дойдя до конечного результата, к которому можно применить принцип *с плеч долой.* Виртуальный процесс может быть в любой момент работы планировщика пересмотрен (поэтому в нём не нужны альтернативы).

9. Модель административной системы как распределенной асинхронной программной системы

9.1. Понятие распределенной системы

Чтобы воспользоваться понятиями информатики для моделирования и анализа административных процессов, необходимо вначале описать сам административный процесс с точки зрения информатики. Конечно же, такое описание можно проделать многими способами: мультиагентные системы, модель потоков данных, распределенные системы. Каждый из этих способов имеет свои достоинства и недостатки.

В мультиагентных системах процесс моделируется как совокупность действий активных объектов (агентов), каждый из которых имеет свои цели и достигает их, взаимодействуя с другими агентами. Одним из способов такого взаимодействия является сотрудничество, а другим — конкуренция.

Несмотря на то, что на первый взгляд такая модель привлекательна, она имеет коренной недостаток. Основная задача административной системы состоит отнюдь не в согласовании интересов отдельных чиновников или чиновников и граждан. Если такая задача возникает, то мы констатируем неудовлетворительное функционирование такой системы. Таким образом, данная модель, широко используемая при моделировании бизнес-процессов или общественных процессов, годится для наших целей лишь для анализа неудовлетворительно работающих подсистем, что не является основной задачей.

В системе потоков данных действия планируются на базе данных. Считается, что действие порождается тогда, когда для него готовы входные данные, и завершается выдачей данных для других действий. Формализмом, наиболее широко используемым в мире для данного подхода, являются сети Петри. Он часто используется для анализа систем документооборота.

Такая модель соответствует логике по крайней мере одного из модулей административной системы и может быть применена к ее компонентам. Но она не учитывает активности лиц, вовлеченных в административный процесс, и плохо справляется с исключительными ситуациями. Таким образом, и данная модель плохо подходит в качестве базисной логики формализации административной деятельности.

Если разобраться с основным содержанием т. н. «электронных административных регламентов» (ЭАР), то на высшем уровне ЭАР менеджера описывает так называемый распределенный процесс. В самом деле, исполнители (особенно у нас в России) ненадежны, работают они одновременно, коммуникации между ними плохо проработаны.

Констатировав эти недостатки, мы должны постараться превратить их в достоинства.

Одним и способов описания распределенных систем, шире всего применяемым в мировой практике для описания бизнес-процессов, является п-исчисление Р. Милнера (см. [12, 13, 14]). Базовые конструкции исчисления Милнера следующие.

- Параллелизм, обозначаемый $P|Q$, где P и Q — два процесса.
- Сообщения:
  - Входной сигнал $c(x)$. *P* означает, что процесс ожидает сообщения, посылаемого по каналу с, прежде чем выполнить P, а получаемое сообщение называется x. Это соответствует, скажем, ожиданию документа с визой перед выполнением некоторого этапа действия.
  - Выходной сигнал $[y]\ P$ означает, что сообщение с именем у посылается по каналу *c* перед тем, как исполнить *P*. Это может быть передача документа на визу или запрос дополнительных данных.
- Дублирование процесса $!P$, порождающее новую копию P. Это означает, например, поручение работ над независимыми частями документа нескольким сотрудникам.
- Создание нового имени $(vx)P$, скажем, создание нового документа или запроса.
- Пустой процесс 0, соответствующий концу работы.

Поразительно, что столь простой набор конструкций (полностью приведенный выше) позволяет ***в принципе*** описывать сколь угодно сложные системы параллельного исполнения. Но здесь уместно вспомнить принципы взаимодействия теоретика и практика из статьи [7], один из которых — ***«в принципе возможно»*** теоретика практик должен переводить как «***практически невозможно».***

В частности, именно на системе Милнера базируется BPML [17], один из самых распространенных языков моделирования бизнес-процессов.

Примером простого описания фрагмента регламента на расширенном для бизнес-приложений исчислении Милнера может служить такой.

