УДК 519.07

П.С. САПАТЫЙ, А.А. МОРОЗОВ, В.П. КЛИМЕНКО

РАСПРЕДЕЛЕННАЯ ТЕХНОЛОГИЯ ГЛОБАЛЬНОГО УПРАВЛЕНИЯ

Анотація. *Успадковуючи холістську ідеологію, де ціле є первинне і більше суми часток, пропонується новий підхід, який охоплює семантику та цілісність великих розподілених систем безпосередньо на верхньому рівні. Це дозволяє переводити більшість традиційних рутин системної організації і управління на автоматизоване та навіть цілком автоматичне виконання, суттєво підвищуючи системну гнучкість, надійність та можливість самовідновлення і самоперебудови в асиметричных ситуаціях. Викладаються основи відповідної мережевої технології, та подаються приклади її численних застосувань.*
Ключові слова: *атомізм, холізм, гештальт, цілісність, динамічні системи, керування високого рівня, мова розподілених сценаріїв, паралельна інтерпретація, гнучкість, самовідновлення, асиметричні ситуації, реальний час.*

Аннотация. *Наследуя холистскую идеологию, где целое первично и больше суммы частей, предлагается новый подход, охватывающий семантику и целостность больших распределенных систем непосредственно на верхнем уровне. Это позволяет эффективно переводить большинство традиционных рутин системной организации и управления на автоматизированное или даже полностью автоматическое исполнение, существенно повышая системную гибкость, надежность и возможность самовосстановления и самоперестройки в асимметричных ситуациях. Излагаются основы соответствующей сетевой технологии, и предоставлены примеры ее многочисленных применений.*
Ключевые слова: *атомизм, холизм, гештальт, целостность, динамические системы, высокоуровневое управление, язык распределенных сценариев, параллельная интерпретация, гибкость, самовосстановление, асимметричные ситуации, реальное время.*

Abstract. *Inheriting holistic ideology, where the whole is primary and greater than the sum of parts, a new approach is offered which is grasping semantics and integrity of large distributed systems directly on top level. This enables most traditional routines on system organization and management to be shifted toward automated up to fully automatic implementation, radically enhancing system flexibility, safety, and capability of self-recovery and self-restructuring in asymmetric situations. Basics of the related networking technology and examples of its numerous applications are revealed.*
Key words: *atomism, holism, gestalt, integrity, dynamic systems, high level management, distributed scenario language, parallel interpretation, flexibility, self-recovery, asymmetric situations, real time.*

1. Введение

Получено гибкое универсальное решение для менеджмента распределенных динамических систем, охватывающее и обеспечивающее напрямую, на семантическом уровне, их целостность. Оно абстрагируется от возможных системных структур и внутренней организации, которые могут быть эффективно переведены на автоматизированный или даже полностью автоматический уровень. Это в корне отличается от других подходов, которые рассматриавают большие системы как заранее состоящие из отдельных, наперед заданных компонент (часто называемых «агентами» [1]), взаимодействие их должно привести к желаемым глобальным результатам, чего, однако, часто не происходит.

Разработанные идеология и технология [2–4] основаны на параллельной интерпретации компактных системных сценариев, представленных на специальном Языке Распределенных Сценариев (ЯРС) и покрывающих виртуальное, физическое или же комбиниро-

ISSN 1028-9763. Математичні машини і системи, 2010, № 4

ванное пространство в реальном времени, задавая требуемое функционирование распределенных систем, автоматически восстанавливая работоспособность после произвольных сбоев, а также всегда поддерживая локальную и глобальную целевую ориентацию. Эти сценарии могут создавать распределенные инфраструктуры различного назначения, а также раширять, модифицировать и даже проникать в другие имеющиеся инфраструктуры, обеспечивая широкий спектр возможных воздействий на распределенные системы (как дружественные, так и враждебные) – от их создания, оптимизации и модификации до подавления и, если нужно, ликвидации.

Данная технология опробована в разных странах, о ней докладывалось на многочисленных международных симпозиумах и конференциях, с исследованными применениями покрывающими такие тематики, как распределенные базы знаний, интеллектуальный менеджмент компьютерных сетей, распределенное моделирование динамических систем типа поля боя, управление транспортными системами, управление системами энергии направленного действия (лазеры), военная авионика, тактические коммуникации, электронные и кибернетические войны, распределенные сенсорные системы, массовая кооперативная роботика, а также решение сложных демографических проблем в развитых странах, связанных, прежде всего, с быстрым относительным ростом стареющего населения. Целый ряд исследованных применений уже опубликован (например, [2–4, 10–13]), большинство, однако, находится в активной фазе и печати, а часть обсуждается в настоящей работе.

2. Холизм и целостность против традиционного атомизма: новая системная организационная модель

2.1. Системный атомизм и его недостатки

Обычно мы преследуем атомистическую, «часть-целое», философию в формализации, дизайне и имплементации больших систем, особенно распределенных. Исходно, идея некоторой системы или же системной кампании появляется в очень общей, целостной форме в уме некоторого индивидуума или же в очень тесном коллективе таких умов (рис. 1а). Затем эта идея детализируется путем разбиения ее на части, где каждая часть прибретает индивидуальность, при этом неизбежно усложняясь и вырастая в объеме, как на рис. 1б.

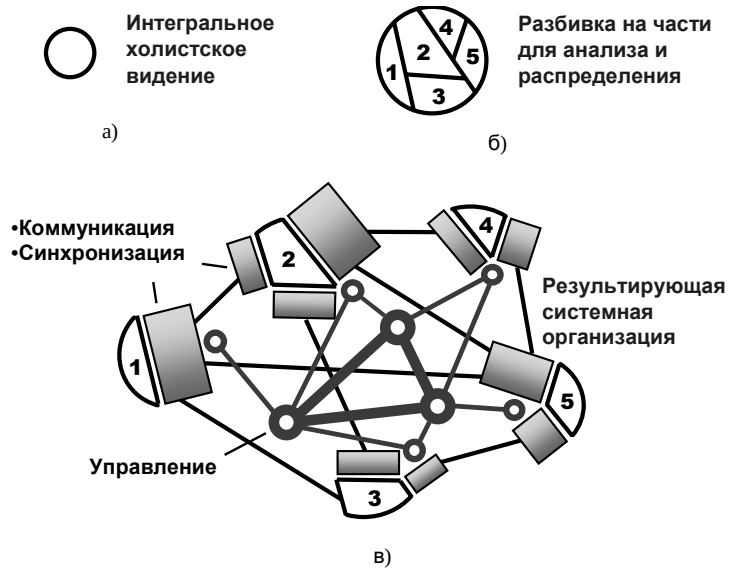

Рис. 1. Традиционный атомистический подход к системному дизайну и менеджменту

Следующим шагом является материализация уже определенных частей и их распределение в физическом или виртуальном пространстве. Чтобы обеспечить совместную работу этих частей в рамках исходной идеи (рис. 1а), может потребоваться развитая система их взаимной коммуникации и синхронизации, а также сложная управляющая инфраструктура, что символично показано на рис. 1в. Все это может привести (что и часто бывает) к очень большим системным издержкам, сущест-венно превышающим исход-ный вес и сложность самой идеи.