Рассмотрим фрагмент ЭАР подготовки ЭАР ЗАГСа.

10.4. ЕСЛИ регламент требует полной переделки, ТО инициировать процесс изменения регламента согласно инструкции 125 от 15 июня 2001 г.
Исполнитель: руководитель

10.5. ЕСЛИ регламент не требует полной переделки, ТО назначить ответственного за корректировку каждого параграфа.
Срок: 1 день.
Исполнитель: руководитель

10.6. Поручить юристу оказывать исполнителям необходимые юридические консультации в процессе корректировки параграфов.
Исполнитель: руководитель

10.7. Все назначенные исполнители выполняют корректировку соответствующих пунктов.
Срок: 2 недели
Исполнители: лица, назначенные руководителем за корректировку соответствующих параграфов,
Консультант: юрист

10.8. Провести рабочее совещание.
Время: через неделю после начала действия п. 7 Исполнители: руководитель
Участники: руководитель, его заместитель, юрист и все исполнители, которым поручена корректировка параграфов

10.9. Провести заключительное совещание.
Время: через 2 недели после начала действия п. 7 Участники: те же, что в п. 8

10.10. Предварительно утвердить изменённые параграфы, поручить окончательно оформить текст откорректированного регламента.
Срок: 1 день
Исполнитель: руководитель

Запишем его в виде π-выражения.

$$\left\{\begin{array}{l}?\ /text\_();\\ ?\ —\ \text{полная\_переделка}\ |\_\text{назначить\_совещание1\_}|\\ \qquad (v\ \text{список})\forall\text{исполнитель}\ \in\ \text{список}\\ !\ \text{начало\_исполнения}\\ \ [\text{готов0}](\text{совещание}\ 1)\\ \text{завершение\_исполнения}\\ \ [\text{готов}]\ (\text{совещание2})()\\ |\ \varepsilon\ \{\text{Назначить\_совещание}\ 1\ \text{ждать\_неделю}\\ (\text{совещание}\ 1)\ \text{совещание\_1 ждать\_неделю}\\ (\text{совещание})\ \text{совещание утверждение}()\}\end{array}\right\}$$

Уже из данного примера видно, что исчисление Милнера как математический формализм не совсем адекватно в отношении времен исполнения и трудно обрабатывает ситуации с неопределенным количеством исполнителей. Его практические варианты до некоторой степени преодолевают данные недостатки и дают возможность программе анализировать времена исполнения процессов. Поэтому в данном случае этап математической формализации можно полностью исключить и переходить от формализованного технического задания непосредственно к поддержанному программой тексту. В данном случае проявляется одно из преимуществ методик, основанных на современных высокоуровневых средствах и технологиях информатики: возможность исключения явных математических моделей нечисленных задач.

Более того, такие описания дают возможность воспользоваться стандартными методами анализа и оптимизации процессов для исчисления Милнера, что приводит к возможности применения программного описания не только для моделирования, но и для анализа административной деятельности.

В работе одного из ведущих сотрудников фирмы IBM Уайта [16] предложен достаточно полный набор шаблонов для реализации милнероподобной системы

взаимодействия в рамках стандартных технологий индустриального либо экстремального программирования, без использования специализированных языков. Это дает возможность переводить теоретические описания в эскизы программ на стандартных языках.

Необходимо заметить, в частности, что в работе [2] высказывается и обосновывается мнение: милнеровская основа практических систем моделирования бизнес-процессов всего лишь маркетинговый ход, на самом деле они никакого отношения к данной математической теории не имеют (кроме внешнего сходства некоторых обозначений и терминологии). Автор по мере знакомства с многочисленными разработками в данной области постепенно и сам приближается к подобной оценке, хотя еще не столь уверен в этом. Подобные прецеденты известны и многочисленны в других областях.