Обычно исходная идея и даже ее логически расщепленная версия (рис. 1 а, б) остаются только в умах создателей, в то время как реальная системная формализация и имплементация начинаются только с ее полностью распределенного этапа (рис. 1в). Это часто порождает следующие проблемы:

• Могут возникать существенные трудности, связанные с попыткой поставить поведение результирующей распределенной системы в соответствие с исходной идеей.

• Скомпонованное (и в надежде, восстановленное в исходном виде) целое может обладать совершенно непредвиденными свойствами, включая явно нежелательные.

• Результирующее решение является преимущественно статическим. Даже при незначительном изменении исходной идеи может возникнуть необходимость абсолютно нового системного дизайна или же полной перекомпоновки системы.

• Просто попытка подогнать существующую многокомпонентную систему под новую идею может вылиться в недопустимую потерю системной целостности и производительности.

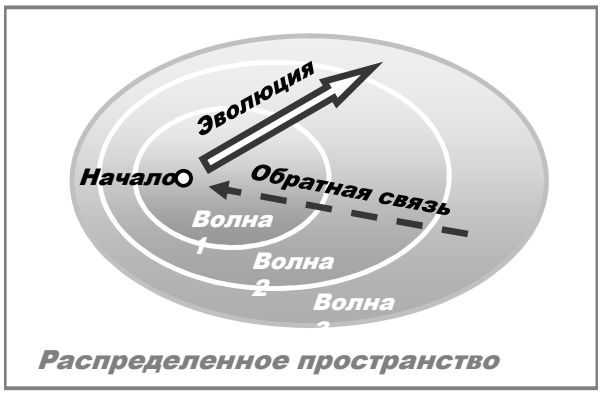

Рис. 2. Покрытие распределенного пространства с помощью параллельных волн

2.2. Волновая модель системного видения и управления

Системная модель, описанная здесь и наследующая холистскую и гештальтистскую философии [5, 6], базируется на распространении в распределенной среде параллельных взаимодействующих волн (как прямых, так и обратных), которые могут в реальном времени покрывать пространство требуемыми решениями, перенося на своих фронтах цели, знания, данные, операции и управление (рис. 2).

Эта модель также созвучна с результатами психологических и биологических исследований о решении человеческим мозгом сложных пространственных задач [7, 8], будучи переведенным в нашем случае на полностью распределенную, высокопараллельную, как концептуальную, так и технологическую основу. Этот подход позволяет обеспечивать целостную, бесшовную навигацию и покрытие распределенных систем в прямом противоречии с традициоными атомистическими, часто называемыми многоагентными, моделями [1]. Наша модель, в значительной мере, также ориентирована на преобразование и воздействие на окружающий мир в виде «активного направленного порыва» (active forward thrust, [8]), что в корне отличается от фрейдовской эквилибристской, сбалансированной системной идеологии [9].

В волновом формализме разработанной системной модели можно представить любой системный сценарий, который, будучи написанным на высокоуровневом Языке Распределеных Сценариев (ЯРС) (очень компактном, где программы часто в сто раз короче, чем на языке Джава), может легко быть модифицирован (и даже порожден с начала) в реальном времени.

Поскольку ЯРС эффективно и напрямую интерпретируется в любых сетевых структурах, большая часть системных организационных рутин (если не все), изображенных на рис. 1в, может исполняться и поддерживаться автоматически. В связи с этим мы получаем возможность радикального повышения целостности и гибкости любых распределенных систем (от гражданских до военных), цели и миссии которых могут определяться и изменяться «на лету», вовремя реагируя на быстроменяющиеся, «асимметричные» ситуации.

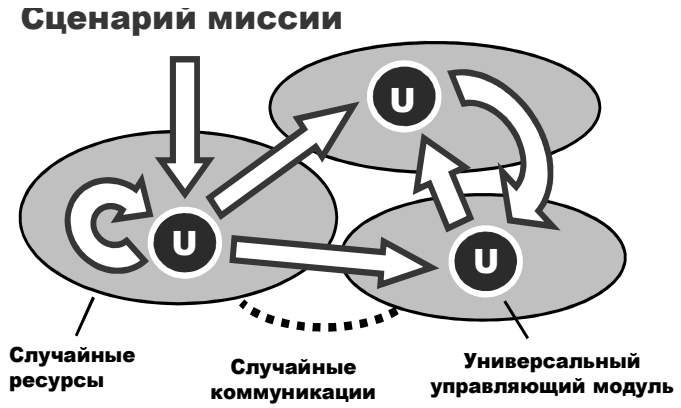

Рис. 3. Саморазвивающиеся сценарии системных миссий

2.3. Распределенная интерпретация волновых сценариев

Атомизм в предложенной модели, описывающей назначение системы и ее необходимое глобальное поведение на высшем, семантическом уровне, появляется только во время исполнения. Здесь система, как обычно, представляется в виде взаимодействующих компонент (все состоит в конечном счете из взаимодействующих атомов, мозг тоже), каждая из которых содержит универсальный управляющий модуль *U* (рис. 3), представляющий одну и ту же копию интерпретатора ЯРС, которые могут обмениваться друг с другом.

Динамическая сеть интерпретаторов ЯРС, вмонтированных в наиболее важные точки системных ресурсов (которые сами могут быть случайными, с непредвиденными наперед взаимодействиями и связями), коллективно, в параллельном и распределенном режимах исполняет интегральные сценарии, которые могут начинаться с любой компоненты. Волновые сценарии на ЯРС в реальном времени могут покрывать всю систему, перемещаясь, модифицируясь и размножаясь, формируя при этом бесшовные распределенные инфраструктуры самого разного характера (от сетей знаний до командно-управляющих структур, дорожных карт, нейронных сетей), обеспечивающие требуемое глобальное решение или поведение, а также всеобщую осведомленность и автоматическое принятие автономных решений.

3. Язык Распределенных Сценариев (ЯРС)

ЯРС кардинально отличается от традиционных языков программирования тем, что вместо обычной обработки данных в компьютерной памяти (но, если нужно, также наряду с ней) позволяет непосредственно (как последовательно, так и параллельно) перемещаться, обозревать и выполнять любые действия в распределенных средах самого разного характера (от чисто физических до чисто виртуальных).

3.1. Миры, с которыми оперируют ЯРС

ЯРС напрямую оперирует со следующими мирами:

а) виртуальным миром (ВМ), конечным и дискретным, представленным распределенной семантической сетью, состоящей из вершин, выражающих имена, концепции или факты, и семантических связей между ними;

б) физическим миром (ФМ), бесконечным и непрерывным, где каждая точка может идентифицироваться, а также быть доступна, с помощью физических координат (с определенной степенью точности);

в) виртуально-физическим миром (ВФМ), конечным, дискретным и сетевым, как и ВМ, но в котором все или некоторые вершины имеют дополнительное представление в ФМ, наделяясь также определенными физическими координатами.

3.2. Основные свойства ЯРС

ЯРС обладает следующими основными свойствами:

1. Выраженный с помощью ЯРС высокоуровневый системный сценарий развивается в распределенной среде как переход между контрольными опорными пунктами (ОП) в виде параллельных волн (рис. 2).

2. Начиная с некоторого ОП, заданное действие может породить один или более новых ОП.

3. Каждый ОП имеет ассоциированное с ним результирующее значение (которое может быть множественным), а также результирующее состояние с перечнем из следующих четырех возможностей:

• Успех – полностью успешное выполнение с определенным результатом, позволяющее также последующим процессам развиваться далее в среде с этого ОП.

• Завершение – успешное выполнение, с оределенным результатом, но с плановым завершением процессов только в этом ОП, без влияния на другие процессы.

• Неудача – провал активности, всегда с пустым результатом, и с завершением процессов только в этом ОП, без влияния на другие процессы.

• Отмена – эквивалент чрезвычайного состояния, также с пустым результатом, когда процесс в данном ОП, а также процессы, связанные с другими ОП, немедленно прекращаются (насколько быстро это возможно организовать в распределенной среде).

4. Различные действия могут развиваться независимо друг от друга (и параллельно) или же, наоборот, взаимозависимо, с одного и того же ОП, формируя совместно новое, результирующее, множество ОП.

5. Различные действия могут также следовать друг за другом в пространстве ОП, где последующие действия выполняются (параллельно) от всех или же части ОП, достигнутых предыдущими действиями.

6. Элементарные операции могут напрямую использовать как локальные, так и удаленные значения ОП, полученные в результате других действий (включающих сценарии любой сложности), результируют в виде значения ОП от этих элементарных операций.

7. Эти результирующие значения могут использоваться как операнды другими операциями в аналитических выражениях (произвольной вложенности) или же непосредственно следующими операциями (которых может быть множество, если процессы расщепляются в пространстве) в их последовательности.

8. Полученные значения могут напрямую быть присвоены местным или же удаленным переменным (для второго случая доступ к ним может вызвать необходимость исполнения пространственных сценариев любой сложности).

9. Любой ОП может ассоциироваться с некоторой вершиной в ВМ или же позицией в ФМ, а также с обоими (как в случае работы с ВФМ); он также может ссылаться на разные места в ВМ и ФМ отдельно, независимо и одновременно.

10. Любое количество ОП может одновременно ассоциироваться с одними и теми же точками в мирах (физических, виртуальных, комбинированных), с которыми работают сценарии.

11. Ассоциируя с конкретными позициями в каком-либо мире, сценарии получают прямой доступ к локальным параметрам мира в этих точках, включая возможность их изменения и, следовательно, преобразования самого мира.

12. Передвижение в физическом, виртуальном или комбинированном мирах, с их возможной модификацией или даже первоначальным созданием, является такой же рутинной операцией, как арифметическая, логическая или потоковая в традиционных языках программирования. При этом ЯРС может также эффективно использоваться как обычный язык программирования (типа С, Джавы или Фортрана), претендуя быть единственным базовым языком системы как для (последовательных или параллельных) вычислений, интеллектуальных протоколов обмена, так и глобального (централизованного или распределенного) управления.

3.3. Синтаксис и основные конструкции ЯРС

ЯРС имеет рекурсивный синтаксис, представленный на верхнем уровне, где сценарий (программа) носит название *захват* (что отражает ее основную семантику как задающую глобальную целевую ориентацию с последующим вовлечением и интеграцией распределенных системных ресурсов (среди которых могут быть как человеческие, так и технические) для выполнения поставленной системной задачи. (Фигурные скобки используются для выражения повторов, а вертикальная черта разделяет альтернативы).

захват → *феномен* | *правило* ({ *захват* , })
феномен → *константа* | *переменная* | *спецслово*
константа → *информация* | *вещество* | *комбинированная*
переменная → *наследуемая* | *фронтальная* | *пространственная* | *вершинная*
правило → *движение* | *создание* | *ликвидация* | *эхо* | *слияние* | *верификация* | *присвоение* | *продвижение* | *ветвление* | *передача* | *временное* | *наделение*

Базовая конструкция, *правило*, может представлять любое определение или действие, например:

• элементарную арифметическую, строковую или логическую операцию;
• прыжок в физическом, виртуальном или комбинированном пространстве;
• иерархическое слияние и возврат (удаленных) данных;
• распределенное управление как последовательное, так и параллельное;
• разнообразие специальных контекстов для навигации пространства, детализирующих последующие операции и решения;
• тип или смысл значения, или же выбранное его использование, направляющее процесс интерпретации.