9.2. Частично-упорядоченное время, проблемы взаимодействия, синхронизации, эффективности и надежности

В работе Милнера и в большинстве работ, основанных на п-исчислении, практически полностью игнорируются те основные характеристики распределенного процесса, которые выделены нами в начале и которые актуальны для административной работы в условиях России.

Надо сказать, что ясному осознанию перечисленных выше факторов сильно мешает общепризнанный в мире позитивный подход, который, декларируя прогресс на словах, на самом деле стал тормозом для него. Одно из правил творческого мышления: превратить вред в пользу. Поскольку процессы все равно нельзя распланировать заранее, нельзя ли их порождать «на ходу»? В данном случае оказывается, что можно, если порождаемые процессы являются автоматными по существу. Это соответствует математической модели автоматного программирования высшего уровня.

Сразу же заметим, что в данном пункте мы вышли на постановку фундаментальной задачи, для которой вообще нет задела в мировой практике, так что в данный момент даже задумываться о возможности немедленного приложения данного подхода невозможно. К приложениям подготовлены подходы, базирующиеся на описанных выше милнероподобных системах и, возможно, UMS.

Цель данного параграфа — показать, насколько серьезны задачи, встающие при серьезном подходе к поставленной проблеме. Есть шансы, что по мере развития работ можно будет поставить цель достичь принципиально нового качества работ: динамического регламента, работающего в системе реальных исполнителей и реальных граждан. Более того, имеется ощущение (безусловно, требующее дополнительной проверки), что именно на данном пути удастся перейти от моделей. основанных на процессах, к моделям, основанным на обязательствах. Но, ввиду весьма начальной стадии исследований, изложение предельно сжатое и даже фрагментарное (поскольку формализм может быть существенно переработан в ходе дальнейших исследований).

Рассмотрим следующую модель распределенных вычислений.

Пространство моментов времени рассматривается как частично-упорядоченное множество, имеющее наименьший и наибольший элементы (начальный и конечный момент времени исполнения). *Нить* — линейно упорядоченное подмножество моментов времени, имеющее наибольший и наименьший элементы, и такое, что любой момент, сравнимый со всеми точками нити, входит в нить. *Базис нитей* —

такое множество нитей, пересекающихся друг с другом лишь, может быть, по начальным и/или конечным элементам, что все моменты времени входят хотя бы в одну нить.

Считается, что в множестве моментов (которое предполагается большим по мощности) можно выделить небольшой базис нитей. Для обычного процесса число нитей в этом базисе может быть порядка десяти. Фиксируем некоторый базис и нити из него будем называть просто нитями (другие нити нигде ниже не рассматриваются). Будем считать, что одна и только одна из нитей базиса включает начальный и конечный моменты времени. Она называется *главной нитью.*

Задано некоторое частично упорядоченное множество приоритетов. В его качестве достаточно взять множество всех конечных подпоследовательностей последовательностей рациональных чисел, упорядоченное отношением: область определенности меньшей последовательности вложена в область определенности второй и на общем отрезке все элементы первой последовательности меньше всех элементов второй последовательности.

Имеется набор объектов. У каждого объекта имеется набор методов, которые считаются элементарными действиями (не в том смысле, что они неделимы, а в том смысле, что нас не интересует их внутреннее устройство). Среди этих действий выделяются создание (нахождение), активация, деактивация, уничтожение, копирование, синхронизация копии с оригиналом, изменение приоритета. Каждое элементарное действие может быть приписано одной или нескольким нитям. С внешней стороны элементарное действие описывается как система состояний (конечный автомат). Элементарное действие может быть специфицировано как пассивное, активное, выполняющееся, приостановленное, прерванное, завершенное, синхронное, асинхронное, локальное, распределенное.

Каждой нити приписан процесс, называемый монитором нити. Монитор нити в каждый момент рассматривает набор реальных процессов и совокупность потенциальных процессов. *Потенциальный процесс* определяется рекурсивно.

1. Пустой набор с приоритетом — потенциальный процесс.

2. Если к набору добавить элементарное действие с приоритетом и спецификациями, то он останется потенциальным процессом.