3.4. Типы переменных в ЯРС

ЯРС использует различные типы переменных:

• Наследуемые переменные, порожденные от некоторого ОП и обслуживающие все последующие ОП, в которых они могут использоваться как общий ресурс для чтения и записи.

• Фронтальные переменные, являющиеся индивидуальной и эксклюзивной собственностью ОП, недоступной для других ОП, и передающиеся только между смежными ОП, реплицируясь, если после некоторого ОП порождается множество других ОП.

• Пространственные переменные, которые обеспечивают доступ к различным элементам физического или виртуального пространства во время его навигации, а также к разнообразию параметров внутреннего мира распределенного интерпретатора ЯРС.

• Вершинные переменные, позволяющие наделять индивидуальным временным свойством отдельные вершины ВМ и ВФМ, которое может выступать как общий ресурс для всех ОП, ассоциирующихся с этими вершинами.

Различные типы переменных, особенно, когда используются совместно, позволяют создавать эффективные пространственные алгоритмы, которые не ассоциируются наперед с какими-либо компьютерными ресурсами, работая при этом скорее между ними, чем в них. Эти алгоритмы могут также свободно перемещаться в активных распределенных средах (частично или полностью), всегда сохраняя целостность, управляемость извне и заданную целеустремленность.

Несмотря на (непосредственно интерпретируемый) функциональный тип представления синтаксиса и семантики, ЯРС позволяет также использовать другие, традиционные, формы записи программ с общепринятыми символами операций и разделителей, если это упрощает и сокращает тексты сценариев.

3.2. Полный перечень конструкций ЯРС

Полный набор конструкций ЯРС (в латинском шрифте) выглядит следующим образом (где курсивом представлены синтаксические категории):

grasp → *constant* | *variable* | *rule* | [*grasp*] [({ *grasp*, })]
constant → *information* | *matter* | *code* | *service*
information → *number* | {*alphadigit*} | '*string*'
matter → "*string*"
code → **{** *string* **}**
service → abort | thru | done | fail | any | first | last | random | all | in | out | infinite | nil | direct | no back | passed | covered | existing | neighbors | virtual | physical | executive | local | global | synchronous | peers | value | destinations | sequential | parallel
variable → *navigational* | *environmental* | *combined*
navigational → *heritable* | *frontal* | *reserved*
heritable → H {*alphadigit*}
frontal → F {*alphadigit*}
reserved → VALUE | STATE
environmental → *nodal* | *dedicated*
nodal → N {*alphadigit*}
dedicated → ADDRESS | CONTENT | DOER | NAME | QUALITIES | SPEED | TIME | TYPE | WHEN | WHERE
combined → IDENTITY | BACK | PREVIOUS | LINK | DIRECTION
rule → *movement* | *creation* | *elimination* | *echoing* | *fusion* | *verification* | *assignment* | *modification* | *advancement* | *branching* | *transference* | *exchange* | *waiting* | *mode* | *type* | *usage*
movement → hop | move
creation → create | linkup
elimination → remove | unlink
echoing → value | state | level | order | rake | count | min | max | sort | sum | product | average
fusion → add | subtract | multiply | divide | degree | separate | unite | attach | append | common | element | position
verification → equal | not equal | less | less equal | more | more equal | empty | not empty | belongs | not belongs | intersects | not intersects
assignment → assign
modification → insert | replicate | substitute
advancement → advance | repeat
branching → choose | independent | sequence | while | if | or | and | cycle | loop | sling | split
transference → run | call
exchange → input | output
waiting → wait | remain | sleep
mode → free | release | quit | none | lift | stay | seize
type → heritable | frontal | nodal | number | string | info | matter | code
usage → name | address | place | center | range | time | speed | doer | node | link | unit | content | index | parameters

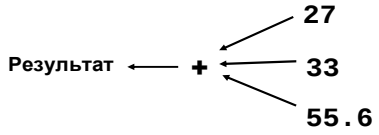

Рис. 4. Присваивание результата операции

4. Элементарные примеры на ЯРС

1. Присваивание переменной результата арифметической операции (рис. 4):

```
присвоить (Результат, сложить (27, 33, 55,6))
```

или

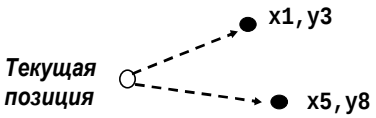

Рис. 5. Одновременное движение в физическом пространстве

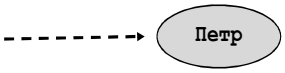

Рис. 6. Создание вершины

Рис. 7. Расширение семантической сети

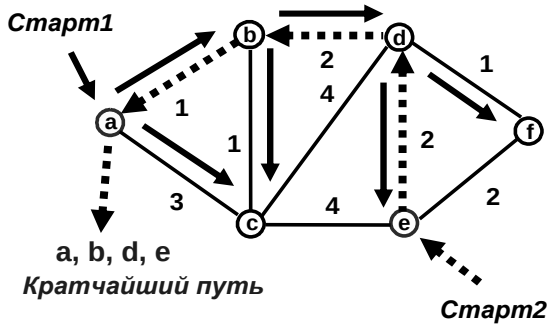

Рис. 8. Нахождение кратчайшего пути

```
Результат = 27 + 33 + 55,6.
```

2. Параллельное движение в физическом мире, рис. 5:

```
переместиться (точка (x1, y3),
    точка (x5, y8).
```

3. Создание новой вершины в виртуальном мире (рис. 6).

```
создать(вершина (Петр)).
```

4. Расширение виртуального мира новой парой «связь-вершина» (рис. 7):

```
войти (Петр); создать (связь
(+отец), вершина (Алексей)).
```

5. Нахождение кратчайшего (между вершинами a и e) пути в распределенной среде – полностью распределенное высокопараллельное решение, опробованное через Интернет, с использованием взаимодействующих компьютеров в разных странах (рис. 8).

```
Фронтальная (Удаленность, Путь);
вершинная (Расстояние,
      Предшественник);
 последовательность (
  (прыжок (прямой, a);
   Расстояние = 0;
   повторять (
     прыжок (все связи);
     Удаленность += СВЯЗЬ;
     или (Дистанция == пусто,
          Дистанция > Удаленность);
     Дистанция = Удаленность;
     Предшественник = НАЗАД),
 (прыжок (прямой, e);
  повторять (
    Путь = ИМЯ & Путь;
    прыжок (Предшественник));
  выход (Путь))))
```

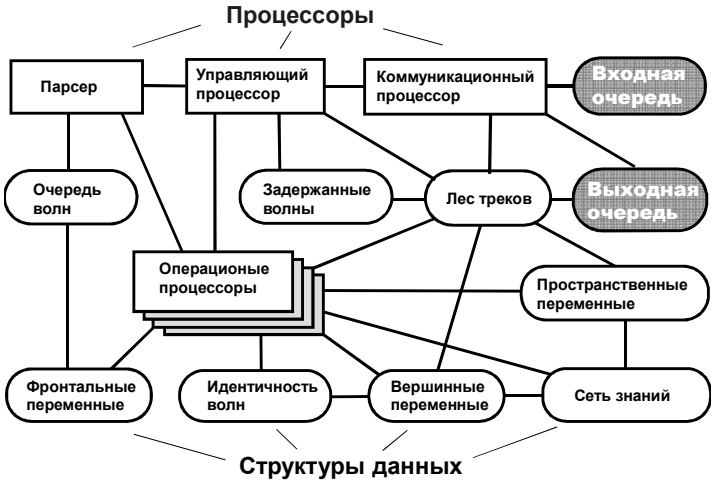

Рис. 9. Организация интерпретатора с ЯРС

5. Распределенный интерпретатор ЯРС

5.1. Структура и основные свойства

Интерпретатор ЯРС (рис. 9) имеет следующую организацию [14]:

1. Он состоит из ряда специализированных модулей, работающих параллельно друг с другом и имеющих доступ к специфическим структурам данных, которые поддерживают часть ассоциированного с данным интерпретатором виртуального мира, а также временные иерархические управляющие механизмы и обмен с другими такими же интерпретаторами.

2. Вся сеть интерпретаторов может быть мобильной и открытой, меняя при этом в реальном времени число вершин и коммуникационную структуру между ними.

3. «Нервной системой» распределенного интерпретатора является его пространственная трековая структура, позволяющая осуществлять иерархическое управление и удаленный доступ к данным и программному коду, обеспечивающая также целостность саморазвивающихся параллельных и распределенных решений.

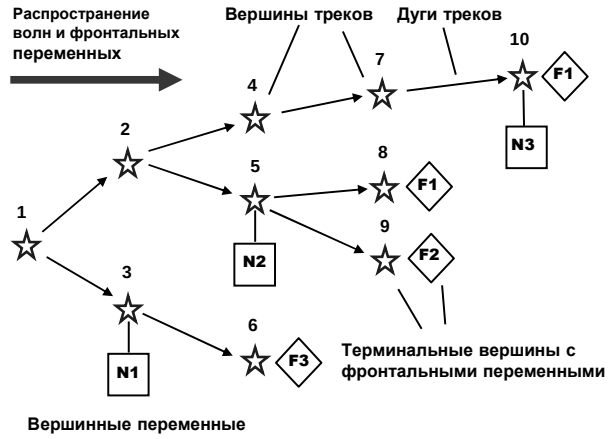

Рис. 10. Поддержка распространения сценариев с помощью системы треков

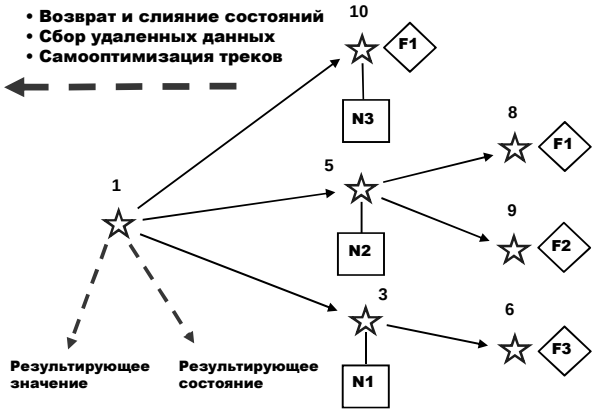

Рис. 11. Эхо-процессы и автоматическая оптимизация треков

5.2. Пространственная трековая инфраструктура

Динамически формируемые трековые деревья (рис. 10, 11), покрывающие пространство (виртуальное, физическое, комбинированное), в котором развиваются сценарии, выраженные в ЯРС, поддерживают существование пространственных переменных и отрабатывают логику смещения, усреднения и возврата терминальных состояний, а также возврат полученных удаленных значений ОП (если для последних был запрос от вышестоящих организационных уровней).

Динамические трековые деревья самооптимизируются в эхо-процессах, удаляя свои, ставшими излишними части (рис. 11 по сравнению с рис. 10). Затем оптимизированные трековые деревья обеспечивают маршрутизацию и доставку последующих волн сценариев в навигируемом пространстве к ОП, достигнутым предыдущими волнами, объединяя их в этих терминальных пунктах с доставленными туда ранее фронтальными переменными (для того, чтобы начать новые пространственные процессы).

6. Решение топологических задач

ЯРС является языком, в котором очень удобно создавать любые распределенные структуры, а также анализировать и обрабатывать их в распределенном и параллельном режимах, без каких-либо центральных вычислений и устройств. Эти структуры могут отражать, например, топологию транспортных систем (наземных, подземных, морских, воздушных), организацию войск на поле боя или же рыночные связи в экономике. Первое, что может быть интересным, это нахождение в них слабых и сильных звеньев для последующего оптимального воздействия на них (или же наоборот, дистанцирования как от угрозы).