3. Если к набору добавить специфицированный потенциальный процесс, то он останется потенциальным процессом.

Каждому потенциальному процессу сопоставляется его *охрана* — некоторое логическое условие. Монитор в порядке убывания приоритетов рассматривает потенциальные процессы. Если охрана истинна, то он проверяет наличие разрешений на все элементарные действия и их спецификации, непосредственно подчиненные потенциальному процессу. При этом он может обращаться за информацией к другим мониторам. Если все разрешения получены, система потенциальных процессов может быть перестроена, а все активизированные действия становятся при этом элементами набора реальных процессов. Реальные процессы, принадлежащие другим мониторам, могут выполняться лишь как распределенные и передаются соответствующему монитору. Завершенные и убитые реальные процессы удаляются из списка.

Перестройка системы потенциальных процессов производится путем замены активизированного потенциального процесса некоторой совокупностью потенциальных процессов. Если вновь созданные потенциальные процессы принадлежат другим мониторам, они отправляются соответствующим мониторам.

Потенциальные процессы, видимо, лучше всего создавать при помощи аппарата нейронных сетей, по самой своей природе также хаотического. А это заодно является первым шагом к важнейшей перспективной задаче информатики: нейронные сети высших порядков, что позволит реализовать принцип Гильберта во множестве ныне недоступных для него ситуаций.

Приведём фрагмент динамического описания того же процесса, который описан исчислением Милнера.

```
Initiate \{
    \guard (jurist.respond_full) \Everything =>
        priority high:<message (Необходима полная переделка)
            <Stop>>;
  \guard (\not jurist.respond_full) \Everything =>
            \Everything priority normal
< Assign (jurist.Getparagraphs) () kadrs.getchiefs>
            priority low:
              <wait(7:day>);}
Assign {
( Paragraph.this. Expr.4) (Expr.5) Expr.1 Person.resp Expr.2
    \guard(Includes(this,resp))
            => <Send_message
          this "Вы ответственны за переработку параграфа"
```

```
                    resp make_responsible(this,resp)>
priority high: <Assign ( Paragraph.this. Expr.4)
        (Expr.5 resp) Expr.1 Expr.2>;
        () (Expr.Responsibilities) => ;
}
```

Список литературы

1. Арнольд В. И. Жесткие и мягкие математические модели. Лекция для аппарата Президента России.
2. Михеев А.Г., Орлов М.В. Перспективы Workflow-систем. // PC Week/RE №23/2004 от 29.06.2004, с. 26-30; №28/2004 от 10.8.2004, с. 21-25; №43/2004 от 23.11.2004, с. 36-40; №36/2005 от 4.10.2005, с. 46-50.
3. Непейвода Н. Н. Прикладная логика. 3-е изд. переработанное и дополненное. Москва, Direct Media, 2019.
4. Непейвода Н. Н. Об уровнях знаний и умений. Логические исследования, вып. 8, 1999.
5. Непейвода Н. Н., Скопин И. Н. Основания программирования. РХД, 2004.
6. Непейвода Н. Н. Стили и методы программирования. Интуит, 2005.
7. Непейвода Н. Н. Математик и прикладник: о взаимо(не)понимании. Вестник Удмуртского университета, математика. №1, 2007, стр. 251-268.
8. N. N. Nepejvoda Abstract Incompleteness Theorems and Their Influence in Methodology Studia Humana Volume 1:3/4 (2012), pp.43—58
9. Альтшуллер Г. С. Теория решения изобретательских задач. М.: Советское Радио, 1983.
10. Бек К. Экстремальное программирование. Питер, 2002.
11. Разработка методических рекомендаций по описанию и оптимизации процессов в органах исполнительной власти в рамках подготовки внедрения ЭАР. Ч. 1, 2. Отчет Высшей школы экономики, 2004 г.
12. Robin Milner. Communicating and Mobile Systems: the Pi-Calculus. — Cambridge University Press, 1999. — 174p. — ISBN 0521658691.
13. Robin Milner. The Polyadic n-Calculus: A Tutorial // Logic and Algebra of Specification, Springer-Verlag, 1993.
14. Davide Sangiorgi and David Walker. The pi-calculus: a Theory of Mobile Processes. — Cambridge University Press, 2001. — ISBN 0521781779.
15. e-Services development framework primer. UK online, 2002
16. Stefen A. White. Process modeling notation and workflow patterns. BP Trends, March 2004.
17. http://www.bpmi.org/bpml-spec.htm