6.1. Точки сочленения

Точки сочленения (как, например, вершина d на рис. 12) являются самыми слабыми в графе, удаление которых расчленяет его на части. В ЯРС решение этой задачи в высокопараллельной и полностью распределенной форме имеет вид, показанный ниже:

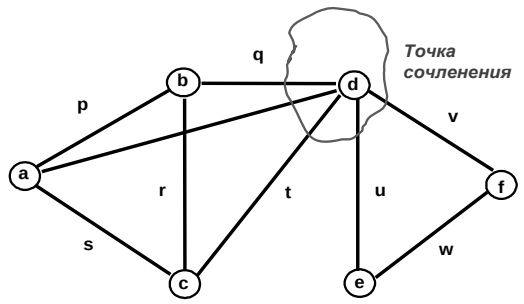

Рис. 12. Нахождение точек сочленения

```
прыжок (все вершины);
вершинная (Метка);
ИДЕНТИЧНОСТЬ = ИМЯ; Метка = 1;
И (
 (случайный выбор (
    прыжок (все дуги));
  повторять (
    захват (
      Метка == пусто; Метка = 1);
    прыжок (все дуги))),
 (прыжок (все дуги);
  Метка == пусто),
 выход (ИМЯ))
```

6.2. Максимальные сильносвязанные компоненты: клики

Максимальные сильносвязанные (то есть полные) подграфы, или *клики*, наоборот, являются самыми сильными компонентами структур, как $(a,\ b,\ c,\ d)$, $(c,\ d,\ e)$ и $(d,\ e,\ f)$ на рис. 13. Их параллельное и полностью распределенное нахождение в ЯРС приведено ниже:

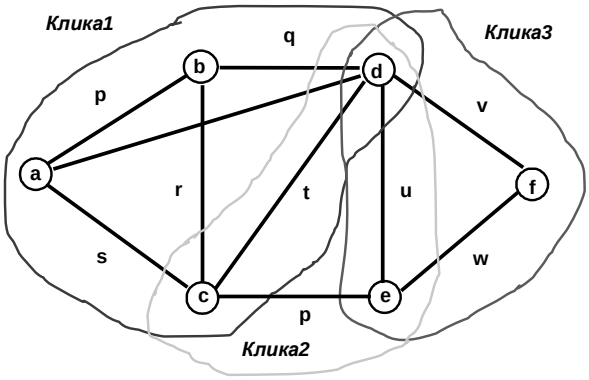

Рис. 13. Нахождение всех максимальных сильносвязанных компонент (клик)

```
прыжок (все вершины);
фронтальная (Клика = ИМЯ);
повторять (
 прыжок (все дуги);
 не принадлежит (ИМЯ, Клика);
 и (
  параллельное и (
   прыжок (все дуги, вершины
    (Клика)); стоп),
  или (
   (НАЗАД > ИМЯ; стоп),
   присоединить (Клика, ИМЯ))));
выход (Клика)
```

7. Коллективная роботика

7.1. Интеграция мобильных роботов на базе параллельной интерпретации ЯРС

Установка интерпретатора в мобильных роботах (наземных, воздушных, водных, подводных) позволяет организовать эффективные групповые роботические решения сложных задач в опасных распределенных средах, полностью переводя локальное и глобальное управление на автоматический уровень. Пример интеграции европейских наземных роботов (обсуждавшийся нами в Германии на полевых испытаниях мобильных роботов и в Англии на международных конференциях) показан на рис. 14.

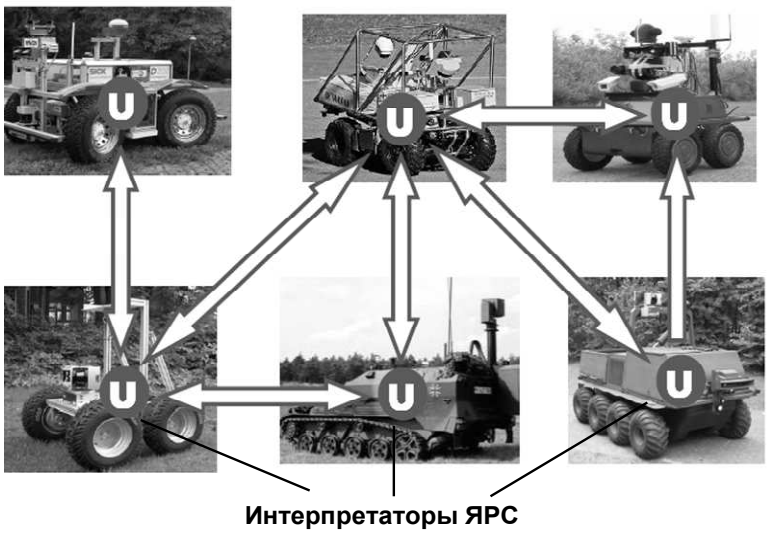

Рис. 14. Интеграция европейских наземных мобильных роботов

Пример интеграции воздушных военных роботов в коллективные беспилотные формирования, обсуждавшийся на других международных встречах, показан на рис. 15.

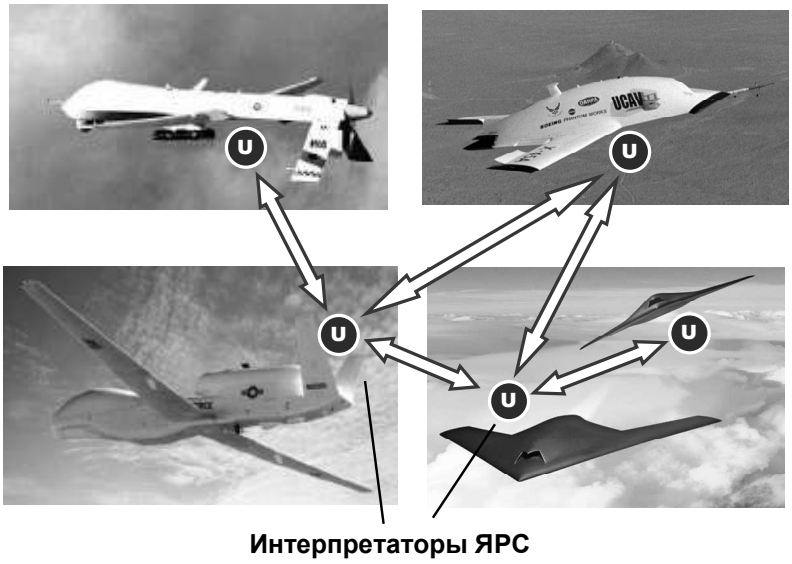

Рис. 15. Интеграция летающих беспилотных средств

Предложенная технология позволяет конвертировать любую группу (в пределе: армию) самых разнообразных мобильных роботов в динамическую параллельную пространственную машину, которая сможет выполнять любую миссию полностью автоматически.

Задание и организация совместной работы роботических единиц на ЯРС может быть на разных уровнях: от чисто семантического, задачного, абстрагирующегося от многокомпонентной структуры системы (где последняя выступает как производная от формулировки задачи на высшем уровне) до более детального, многокомпонентного уровня (где производной, наоборот, являются задача и ее конечный результат).

7.2. Семантический, задачный уровень

Пример семантического, задачного уровня может выглядеть таким образом, когда программа на ЯРС напрямую соответствует формулировке задания на естественном языке.

Задача на естественном языке:

Осуществить доступ к трем пунктам в аварийной зоне с координатами (50,433; 30,633), (50,417; 30,490), и (50,467; 30,517). Оценить ущерб в каждом пункте, определить пункт с максимальным ущербом и передать величину ущерба в этом пункте вместе с его уточненными физическими координатами в центральный штаб.

Соответствующая программа на ЯРС:

```
передать ( максимум (
  переместиться ((50,433, 30,633),
                 (50,417, 30,490),
                 (50,467, 30,517));
  присоединить (оценить (разрушения), ГДЕ)))
```

В ряде публикаций, включая [2, 3], описаны детали полностью автоматического исполнения данного и других подобных сценариев группами роботов с меняющимся (что может происходить в реальном времени) числом мобильных единиц.

7.3. Групповой поведенческий уровень

Установив интепретаторы ЯРС в мобильные роботы и интегрировав их с традиционной функциональностью роботов, мы можем эффективно организовывать их любое совместное поведение – от слабосвязанных «свормов» до интегрированной, слаженной команды с сильным иерархическим подчинением. В ЯРС также возможна и эффективна любая комбинация различных поведений. Рассмотрим соответствующие примеры.

Пусть исходное распределение роботов (предположительно, беспилотников) имеет вид, изображенный на рис. 16.

Вживив в эту группу следующий коллективный сценарий (назовем его ***движение_группы***), начав с любого робота, мы легко задаем индивидуально случайное, но глобально ориентированное групповое движение, покрывающее нужное пространство, где роботы постоянно держат определенную пороговую дистанцию друг от друга.

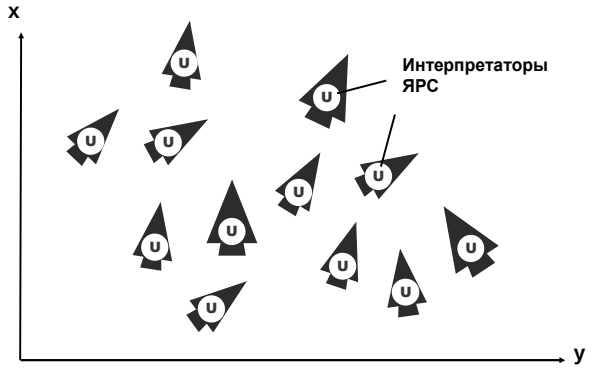

Рис. 16. Начальное группирование беспилотных единиц

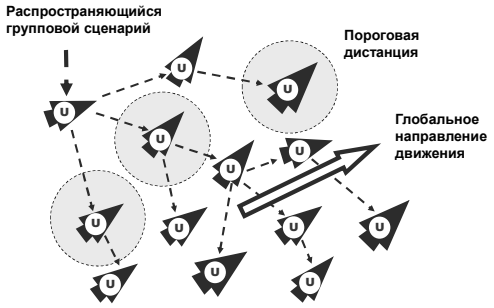

Рис. 17. Групповое глобально направленное случайное движение

```
прыжок (все вершины); вершинная (Пределы);
Пределы = (dx(0,8), dy(-2,5)); Глубина = 500;
повторять (
  Сдвиг = случайный выбор (Пределы);
  если (пусто (прыжок (Сдвиг, Глубина)),
        переместиться (Сдвиг)))
```

Моментный снимок движения группы по этому сценарию изображен на рис. 17.

Рассмотрим следующее функциональное обогащение группы, где она самостоятельно определяет свой топологически центральный робот в некоторый момент времени (этот центр может постоянно меняться в связи с локально случайным движением роботов). Последующий сценарий с результирующим общим значением в виде адреса центра (назовем его ***нахождение_центра***) может начаться (как и предыдущий) с любой системной единицы:

```
Фронтальная (Средняя);
Средняя = усреднить (прыжок (все вершины); ГДЕ);
минимум (
  прыжок (все вершины); дистанция (Средняя, ГДЕ) & АДРЕС) : 2
```

На рис. 18 показана найденная таким образом центральная единица.

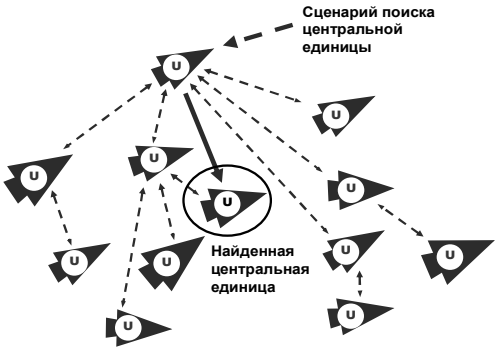

Рис. 18. Самоопределение перемещающейся группой своего топологического центра

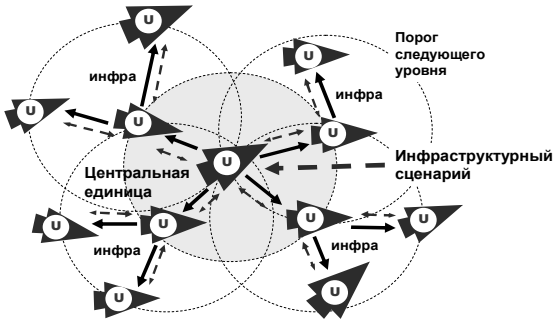

Рис. 19. Создание иерархической инфраструктуры, базирующейся на найденном центре

Найдя топологический центр, начиная с него, мы можем построить иерархическую инфраструктуру, оптимальную для глобального управления и всевозможных совместных операций. Такой сценарий (назовем его ***построение_инфраструктуры***), начавшись в найденном центре, будет выглядеть следующим образом (рис. 19):

```
фронтальная (Глубина = 20);
оставаться (
  повторять (соединить (дуга (+ инфра), первым пришел, Глубина)))
```

Создав оптимальную пространственную инфраструктуру, мы теперь можем максимально упростить целый ряд групповых операций, например, распределенный сбор всех видимых целей (от всех роботов, где каждый может видеть только их ограниченное число из-за ограниченной глубины доступа сенсоров) с их последующим обратным распределением между всеми беспилотниками, где каждый затем сможет выбрать, преследовать и, если нужно, атаковать наиболее подходящие для него цели.