References in Cyrillics

1. Arnol`d V. I. Zhestkie i myagkie matematicheskie modeli. Lekciya dlya apparata Prezidenta Rossii.
2. Mixeev A.G., Orlov M.V. Perspektivy` Workflow-sistem. // PC Week/RE №23/2004 ot 29.06.2004, c. 26-30; №28/2004 ot 10.8.2004, c. 21-25; №43/2004 ot 23.11.2004, c. 36-40; №36/2005 ot 4.10.2005, c. 46-50.
3. Nepejvoda N. N. Prikladnaya logika. 3-e izd. pererabotannoe i dopolnennoe. Moskva, Direct Media, 2019.
4. Nepejvoda N. N. Ob urovnyax znanij i umenij. Logicheskie issledovaniya, vy`p. 8, 1999.
5. Nepejvoda N. N., Skopin I. N. Osnovaniya programmirovaniya. RXD, 2004.
6. Nepejvoda N. N. Stili i metody` programmirovaniya. Intuit, 2005.
7. Nepejvoda N. N. Matematik i prikladnik: o vzaimo(ne)ponimanii. Vestnik Udmurtskogo uni-versiteta, matematika. №1, 2007, str. 251-268.
8. N. N. Nepejvoda Abstract Incompleteness Theorems and Their Influence in Methodology Studia Humana Volume 1:3/4 (2012), pp.43—58
9. Al`tshuller G. S. Teoriya resheniya izobretatel`skix zadach. M.: Sovetskoe Radio, 1983.
10. Bek K. E`kstremal`noe programmirovanie. Piter, 2002.
11. Razrabotka metodicheskix rekomendacij po opisaniyu i optimizacii processov v organax ispolnitel`noj vlasti v ramkax podgotovki vnedreniya E`AR. Ch. 1, 2. Otchet Vy`sshej shkoly` e`konomiki, 2004 g.
12. Robin Milner. Communicating and Mobile Systems: the Pi-Calculus. — Cambridge University Press, 1999. — 174p. — ISBN 0521658691.
13. Robin Milner. The Polyadic n-Calculus: A Tutorial // Logic and Algebra of Specification, Springer-Verlag, 1993.
14. Davide Sangiorgi and David Walker. The pi-calculus: a Theory of Mobile Processes. — Cambridge University Press, 2001. — ISBN 0521781779.
15. e-Services development framework primer. UK online, 2002
16. Stefen A. White. Process modeling notation and workflow patterns. BP Trends, March 2004.
17. http://www.bpmi.org/bpml-spec.htmСведения об авторах

Непейвода Николай Николаевич, ORCID 0000-0002-7869-8053, д.ф.-м.н., профессор,
Институт программных систем им. А. К. Айламазяна РАН

Ключевые слова
Ключевые слова: деятельность, инициатива, неформализуемость, надёжность.

Nikolai Nepeyvoda. Introduction to chaotic control (network version)

Keywords
Keywords: activity, initiative, informalizability, robustness.

DOI: 10.34706/DE-2020-02-03

JEL Classification: C63 – Computational Techniques, Simulation Modeling, C88 – Other Computer Software.

Abstract
This work presents an approach to planning and moderating activity when our agents are untrustworthy and creative in the same time. In this case we try to organize a system of partially ordered priorities and not make plans deeper than three steps forward. Thus, we can use this model for agents that could accidentally be initiative.