Такой сценарий, стартуя от центральной компоненты (назовем его ***сбор_распределение_атака***), будет иметь на ЯРС следующий вид (рис. 20):

```
фронтальная (Видимые);
цикл (
  непусто (Видимые = повторять (
    свободный (определить (цели)), прыжок (дуги (все, + инфра))));
  повторять (
    свободный (выбрать_переместиться_атаковать (Видимые)),
    прыжок (все, + инфра)))
```

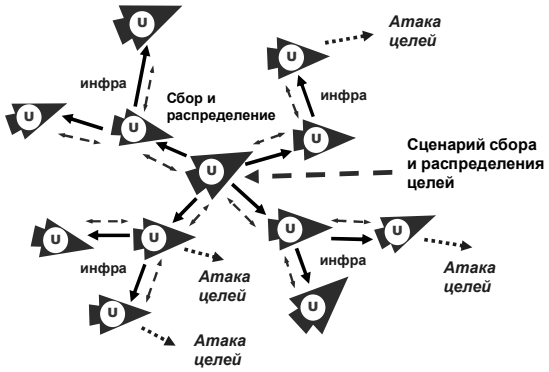

Рис. 20. Иерархический сбор и распределение целей с автономным отбором для возможной атаки

Поскольку в нашем случае роботические единицы движутся в пространстве с определенной, заданной, долей вероятности, для сохранения оптимальности пространственного управления центральную единицу и исходящую от нее иерархическую инфраструктуру целесообразно регулярно переопределять, для чего следует удалять прежнюю. Это достигается следующей программой, начинающейся в существующей центральной единице (назовем ее ***удаление_инфраструктуры***):

```
оставаться (
    прыжок (вершины, все);
    удалить (дуги, все))
```

Результирующее комбинированное решение, объединяющее групповое случайное движение с регулярным обновлением топологического центра и переопределением исходящей от него управленческой инфраструктуры, а также базирующиеся на ней распределенный сбор, обратное распределение и параллельная атака целей, может быть легко получено путем елементарного объединения предыдущих решений в цельную, холистскую программу, начинающуюся с любого робота:

```
движение_группы,
повторять (
  прыжок (нахождение_центра);
  удаление_инфраструктуры; построение_инфраструктуры;
  параллельное или (сбор_распределение_атака, спать (360)))
```

Предложенная технология успешно опробована также на множестве других «горячих» проблем (частично упомянутых во Введении) – от гражданских до военных.

8. Заключение

Мы представили радикально новый подход к системному видению, организации и глобальному управлению, базирующийся на задании основ системной организации и целевой ориентации на специальном языке высокого уровня, который может эффективно интерпретироваться в распределенных средах в высокопараллельном режиме. Разработанная технология позволяет напрямую схватывать целостность сложных динамических систем, переводя многочисленные традиционные системные рутины на автоматизированный или даже полностью автоматический уровень. Вирусоподобное (в положительном смысле этого слова, имея в виду массовость, распределенность, параллельность и живучесть) воплощение этого метода в сложных распределенных системах самого разного характера – от гражданских до военных – может существенно поднять их производительность и способность самовосстанавливаться после произвольных сбоев и разрушений.

СПИСОК ЛИТЕРАТУРЫ

1. Minsky M. The Society of Mind. Simon and Schuster / Minsky M. – New York, 1988. – 336 p.
2. Sapaty P. Distributed Technology for Global Control / P. Sapaty // Book chapter, Lecture Notes in Electrical Engineering. – 2009. – Vol. 37. – Part 1, 3-24, DOI: 10.1007/978-3-642-00271-7_1. – P. 3 – 24.
3. Sapaty P. Ruling Distributed Dynamic Worlds / Sapaty P. – New York: John Wiley & Sons, 2005. – 254 p.
4. Sapaty P. Mobile Processing in Distributed and Open Environments / Sapaty P. – New York: John Wiley & Sons, 1999. – 450 p.

5. Wertheimer M. Gestalt Theory / Wertheimer M. – Erlangen, Berlin, 1925. – 345 p.
6. Sapaty P. Gestalt-based ideology and technology for spatial control of distributed dynamic systems" / P. Sapaty // Neuroscience meets Gestalt psychology // Proc. International Gestalt Theory Congress, 16th Scientific Convention of the GTA. – Germany: University of Osnabruck, 2009. – March 26–29. – 2 p.
7. Wilber K. Waves, Streams, States and Self: A Summary of My Psychological Model (Or, Outline of An Integral Psychology) / Wilber K. – Shambhala Publications, 2009. – 60 p.
8. Rogers C.R. Carl Rogers on Personal Power: Inner Strength and Its Revolutionary Impact / Rogers C.R. – USA: Trans-Atlantic Publications, 1978. – 305 p.
9. Freud S. General Psychological Theory/ Freud S. – New York: Touchstone, 1997. – 224 p.
10. Sapaty P. Dynamic Air Traffic Management Using Distributed Brain Concept / P. Sapaty, V. Klimenko, M. Sugisaka // Mathematical machines and systems. – 2004. – N 1. – P. 3 – 8.
11. Sapaty P. Dynamic Air Traffic Management Using Distributed Brain Concept / P. Sapaty, V. Klimenko, M. Sugisaka // Proc. Ninth International Symposium on Artificial Life and Robotics (AROB 9th). – Beppu, Japan, 2004. – P. 156 – 159.
12. A New Concept of Flexible Organization for Distributed Robotized Systems / P. Sapaty, A. Morozov, R. Finkelstein [et al.] // Proc. Twelfth International Symposium on Artificial Life and Robotics (AROB 12th'07). – Beppu, Japan, 2007. – Jan 25–27. – 8 p.
13. Sapaty P. DEW in a Network Enabled Environment / P. Sapaty, A. Morozov, M. Sugisaka // Proc. of the international conference Directed Energy Weapons 2007. – London, UK: Le Meridien Piccadilly, 2007. – Feb. 28 – March 1. – 81 slide.
14. European Patent N 0389655. A Distributed Processing System / Sapaty P.; Publ. 10.11.93.

Стаття надійшла до редакції 29.10.2010